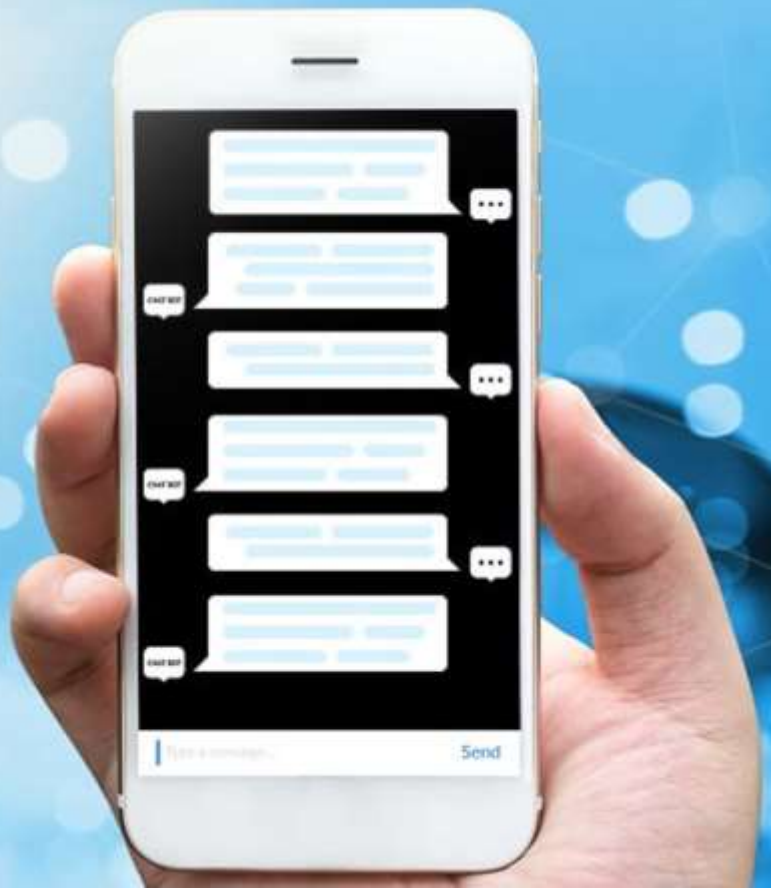
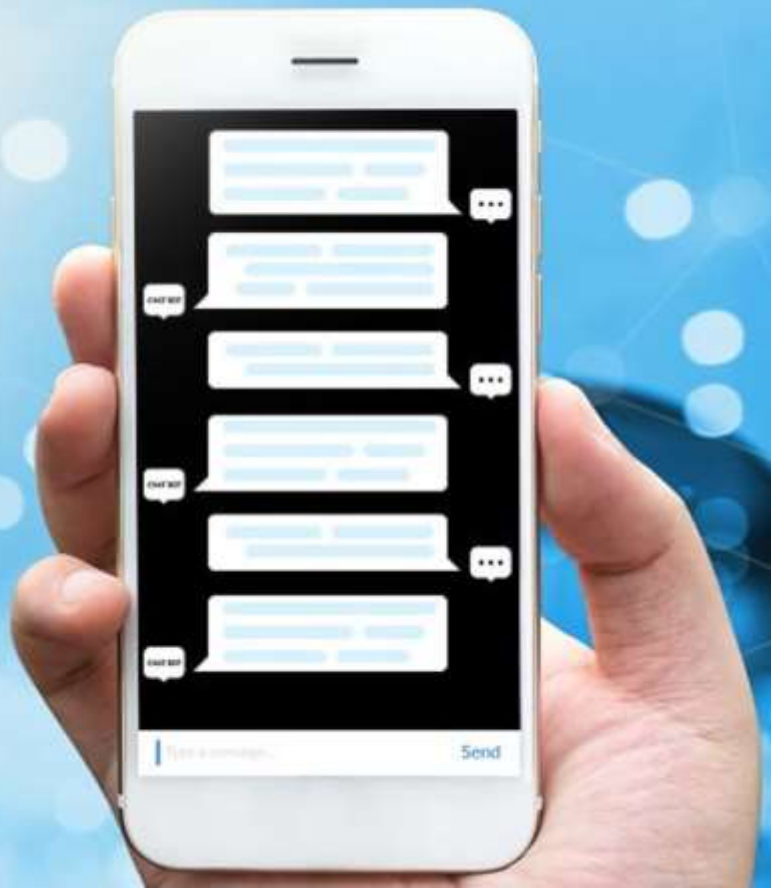
AI기술을 이용한 이미지학습과 자연어처리를 통한 챗봇개발과정

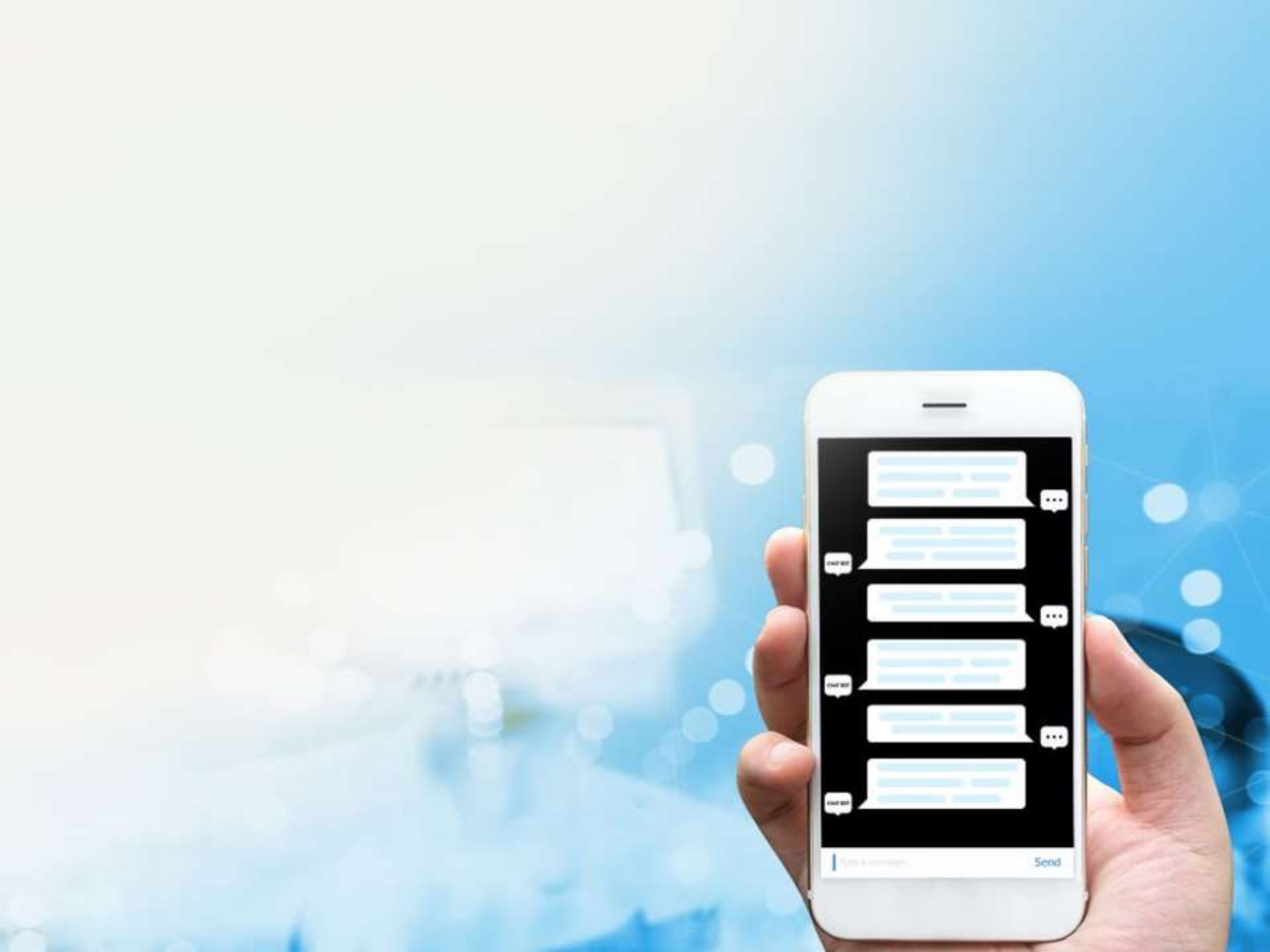


LIMA

발표제목

2019.01.22




Send

전체 목차

사전 준비 사항-컴퓨터 작업 환경

- Windows 10(64비트)
- Chrome Ver 71.0.3578.98
- Python Ver 3.5.4
- Tensorflow Ver 1.12.0
- Jupyter Ver 1.0.0

1일차

기본 환경 설치 및 사용법

목차

파이썬(Python)

- Python 이란?
  - 1990년 암스테르담의 귀도 반 로섬에 의해 개발된 고급 프로그래밍 언어로, 현재까지 실무와 교육에서 큰 인기를 끌고 있다.
  - 전 세계에서 교육 뿐만 아니라 실무에서도 많이 사용하며, 구글이나 인스타그램, 드롭박스 등에서 사용하고 있다.
  - 파이썬 프로그램은 배우기 쉽고, 속도가 빠르며, 공동 작업과 유지 보수가 편리하다. 이러한 이유 때문에 이미 다른 언어로 만들어진 많은 프로그램과 모듈들이 파이썬으로 재구성되고 있으며, 최근에는 사용자 층이 늘어가는 추세이다.

파이썬(Python)

- Python 특징(1/2)
  - **인터프리터(Interpreter) 언어**

    C, Java와 같은 언어는 컴퓨터가 이해할 수 있는 소스코드를 별도의 실행 파일로 바꾸는 작업(compile)을 거쳐야 하는 언어다. 하지만 파이썬(python)은 인터프리터 언어에 속해 컴파일 없이 코드를 바로 실행할 수 있는 언어로 장점을 가지고 있다.

  - **인덴트(Indent)**

    파이썬은 다른 프로그래밍 언어에 비해서 인덴트(띄어쓰기, 공백) 매우 민감한 언어다. 일반적인 다른 프로그래밍 언어는 중괄호 `{}`를 이용해서 블록(범위)을 표현하는 반면, 파이썬은 공백을 이용 한다. 인덴트에 대해 신경 쓰지 않으면 에러가 자주 발생한다.

파이썬(Python)

- Python 특징(2/2)
  - **웹 프로그래밍**

    Django, Flask라는 오픈 소스 기반의 웹 프레임워크를 사용하여 초보 개발자도 쉽게 웹 개발을 할 수 있다.

  - **빅 데이터 분석**

    사람의 힘으로 처리하기 힘든 대용량 데이터를 분석하고 데이터 간의 패턴을 알려주고, 이를 시각화해서 보여 준다. 그리고 기계학습, 인공지능 개발에 이용 되기도 한다.

파이썬(Python)

- Python 사용 현황

TIOBE programming language index(2018.4)

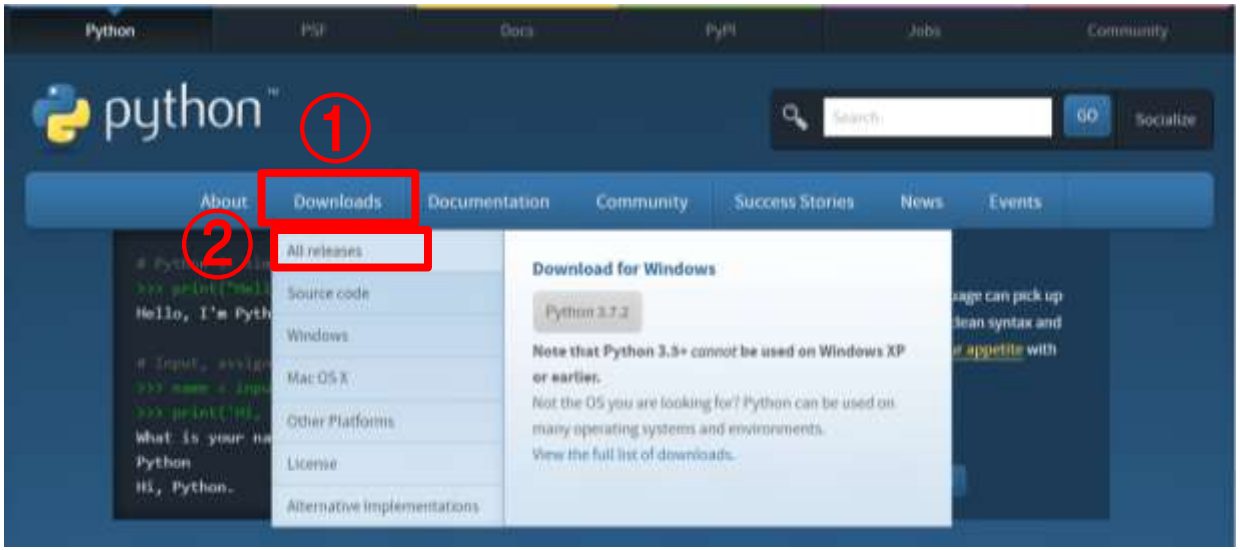
Apr 2018	Apr 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.777%	+0.21%
2	2		C	13.569%	+6.62%
3	3		C++	7.218%	+2.66%
4	5	^	Python	5.803%	+2.35%
5	4	v	C#	5.265%	+1.69%
6	7	^	Visual Basic .NET	4.947%	+1.70%
7	6	v	PHP	4.218%	+0.64%
8	8		JavaScript	3.492%	+0.64%
9	-	^	SQL	2.650%	+2.65%
10	11	^	Ruby	2.016%	-0.29%
11	9	v	Delphi/Object Pascal	1.961%	-0.86%
12	15	^	R	1.606%	-0.33%
13	16	^	Visual Basic	1.796%	-0.26%
14	13	v	Assembly language	1.655%	-0.51%

RedMonk Programming Language Rankings(2018.1)

Jan 2018	Jan 2017	Change	Programming Language
1	1	–	JavaScript
2	2	–	Java
3	3	–	Python
4	4	–	PHP
5	5	–	C#
6	5	↓	C++
7	7	–	CSS
8	7	↓	Ruby
9	9	–	C
10	11	↑	Swift

파이썬(Python)

- Python 설치 방법(1/5)
  - Python 공식사이트 : https://www.python.org
  - ① Downloads -> ② All releases 클릭



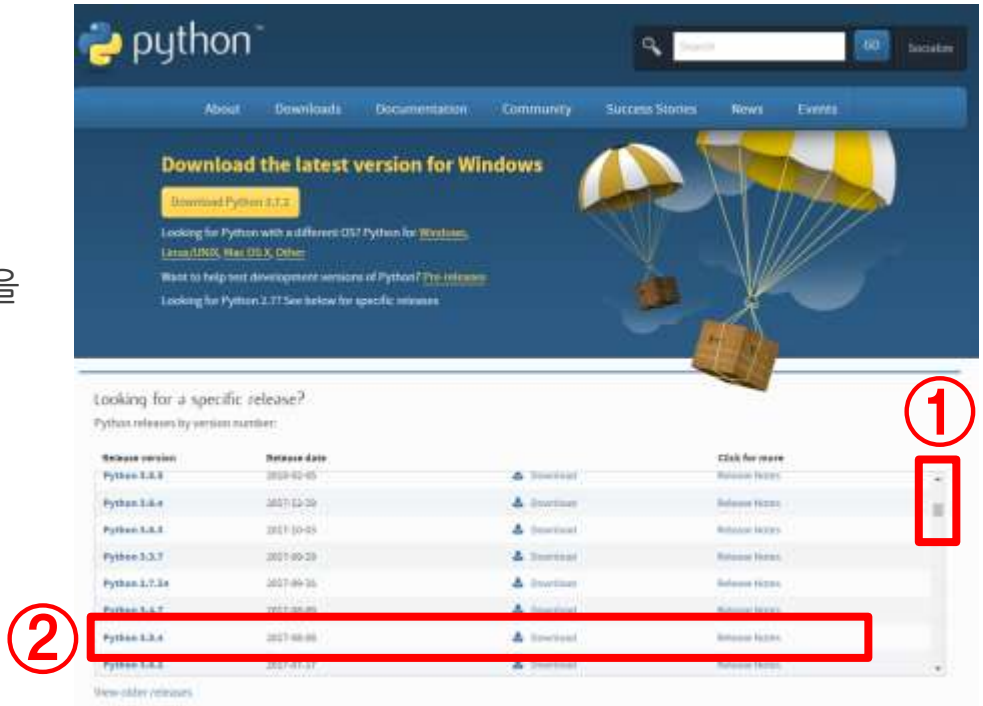
파이썬(Python)

- Python 설치 방법(2/5)

① 스크롤 내린 후

② Python 3.5.4 2017-08-08
Download 클릭

※ Tensorflow는 python 3.7 버전 지원을 하지 않습니다.



LIMA

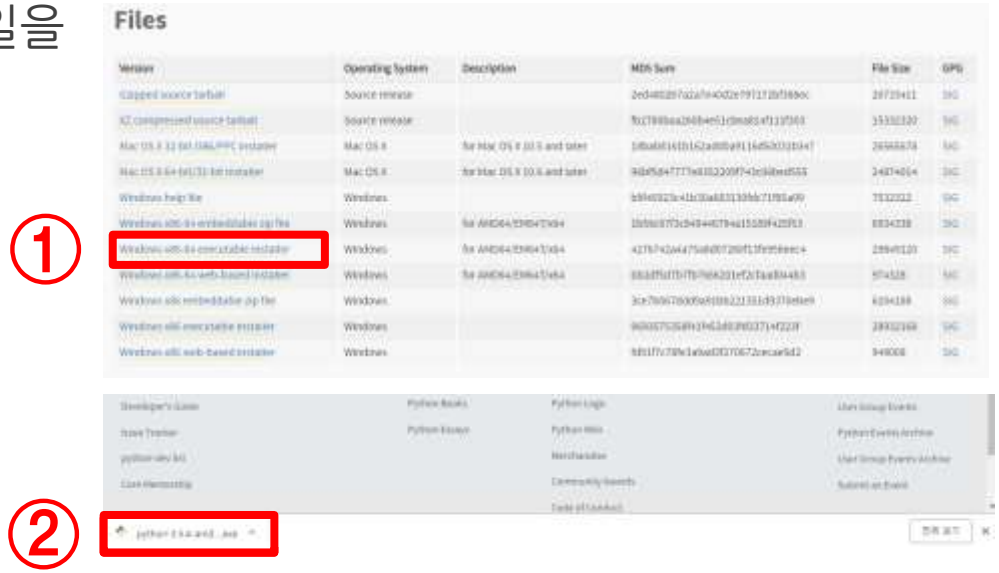
python

파이썬(Python)

- Python 설치 방법(3/5)

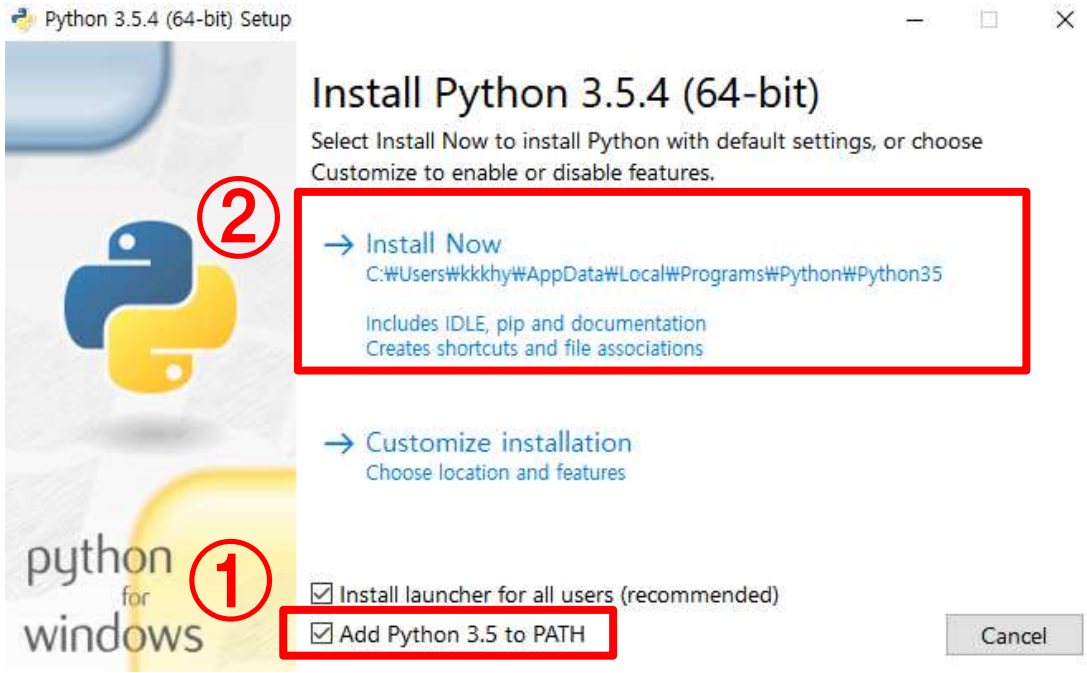
① 화면 아래쪽에
Windows x86-64 executable installer 클릭하여 다운로드

② 다운로드 한 파일을
클릭 후 설치



파이썬(Python)

- Python 설치 방법(4/5)
  ① Add Python 3.5 to PATH 클릭
  ② Install Now 클릭



파이썬(Python)

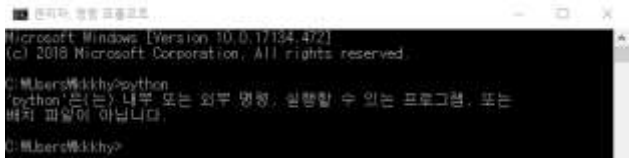
- Python 설치 방법(5/5)

  설치 완료 후 ① > ② 명령 프롬프트(CMD)

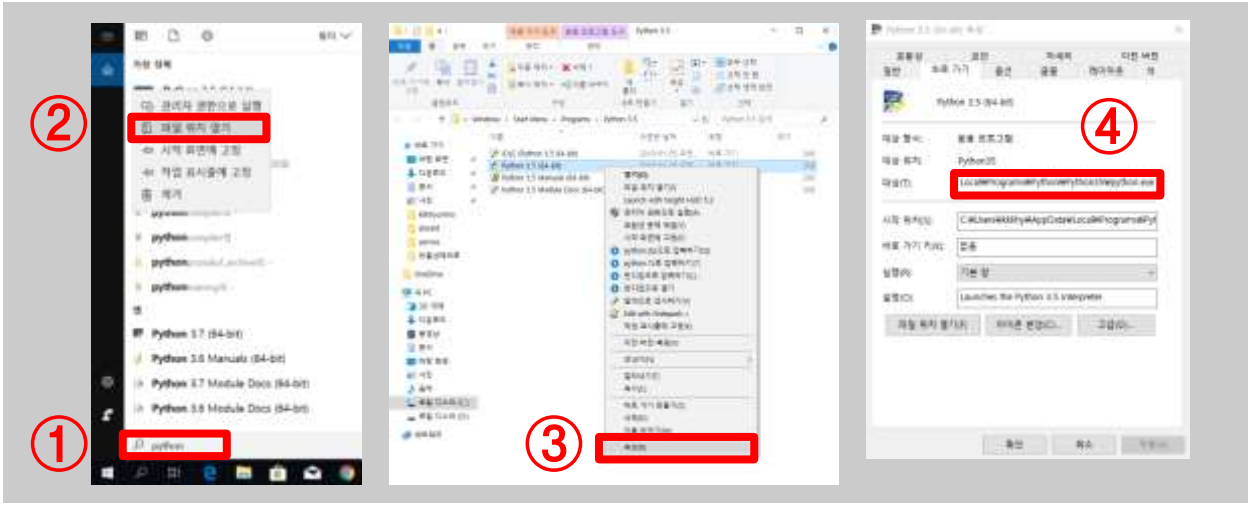
  ③ python -V 입력 후 Python 3.5.4 나오면 설치 성공

파이썬(Python)

- 환경 변수 설정(1/4)
  - 우측 그림처럼 Python 명령어가 안될 때 시스템 환경 변수를 편집 해야 한다.
    우선 Python 설치 경로를 찾는다.

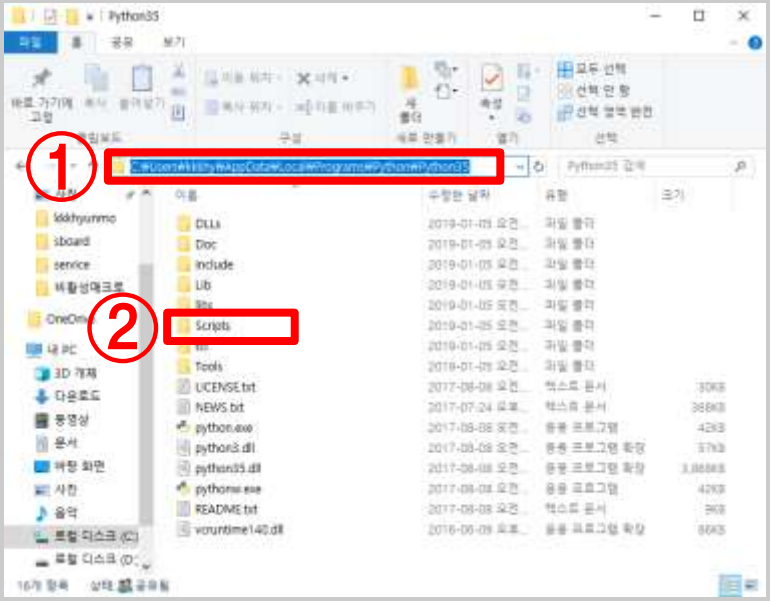


  - ①~③번의 과정을 거친 후 ④번의 경로(Python35₩까지) 복사



파이썬(Python)

- 환경 변수 설정(2/4)
  - 복사한 경로로 접근 하면 폴더와 파일들이 나온다.



- **python명령어 위해**

① C:₩..₩..₩Python₩Python35₩

- **pip명령어를 위해**

② C:₩..₩..₩Python₩Python35₩Scripts₩

①, ② 두 경로를 메모장에 적어 둔다.

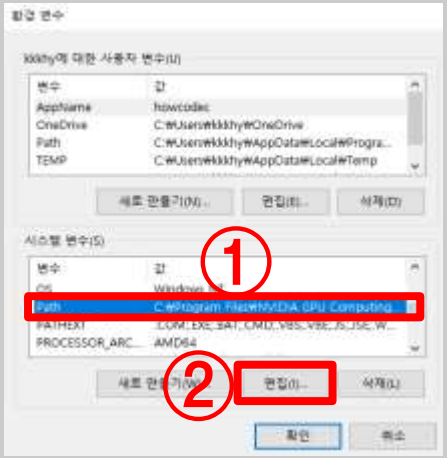
파이썬(Python)

- 환경 변수 설정(3/4)

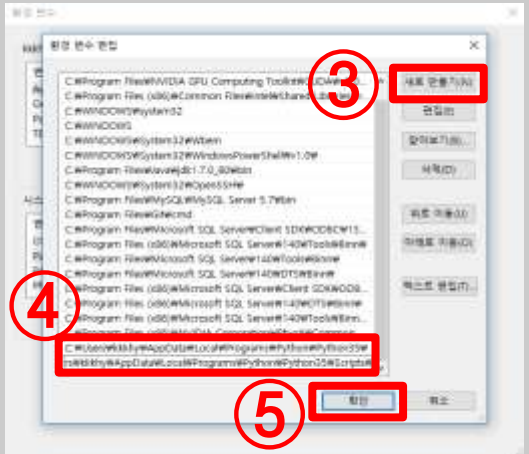
파이썬(Python)

- 환경 변수 설정(4/4)

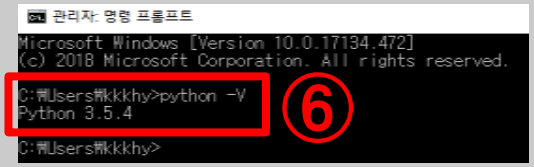
①시스템 변수 Path값 클릭
②편집을 클릭



③새로 만들기 클릭, 메모장에 있는 파이썬 설치 경로 2개를 ④에 추가하고, ⑤확인을 클릭



⑥명령 프롬프트 창(CMD)을 반드시 다시 실행해서 오른쪽 그림과 같이 나오면 세팅 완료



LIMA

python

텐서플로(Tensorflow)

- Tensorflow란?
  - 기계 학습과 딥러닝을 위해 구글에서 만든 오픈 소스 라이브러리이다. Android와 iOS 같은 모바일 환경과 64비트 리눅스, MacOS 또는 데스크 탑 등에서 CPU와 GPU 환경으로 구동될 수 있다.
  - 데이터 플로우 그래프 방식을 사용하며, 수학 계산과 데이터의 흐름을 노드와 엣지를 이용해서 방향 그래프(Directed Graph)로 표현 한다.
  - 노드(Node): 수학적 계산, 데이터 입/출력, 읽기/저장 등의 작업을 수행한다.
  - 엣지(Edge): 노드들 간 데이터의 입출력 관계를 나타낸다.

텐서플로(Tensorflow)

- Tensorflow 특징
  - 데이터 플로우 그래프를 통한 풍부한 표현력
  - 코드 수정 없이 CPU/GPU 모드로 동작
  - 아이디어 테스트에서 서비스 단계까지 이용 가능
  - 계산 구조와 목표 함수만 정의하면 자동으로 미분 계산을 처리
  - Python/C++를 지원하며, SWIG를 통해 다양한 언어 지원 가능

텐서플로(Tensorflow)

- 가상환경 구축
  - 가상 환경이(Virtualenv)란?
    하나의 컴퓨터에서 여러 개의 프로젝트를 개발할때 라이브러리들을 프로젝트별로 각각의 디렉토리에 격리시켜 관리하는 것을 말한다.
  - 가상환경 생성하기 : **python -m venv venv**
  - 가상환경 활성화하기 : **venv₩Scripts₩Activate**
  - 가상환경 비활성화하기 : **Deactivate**

```
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:₩Users₩phm>cd ../../

C:₩>python -m venv venv

C:₩>venv₩scripts₩activate
(venv) C:₩>deactivate
C:₩>
```

텐서플로(Tensorflow)

- Tensorflow 설치
  - mkdir chatbot → cd chatbot

```
(venv) C:₩>mkdir chatbot

(venv) C:₩>cd chatbot
```

  - pip install tensorflow 명령어로 설치

```
(venv) C:₩chatbot>pip install tensorflow
```

  - 아래와 같이 Successfully가 나오면 설치 완료

```
Successfully installed absl-py-0.6.1 astor-0.7.1 gast-0.2.0 grpcio-1.17.1 h5py-2.9.0 keras-applications-1.0.6 keras-preproce
1.0.5 markdown-3.0.1 numpy-1.15.4 protobuf-3.6.1 six-1.12.0 tensorboard-1.12.2 tensorflow-1.12.0 termcolor-1.1.0 werkzeug-0
heel-0.32.3
```

  - python 입력 후 import tensorflow as tf를 입력하여 확인

```
(venv) C:₩chatbot>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>>
```

  - 오류가 나지 않으면 설치 완료

Jupyter notebook

- Jupyter Notebook은 오픈 소스 웹 애플리케이션으로 라이브 코드, 등식, 그래프, 시각화 설명을 위해 텍스트 등을 포함한 문서를 만들고 공유할 수 있다.

- 파이썬, R, Scala 등 데이터 과학 분야에서 인기 있는 프로그래밍 언어이며, 주로 데이터 클리닝과 변형, 수치 시뮬레이션, 통계 모델링, 머신 러닝 등에 사용할 수 있다.

- 가장 큰 장점은 실시간으로 인터랙티브(interactive)한 데이터를 만들고, 시각화할 수 있다.

Jupyter notebook 주피터 노트북 설치

- pip install jupyter 명령어로 설치

```
(venv) C:₩chatbot>pip install jupyter
```

- 아래 부분 처럼 Successfully built ~ 나오면 설치 완료

```
  Running setup.py bdist_wheel for pandocfilters ... done
  Stored in directory: C:₩Users₩kkkhy₩AppData₩Local₩pip₩Cache₩wheels₩39₩01₩56₩f1b08a6275acc59e846fa4c1e1b65dbc1919f20157d9e66c20
  Running setup.py bdist_wheel for prometheus-client ... done
  Stored in directory: C:₩Users₩kkkhy₩AppData₩Local₩pip₩Cache₩wheels₩1a₩74₩d7₩dc59e0bf44fdfd6395c0076129453abf563e4aeca5d72c8574
  Running setup.py bdist_wheel for backcall ... done
  Stored in directory: C:₩Users₩kkkhy₩AppData₩Local₩pip₩Cache₩wheels₩98₩b0₩dd₩29e28ff615af3dda4c67cab719dd51357597eabff926976b45
  Running setup.py bdist_wheel for win-unicode-console ... done
Successfully built pandocfilters prometheus-client backcall win-unicode-console
Installing collected packages: ipython-genutils, decorator, traitlets, jupyter-core, tornado, python-dateutil, pyzmq, jupyter-clie
nt, pygments, backcall, colorama, wcwidth, prompt-toolkit, win-unicode-console, pickleshare, parso, jedi, ipython, ipykernel, qtco
nsole, jupyter-console, mistune, jsonschema, nbformat, entrypoints, testpath, defusedxml, pandocfilters, MarkupSafe, jinja2, weben
codings, bleach, nbconvert, prometheus-client, Send2Trash, pywinpty, terminado, notebook, widgetsnbextension, ipywidgets, jupyter
Successfully installed MarkupSafe-1.1.0 Send2Trash-1.5.0 backcall-0.1.0 bleach-3.0.2 colorama-0.4.1 decorator-4.3.0 defusedxml-0.5
.0 entrypoints-0.2.3 ipykernel-5.1.0 ipython-7.2.0 ipython-genutils-0.2.0 ipywidgets-7.4.2 jedi-0.13.2 jinja2-2.10 jsonschema-2.6.
0 jupyter-1.0.0 jupyter-client-5.2.4 jupyter-console-6.0.0 jupyter-core-4.4.0 mistune-0.8.4 nbconvert-5.4.0 nbformat-4.4.0 noteboo
k-5.7.4 pandocfilters-1.4.2 parso-0.3.1 pickleshare-0.7.5 prometheus-client-0.5.0 prompt-toolkit-2.0.7 pygments-2.3.1 python-dateu
til-2.7.5 pywinpty-0.5.5 pyzmq-17.1.2 qtconsole-4.4.3 terminado-0.8.1 testpath-0.4.2 tornado-5.1.1 traitlets-4.3.2 wcwidth-0.1.7 w
ebencodings-0.5.1 widgetsnbextension-3.4.2 win-unicode-console-0.5
```

LIMA

jupyter

Jupyter notebook 주피터 노트북 실행하기

- Jupyter notebook을 입력해서 인터넷 창이 나오면 실행 완료

```
(venv) C:₩chatbot>jupyter notebook
[I 16:47:06.747 NotebookApp] Writing notebook server cookie secret to C:₩Users₩kkkhy₩AppData₩Roaming₩jupyter₩runtime₩notebook_cook
ie_secret
[I 16:47:07.341 NotebookApp] Serving notebooks from local directory: C:₩chatbot
[I 16:47:07.341 NotebookApp] The Jupyter Notebook is running at:
[I 16:47:07.341 NotebookApp] http://localhost:8888/?token=40f7161f1d181754ef43b731880a9d66d109a0948c2778da
[I 16:47:07.342 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:47:07.346 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/kkkhy/AppData/Roaming/jupyter/runtime/nbserver-22408-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=40f7161f1d181754ef43b731880a9d66d109a0948c2778da
```

Jupyter notebook 주피터 노트북 사용하기(1/3)

- 새로운 노트북 생성 : ① New -> ② Python 3 클릭

- 실습 예제 1. 아래의 사각형 내부의 코드를 타이핑후 Shift + Enter(실행 단축키)를 쳐보세요

In [1]:
```
x = 1       # x에 1저장
print(x)    # 1 출력
print(x+1) # x + 1 출력, 결과 값 2
y = 2       # y에 2 저장
print(x+y) # x + y 출력, 결과 값 3
```

```
1
2
3
```

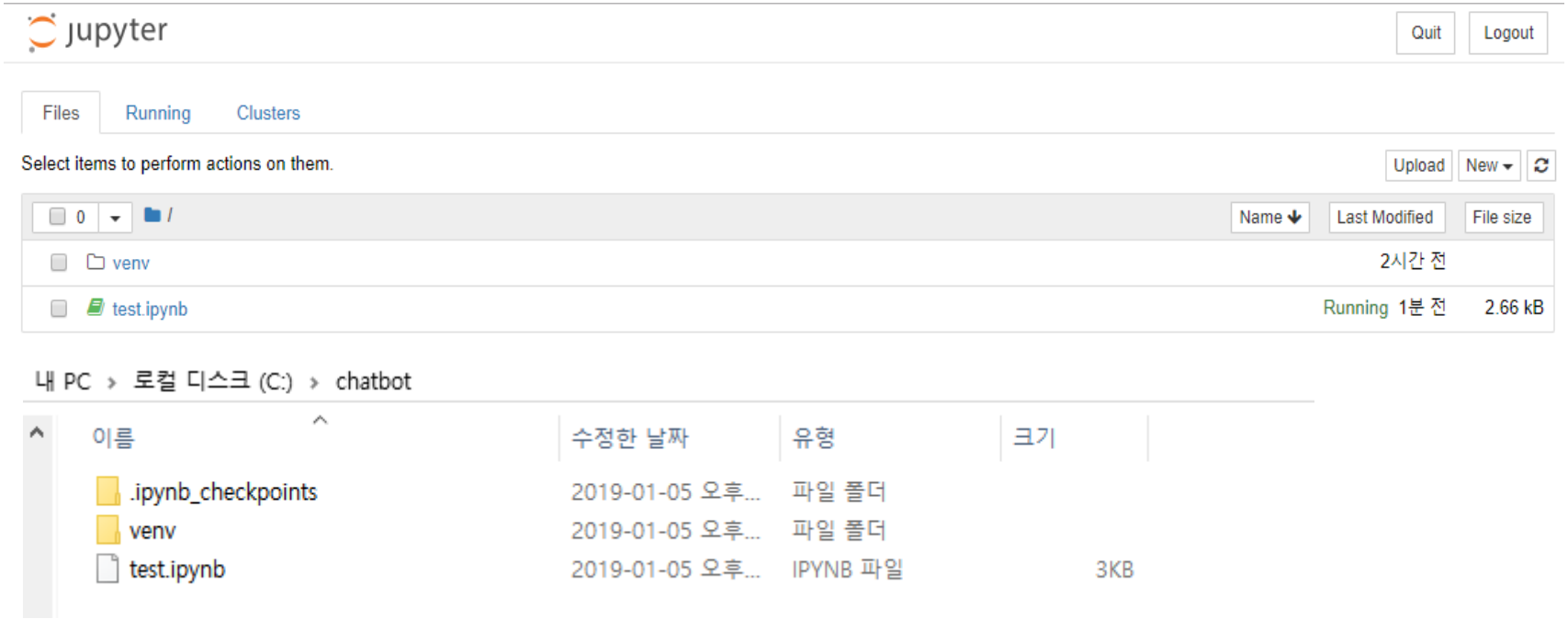
Jupyter notebook 주피터 노트북 사용하기(2/3)

- 실습 예제 2. 아래의 사각형 내부의 코드를 타이핑후 Shift + Enter(실행 단축키)를 쳐보세요

```
In [2]: import tensorflow as tf
        print(tf.__version__) # 텐서플로우 버전 확인

        1.12.0

In [3]: hello = tf.constant('Hello, TensorFlow!') # tf.constant로 상수를 hello 변수에 저장
        print(hello)                              # tensor 자료형이고 상수를 담고 있다.

        Tensor("Const:0", shape=(), dtype=string)

In [4]: a = tf.constant(10)
        b = tf.constant(32)
        c = tf.add(a, b)
        print(c)                                  # tensor 자료형이고 Add를 담고 있다.

        Tensor("Add:0", shape=(), dtype=int32)

In [5]: sess = tf.Session()                       # Session()으로 연결 후 run 메서드를 사용해야 출력 할 수 있다.
        print(sess.run(hello))
        print(sess.run([a, b, c]))

        b'Hello, TensorFlow!'
        [10, 32, 42]
```

Jupyter notebook 주피터 노트북 사용하기(3/3)

- 아래와 같이 .ipynb 확장자로 자동 저장 된다.



2일차

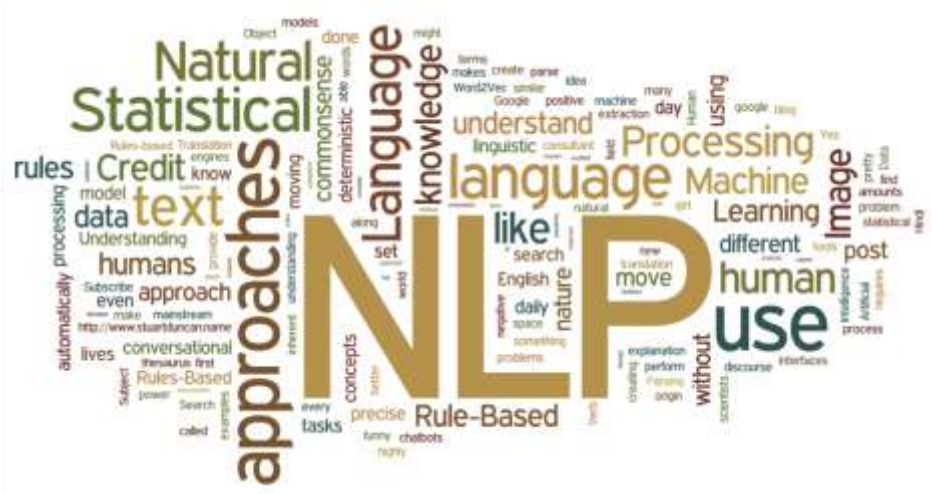
자연어처리

목차

자연어 처리란?

- Natural Language Processing

- 인간의 언어 현상을 컴퓨터와 같은 기계를 이용해서 모사 할 수 있도록 연구하고 이를 구현하는 **인공지능**의 주요 분야 중 하나다.



응용 분야

- 자동번역 (파파고, google translate)
- QA 시스템 (챗봇)
- 문서 자동 분류 (신문기사 클러스터링)
- 자동 요약 (뉴스, 웹 페이지 본문 요약)

자연어 처리 오픈 소스 툴

- NLTK(영어)
- KoNLPy (한국어)
- NLPIR(중국어)

NLTK(영어)

- Natural Language Toolkit
- Steven Bird와 Edward Loper 펜실베이니아 대학교에서 개발
- 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지
- 다양한 기능 및 예제
- 자연어 처리에 대한 실무 및 연구에서 많이 사용
- 오픈 소스
- Windows, Mac OS X, Linux 지원
- Python 버전 2.7, 3.4, 3.5, 3.6, or 3.7

주요 기능

- 말뭉치(corpus) 제공
- 토큰 생성(tokenizing)
- 형태소 분석(morphological analysis)

환경 설치

- NLTK 설치
  - cmd에서 pip install nltk 명령을 통해 설치

```
(venv) C:₩NLP>pip install nltk
Successfully installed nltk-3.4 singledispatch-3.4.0.3
```

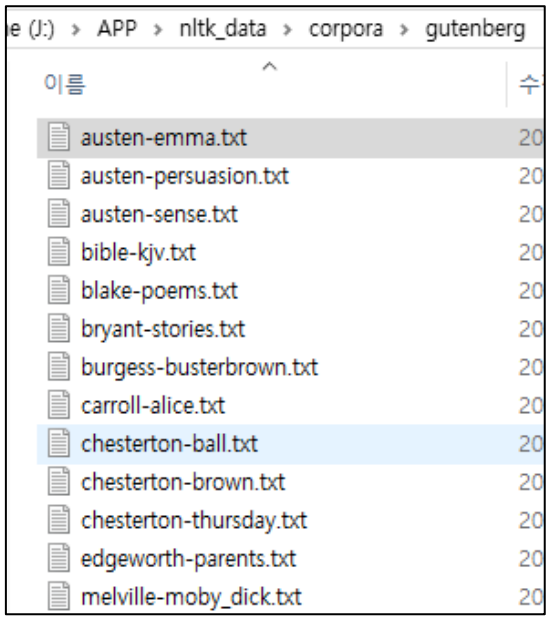
  - Jupyter notebook 실행 → 설치 확인 및 소설 다운로드

```
In [1]: import nltk

In [2]: nltk.download('gutenberg')
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:₩Users₩phm₩AppData₩Roaming₩nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
Out[2]: True

In [6]: nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data]     C:₩Users₩phm₩AppData₩Roaming₩nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Out[6]: True
```

말뭉치(corpus) 제공

- 자연어 분석 작업을 위해 만든 샘플 문서 집합
- NLTK 패키지에서 다양한 연구용 말뭉치를 제공
- Gutenberg말뭉치- 저작권이 말소된 문학작품을 포함

```
In  [2]: nltk.corpus.gutenberg.fileids()
Out[2]: [u'austen-emma.txt',
         u'austen-persuasion.txt',
         u'austen-sense.txt',
         u'bible-kjv.txt',
         u'blake-poems.txt',
         u'bryant-stories.txt',
         u'burgess-busterbrown.txt',
         u'carroll-alice.txt',
         u'chesterton-ball.txt',
         u'chesterton-brown.txt',
         u'chesterton-thursday.txt',
         u'edgeworth-parents.txt',
         u'melville-moby_dick.txt',
         u'milton-paradise.txt',
         u'shakespeare-caesar.txt',
         u'shakespeare-hamlet.txt',
         u'shakespeare-macbeth.txt',
         u'whitman-leaves.txt']
```



토큰 생성(tokenizing)

- 땅에서 농작물을 수확
- Document 에서 가치 있는 Token들을 수확

토큰 생성(tokenizing)

- 자연어 문서를 분석하기 위해 긴 문자열을 작은 단위로 분할.
- 토큰(token): 분할된 문자열(단어, 절, 문장 등)
- 토큰으로 나누는 작업을 토큰 생성(tokenizing)이라고 한다.
- sent_tokenize: 문장 단위로 분리

In [5]:
```
from nltk.tokenize import sent_tokenize
emma_raw = nltk.corpus.gutenberg.raw("austen-emma.txt")
print(sent_tokenize(emma_raw[:1000])[3])
```

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

형태소 분석(morphological analysis)

- 형태소(morpheme): 언어학에서 일정한 의미가 있는 가장 작은 말의 단위
- 자연어 처리에서는 토큰으로 형태소를 이용한다.
- 단어로부터 어근, 접두사, 접미사, 품사 등 다양한 언어적 속성을 파악하고 이를 이용하여 형태소를 찾아내거나 처리하는 작업이다.
- 형태소 분석의 예
  - 어간 추출(stemming)
  - 원형 복원(lemmatizing)
  - 품사 태깅(Part-Of-Speech tagging)

어간 추출(stemming)

- 변화된 단어의 접미사나 어미를 제거하여 같은 의미를 가지는 형태소의 기본형을 찾는 방법이다. 단순히 어미를 제거하기 때문에 단어의 원형을 정확히 찾아 주지 않는다.

- PorterStemmer: Martin Porter가 제시한 알고리즘이다.

```
In [1]: words = ['lives', 'crying', 'flies', 'dying']

In [2]: # 원형 복원
        from nltk.stem import WordNetLemmatizer
        lm = WordNetLemmatizer()
        [lm.lemmatize(w) for w in words]

Out[2]: ['life', 'cry', 'fly', 'dying']
```

원형 복원(lemmatizing)

- 같은 의미를 가지는 여러 단어를 가장 근본적인 형태
  간단히 말해 사전형으로 통일하는 작업이다.

In [2]:
```
words = ['lives', 'crying', 'flies', 'dying']
```

In [3]:
```
# 원형 복원
from nltk.stem import WordNetLemmatizer
lm = WordNetLemmatizer()
[lm.lemmatize(w) for w in words]
```

Out[3]:
```
['life', 'cry', 'fly', 'dying']
```

In [4]:
```
# 동사로 복원
lm.lemmatize("dying", pos="v")
```

Out[4]:
```
'die'
```

품사 태깅 (POS tagging)

- 품사(POS, part-of-speech)는 낱말을 문법적인 기능이나 형태, 뜻에 따라 구분한 것

- NLTK에서는 펜 트리뱅크 태그세트(Penn Treebank Tagset)를 이용한다.
  - NNP: 단수 고유명사
  - VB: 동사
  - VBP: 동사 현재형
  - TO: to 전치사
  - NN: 명사(단수형 혹은 집합형)
  - DT: 관형사

품사 태깅 (POS tagging)

- 품사 태깅 예시

In [6]:
```
# 품사 태깅
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
sentence = "Emma refused to permit us to obtain the refuse permit"
tagged_list = pos_tag(word_tokenize(sentence))
tagged_list
```

Out[6]:
```
[('Emma', 'NNP'),
 ('refused', 'VBD'),
 ('to', 'TO'),
 ('permit', 'VB'),
 ('us', 'PRP'),
 ('to', 'TO'),
 ('obtain', 'VB'),
 ('the', 'DT'),
 ('refuse', 'NN'),
 ('permit', 'NN')]
```

과제

1. Tokenizing
   Nltk 말뭉치 혹은 직접 모은 txt파일을
   단어 단위로 분리하여 txt파일에 저장하기

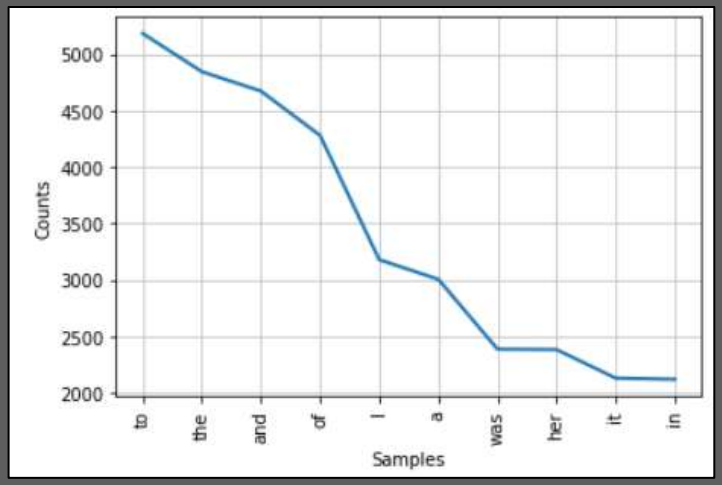
2. Tagging
   단어 단위로 분리한 토큰에 품사를 표시하여 txt파일에 저장하기

3. Extract noun
   명사만 추출하여 txt파일에 저장하기

과제

4. 빈도가 높은 단어 20개를 그래프로 그리기.

   Y축은 단어가 나타난 횟수,
   X축은 빈도가 높은 단어



5. 출현 횟수가 높은 5개 단어와 그 횟수 출력하기(선택)

```
[(('의', 'J'), 396),
 (('.', 'S'), 340),
 (('하', 'X'), 291),
 (('에', 'J'), 283),
 (('ㄴ다', 'E'), 241)]
```

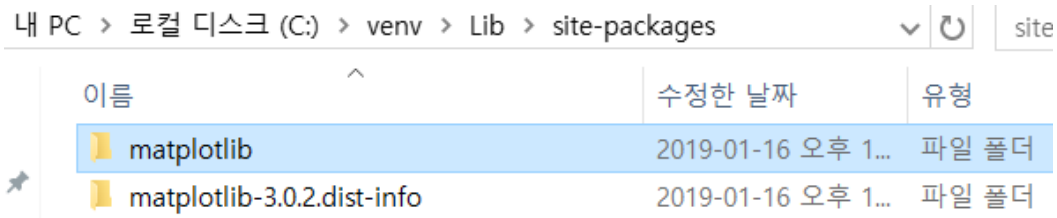
2일차 부록 - 환경 설치

- matplotlib 설치(2가지 방법)
  - Matplotlib는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지이다.

    ✓ Cmd에서 pip install matplotlib 명령을 통해 설치

    ```
    (venv) C:₩NLP>pip install matplotlib
    ```

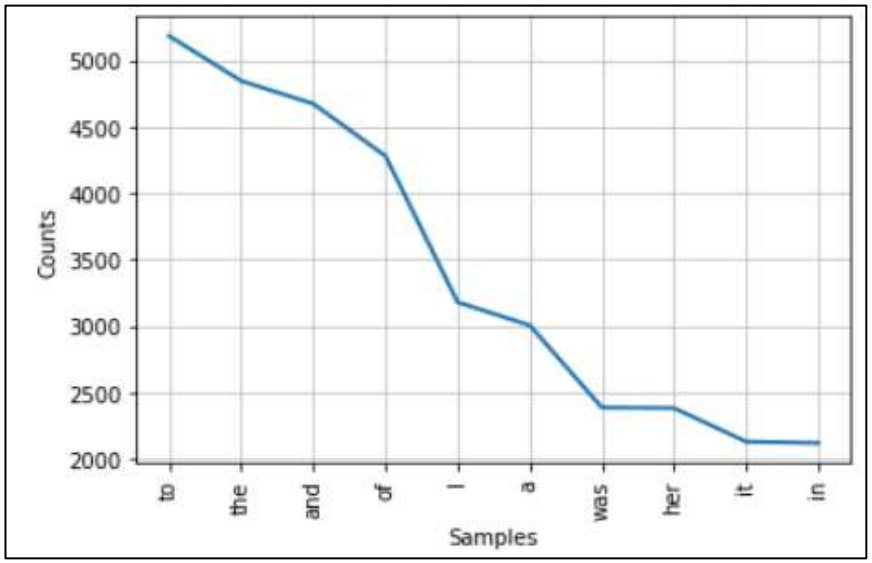
    ✓ 아래 사이트에서 matplotlib 파일을 다운 받고 압축 풀고 C:₩venv₩Lib₩site-packages에 복사한다.
    https://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib



2일차 부록 - 과제 설명

- 빈도가 높은 단어를 그래프로 그리기(선택)
  - NLTK Text클래스의 plot 메서드를 사용하여 단어의 빈도를 그래프로 그릴 수 있다.

```
In [16]: # (선택) 빈도 높은 단어 10개
from nltk import Text
import matplotlib.pyplot as plt
text = Text(retokenize.tokenize(emma_raw), name="Emma")
text.plot(10)
plt.show()
```



2일차 부록 - 과제 설명

- 출현 횟수가 높은 단어 출력하기(선택)
  - most_common 메서드를 사용하여 출현 횟수가 가장 높은 단어를 찾을 수 있다.

```
# (선택) 출현 횟수가 높은 단어 출력하기
from collections import Counter
from konlpy.corpus import kolaw
from konlpy.tag import Hannanum
from konlpy.utils import pprint

doc = kolaw.open('constitution.txt').read()
pos = Hannanum().pos(doc)
cnt = Counter(pos)

pprint(cnt.most_common(5))
```

```
[(('의', 'J'), 396),
 (('.', 'S'), 340),
 (('하', 'X'), 291),
 (('에', 'J'), 283),
 (('ㄴ다', 'E'), 241)]
```

3일차

KoNLPy

목차

3. KoNLPy

- 한국어 정보 처리를 위한 파이썬 패키지
- 한국어 연구에 많이 사용
- Windows, Mac OS X, Linux 지원

주요 기능

- 한국어 말뭉치
- 형태소 분석 및 품사 태깅

환경 설치

- **Java JDK 설치(1/2)**

  ✓ Windows x64, 버전 1.7 이상

  ✓ JDK 8 다운로드

  - 구글 에서 < jdk 다운로드 > 입력
  - Java SE Development Kit 8 - Downloads - Oracle 클릭

환경 설치

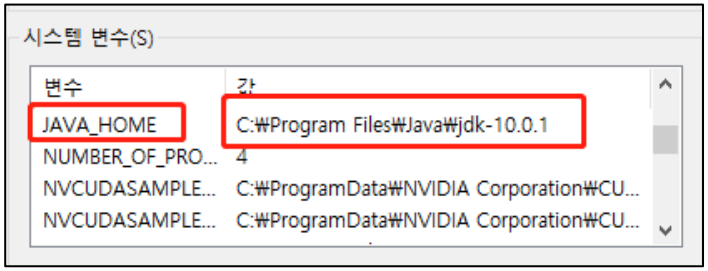
- Java JDK 설치(2/2)
  - ① Accept License Agreement
  - ② jdk-8u201-windows-x64.exe 클릭
  - ③ 다운로드 파일 기본 값으로 설치

설치 완료 화면

환경 설치

- JAVA_HOME 설정

✓ 내 PC 우클릭 → 속성 →
고급 시스템 설정 → 환경 변수
→ 시스템 변수에 JAVA_HOME,
Jdk 설치 파일 경로를 추가한다.



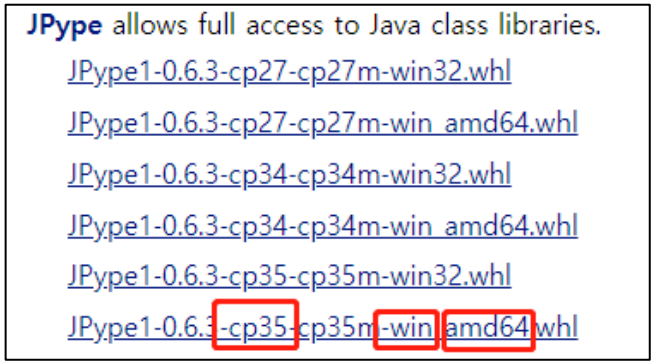
✓ cmd창을 새로 실행 해
설치 확인: java -version

```
J:₩APP>java -version
java version "10.0.1" 2018-04-17
Java(TM) SE Runtime Environment 18.3 (build 10.0.1+10)
Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.1+10, mixed mode)
```

환경 설치

- JPype1 설치

  ✓ 64 bit JPype 파일을 다운 한다.
  https://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype



  ✓ 다운로드 한 파일을 NLP 폴더로 가져 온 후
  ✓ pip install을 통해 설치

```
 C:₩NLP 디렉터리

2019-01-16  오후 04:06    <DIR>          .
2019-01-16  오후 04:06    <DIR>          ..
2019-01-16  오후 01:59    <DIR>          .ipynb_checkpoints
2019-01-16  오후 04:06           194,775 JPype1-0.6.3-cp35-cp35m-win_amd64.whl
2019-01-15  오후 03:40             7,583 konlpy.ipynb
2019-01-16  오후 02:01            27,779 nltk.ipynb
```

```
(venv) C:₩NLP>pip install JPype1-0.6.3-cp35-cp35m-win_amd64.whl
```

LIMA

환경 설치

- KoNLPy 설치

  ✓ pip install konlpy 명령어로 설치

```
(venv) C:₩NLP>pip install konlpy
```

  ✓ 설치 확인

```
(venv) C:₩NLP>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import konlpy
>>>
```

한국어 말뭉치

- 파일경로:

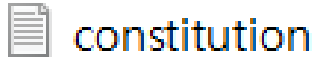
```
C:₩venv₩Lib₩sitepackages₩konlpy₩data₩corpus
```

Kobill 폴더

- 대한민국 국회 의안 말뭉치
- 파일명은 의안 번호를 의미

Kolaw 폴더

- 한국 법률 말뭉치



형태소 분석 클래스

- KoNLpy의 Tag서브 패키지에서 형태소 분석을 위한 5개 클래스를 제공한다.

- Hannanum:
  http://semanticweb.kaist.ac.kr/home/index.php/HanNanum
- Kkma:
  http://kkma.snu.ac.kr/
- Komoran:
  https://github.com/shin285/KOMORAN
- Mecab: (윈도우 지원하지 않음)
  https://bitbucket.org/eunjeon/mecab-ko
- twitter:
  https://github.com/twitter/twitter-korean-text

형태소 분석기 성능 비교

로딩 시간: 사전 로딩을 포함하여 클래스를 로딩하는 시간.

Kkma: 5.6988 secs

Komoran: 5.4866 secs

Hannanum: 0.6591 secs

Twitter: 1.4870 secs

Mecab: 0.0007 secs

형태소 분석기 성능 비교

실행 시간: 10만 문자의 문서를 대상으로
각 클래스의 pos 메소드를 실행하는데 소요되는 시간.

Kkma: 35.7163 secs

Komoran: 25.6008 secs

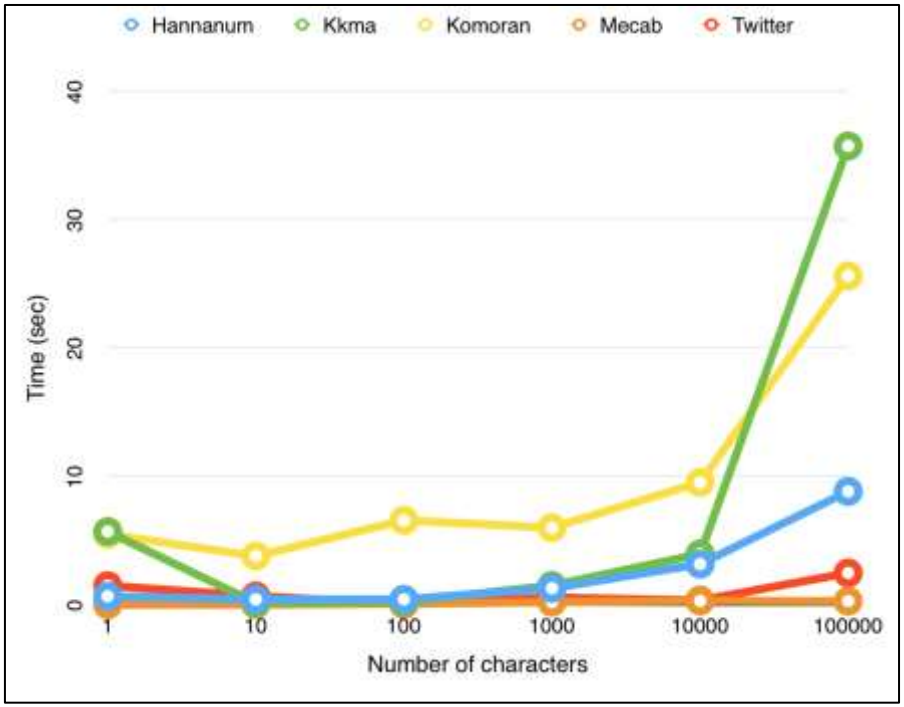
Hannanum: 8.8251 secs

Twitter: 2.4714 secs

Mecab: 0.2838 secs

형태소 분석기 성능 비교

- 문자를 추가할수록 모든 클래스의 실행 시간은 기하급수적으로 증가합니다.



형태소 분석기 성능 비교

"아버지가방에들어가신다"

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

형태소 분석기 성능 비교

"하늘을 나는 자동차"

Hannanum	Kkma	Komoran	Mecab	Twitter
하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

형태소 분석기 성능 비교

- 결론

  빠른 분석이 중요할 때: **Twitter**

  정확한 품사 정보가 필요할 때: **Kkma**

  정확성, 시간 모두 중요할 때: **Komoran**

KoNLPy함수

- morphs(): 의미가 있는 가장 작은 말의 단위로 분리

In [1]:
```
from konlpy.tag import Kkma
# 꼬꼬마 형태소분석기 사용
kkma = Kkma()
```

In [3]:
```
# 의미 있는 단어로 분리
print(kkma.morphs('공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
```

```
['공부', '를', '하', '면', '하', 'ㄹ수록', '모르', '는', '것', '이', '많', '다는',
'것', '을', '알', '게', '되', 'ㅂ니다', '.']
```

KoNLPy함수

- nouns(): 명사만 추출

```
In [4]: # 명사만 추출
        print(kkma.nouns('공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
```

```
['공부']
```

- pos(): 품사 태깅

```
In [4]: # 품사 태깅
        print(kkma.pos('공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
```

```
[('공부', 'NNG'), ('를', 'JKO'), ('하', 'VV'), ('면', 'ECE'), ('하', 'VV'), ('ㄹ수
록', 'ECD'), ('모르', 'VV'), ('는', 'ETD'), ('것', 'NNB'), ('이', 'JKS'), ('많', 'V
A'), ('다는', 'ETD'), ('것', 'NNB'), ('을', 'JKO'), ('알', 'VV'), ('게', 'ECD'),
('되', 'VV'), ('ㅂ니다', 'EFN'), ('.', 'SF')]
```

과제

1. **전처리**
   형태소 분석기 구동을 위한 전처리
   한국어말뭉치 혹은 직접 모은 txt파일의 불필요한 행 (‘ ’, ‘₩n’ 등) 제거

2. **morphs()**
   한국어말뭉치 혹은 직접 모은 txt파일을 형태소 분리

3. **pos()**
   한국어말뭉치 혹은 직접 모은 txt파일에 품사를 표시

4. **Extract noun**
   명사만 추출하여 txt파일에 저장하기

과제

5. 출현 횟수가 높은 5개 단어와 그 횟수 출력하기(선택)

```
[(('의', 'J'), 396), (('.', 'S'), 340), (('하', 'X'), 291),
(('에', 'J'), 283), (('ㄴ다', 'E'), 241)]
```

6. 출현 횟수가 높은 30개 단어를 워드 클라우드로 그리고 출현 횟수가 큰 단어일 수록 크게 그리기. (선택)

3일차 부록 - 과제 설명

- 출현 횟수가 높은 단어 출력하기(선택)
  - most_common 메서드를 사용하여 출현 횟수가 가장 높은 단어를 찾을 수 있다.

```
# (선택) 출현 횟수가 높은 단어 출력하기
from collections import Counter
from konlpy.corpus import kolaw
from konlpy.tag import Hannanum
doc = kolaw.open('constitution.txt').read()
pos = Hannanum().pos(doc)
cnt = Counter(pos)
print(cnt.most_common(5))
```

```
[(('의', 'J'), 396), (('.', 'S'), 340), (('하', 'X'), 291),
(('에', 'J'), 283), (('ㄴ다', 'E'), 241)]
```

3일차 부록 - 환경 설치

- **환경 설치(1/2)**

  PyGame

  - 게임 설계를 위한 python 모델이다.

  ✓ 설치 방법

```
(venv) C:₩NLP>pip install pygame
```

  ✓ 설치 확인

```
(venv) C:₩NLP>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygame
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
>>>
```

3일차 부록 - 환경 설치

- **환경 설치(2/2)**

  PyTagCloud

  - 워드 클라우드(word cloud)를 생성 하는 오픈 소스이다.
  - 기존 워드 클라우드와 달리 한국어도 지원한다.

  ✓ 설치 방법: (다운로드 시간 몇 분 소요)
  pip install git+https://github.com/e9t/PyTagCloud.git

```
(venv) C:₩NLP>pip install git+https://github.com/e9t/PyTagCloud.git
```

  ✓ 설치 확인

```
(venv) C:₩NLP>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pytagcloud
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
>>>
```

3일차 부록 - 과제 설명

- 단어를 워드클라우드로 그리기(선택)(1/3)
  - 워드클라우드 그릴 때 txt에서 출현 횟수가 높은 단어일 수록 크게 설정한다.

3일차 부록 - 과제 설명

- 단어를 워드클라우드로 그리기(선택)(2/3)

```
# (선택) 단어를 워드클라우드로 그리기
from collections import Counter
from konlpy.corpus import kolaw
from konlpy.tag import Hannanum
import webbrowser
import random
import pytagcloud # requires Korean font support

# 색갈을 랜덤으로 생성
r = lambda: random.randint(0,255)
color = lambda: (r(), r(), r())

# 형태소분석을 통해 명사만 추출
def get_tags(text, ntags=50, multiplier=3):
    h = Hannanum()
    nouns = h.nouns(text)
    count = Counter(nouns)
    # 워드클라우드에 출현횟수가 많은 단어일수록 사이즈를 크게 설정한다.
    return [{ 'color': color(), 'tag': n, 'size': c*multiplier }\
                for n, c in count.most_common(ntags)]
```

3일차 부록 - 과제 설명

- 단어를 워드클라우드로 그리기(선택)(3/3)

```
# 워드클라우드 생성
def draw_cloud(tags, filename, fontname='Noto Sans CJK', size=(1280, 720)):
    pytagcloud.create_tag_image(tags, filename, fontname=fontname, size=size)
    webbrowser.open(filename)

doc = kolaw.open('constitution.txt').read()
tags = get_tags(doc)
print(tags)
draw_cloud(tags, 'wordcloud.png')
```

```
[{'tag': '법률', 'size': 345, 'color': (35, 40, 141)}, {'tag': '수', 'size': 27
3, 'color': (142, 101, 57)}, {'tag': '①', 'size': 234, 'color': (195, 137, 10
3)}, {'tag': '때', 'size': 165, 'color': (139, 13, 252)}, {'tag': '국회', 'siz
e': 159, 'color': (127, 246, 140)}, {'tag': '국민', 'size': 153, 'color': (28,
53, 155)}, {'tag': '대통령', 'size': 150, 'color': (115, 108, 238)}, {'tag':
'바', 'size': 111, 'color': (24, 14, 177)}, {'tag': '국가', 'size': 108, 'colo
```

4일차

Word2Vec

목차

- 단어들을 백터화 하기
- One-hot Encoding
- Word Embedding
- Word2Vec이란?
- 알고리즘
- Skip-gram(SG) 와 CBOW
- 환경 설치
- Gensim Word2Vec
- 트레이닝
- 모델 저장하고 불러오기
- 유사도(관련성) 계산1
- 유사도(관련성) 계산2
- 과제

단어들을 백터화 하기

- “어떻게 단어들을 Vector로 표현할 것인가?”

- “**One-hot Encoding**”방식:
  - N개 단어가 있는 “사전(Dictionary)”이 있다. 단어를 표현하기 위해 길이가 N인 Vector을 만들고 그 단어가 해당되는 자리에 1을 넣고 나머지 자리에 0을 넣는다.
  - 사전이 [감자, 딸기, 사과, 수박]이면, 사과를 표현하는Vector는 [0,0,1,0]이다.

One-hot Encoding

- 모든 단어를 N차원의 공간에서 vector로 표시할 수 있다.
- One-hot Encoding으로는 단어 간의 유사성을 찾을 수 없다.
- 단어의 vector 공간 크기를 줄일 필요가 있다.
- 각 단어의 관계를 공간에서 표현하는 방법이 필요하다.

Word Embedding

- 특정 단어가 문장 주변에서 나온 각 단어들을 dense vector로 표현하여, 이웃 단어들을 통해 그 의미를 예측한다.
- Word vector 관점에서 중심 단어와 그 주변 단어(context words) 사이에서, 중심 단어의 의미를 예측한다.
- 단어를 vector로 표현하는데 있어 손실(loss)을 최소화 한다.

Word2Vec

- 2013년 구글 연구원에서 Tomas Mikolov 등 여러 연구자들이 모여서 만든 Continous Word Embedding 학습 모형이다.
- 계산량을 줄여서 기존의 방법에 비해, 매우 빠른 학습을 가능하게 하였다.
- 현재 가장 많이 사용하는 Word Embedding 모델이다.

Word2Vec

- 모든 단어를 Vector로 표현하여 단어 사이의 유사성과 차이점을 계산한다.
- 계산된 결과를 바탕으로 그 주변 단어(context words)의 관계를 통해 단어가 표현하는 바를 예측한다.

알고리즘

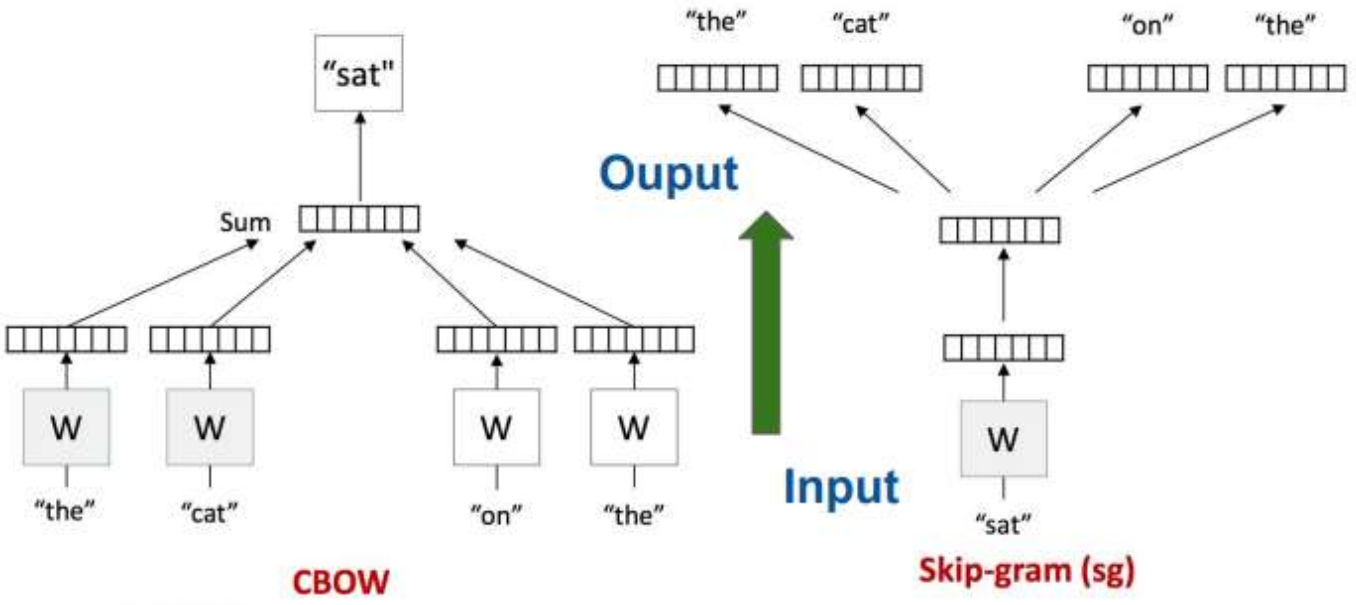
- **Skip-gram(SG)**
  - 타겟 단어를 이용해 주변 단어(context words)를 예측한다.
- **Continous Bag of Words(CBOW)**
  - 주변 단어(context words)를 이용해 타켓 단어를 예측한다.

Skip-gram(SG) 와 CBOW

- Skip-gram(SG)은 주어진 단어 하나로 주위에 등장하는 단어를 유추한다. (CBOW와 반대 방향의 모델)
- 현재 주어진 중심 단어(target word/center word)부터 가까이에 있는 단어의 값을 최대화하는 방향으로 함수를 구성한다.
- 가까이 위치한 단어일수록 현재 단어와 관련이 더 많고 멀리 떨어져 있는 단어일수록 관련성이 낮다고 판단한다.

Skip-gram(SG) 와 CBOW

- **CBOW** input-전후 문맥 output-단어 예측
- **Skip-gram** input-단어 output-전후 문맥 예측



환경 설치

- Scipy 설치(2가지 방법)
  - cmd에서 pip install scipy 명령을 통해 설치

```
(venv) C:₩NLP>pip install scipy
```

  - 아래 사이트에서 scipy 파일을 다운 받고, 압축을 풀고, C:₩venv₩Lib₩site-packages에 복사한다.
    https://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy

환경 설치

- Gensim 설치(2가지 방법)
  - cmd에서 pip install -U gensim 명령을 통해 설치

```
(venv) C:₩NLP>pip install -U gensim
```

  - 아래 사이트에서 gensim 파일을 다운 받고, 압축을 풀고 C:₩venv₩Lib₩site-packages에 복사한다.
    https://www.lfd.uci.edu/~gohlke/pythonlibs/#gensim

Gensim Word2Vec

- 주요한 기능을 메소드 함수로 제공
- 데이터와 모델을 저장하고 호출하는 방식으로 제공
- https://radimrehurek.com/gensim/models/word2vec.html

트레이닝

- size=32: 32차원 vector 사용
- window=5: 주변 단어는 앞 뒤 5개
- min_count=10: 출현 빈도가 10번 미만인 단어 제외
- sg=1: Skip-gram(SG) 와 CBOW 중 Skip-gram(SG)을 사용
- workers=4: 트레이닝 프로세스 4개

In [3]:
```
from gensim.models import word2vec
sents = word2vec.Text8Corpus("Full_nouns2.txt")
model = word2vec.Word2Vec(sents,size=32,window=5,min_count=10,
                          sg=1,hs=1,iter=100,workers=4)
```

모델 저장하고 불러오기

- 저장

```
In [10]: model.save(u"fullnoun2.model")
         print ("Saved Model")

         Saved Model
```

- 불러오기

```
In [14]: model =word2vec.Word2Vec.load('fullnoun2.model')
         print (model)
         print ("Model Loaded!")

         Word2Vec(vocab=484, size=32, alpha=0.025)
         Model Loaded!
```

유사도(관련성) 계산1

- 두개 단어의 유사도(관련성) 계산하기

In [15]:
```
# 【두개 키워드의 유사도 계산하기】
try:
    y1 = model.wv.similarity("Emma", "Taylor")
except KeyError:
    y1 = 0
print(y1)
```

```
0.31270146
```

유사도(관련성) 계산2

- 지정 단어와 유사도(관련성)가 가장 높은 단어5개 출력

In [18]:
```
#  【지정 키워드와 유사도가 제일 높은 5개 단어】
y2 = model.most_similar("Emma", topn=5)
for item in y2:
    print(item)
```

```
(u'Mr.', 0.7077715396881104)
(u'Woodhouse', 0.7009375095367432)
(u'thing', 0.7000381946563721)
(u'Hartfield', 0.6620622873306274)
(u'Harriet', 0.6565461158752441)
```

유사도(관련성) 계산3

- 지정 단어 2개와 유사도(관련성)가 가장 높은 단어 5개 출력

In [19]:
```
#  【두개 키워드와 유사도가 제일 높은 5개 단어】
y3 = model.most_similar(['Emma', 'Taylor'], topn=5)
for item in y3:
    print(item)
```

```
(u'Woodhouse', 0.676986813545227)
(u'father', 0.6273468732833862)
(u'Weston', 0.595104992389679)
(u'friend', 0.571246862411499)
(u'body', 0.547289490699768 1)
```

과제

1. 영어 txt 파일들로 word2vec모델을 트레이닝하고 저장하기

2. 단어들의 유사도 계산하기

과제 (선택)

1. 한국어 txt 파일들로 word2vec모델을 트레이닝하고 저장하기

2. 단어들의 유사도 계산하기

5일차

CNN

(Convolutional Neural Network)

목차

CNN이란?

- 합성곱 신경망(Convolutional neural network)은 시각적 이미지를 분석하는 데 사용되는 깊고 피드-포워드적인 인공신경망의 한 종류이다.
- 이미지 인식, 이미지 분류 등에 많이 사용 되는 알고리즘이다.
- Tensorflow, Caffe, Keras 등이 지원한다.

CNN 발전 과정(1/3) 1998년: LeNet

- CNN(Convolutional Neural Network)의 개념을 최초로 개발한 사람은 프랑스 출신의 Yann LeGun이다.
- 우편번호와 수표의 필기체를 인식하기 위한 용도로 개발을 시작하였으며, 그 연구 그룹이 최종적으로 발표한 구조가 LeNet5이다.
- 트레이닝 할 때 많은 계산이 필요 하고, 당시 컴퓨터 하드웨어의 계산 능력이 제한 되어 있어서 CNN의 발전은 잠시 멈추었다.

CNN 발전 과정(2/3) 2012년: Alexnet

- ILSVRC(ImageNet Large-Scale Visual Recognition Challenge)의 2012년 대회에서 제프리 힌튼 교수팀의 AlexNet이 2위를 큰 폭으로 따돌리고 1위를 차지 했다.
- Alexnet은 GPU를 사용하여 사람들이 받아 들일 수 있는 시간에서 좋은 트레이닝 결과를 얻을 수 있게 되었다.
- 이 때부터 CNN, GPU에 대해 사람들이 관심을 가지게 되었고, Deep Learning이 발전하는 계기가 되었다.

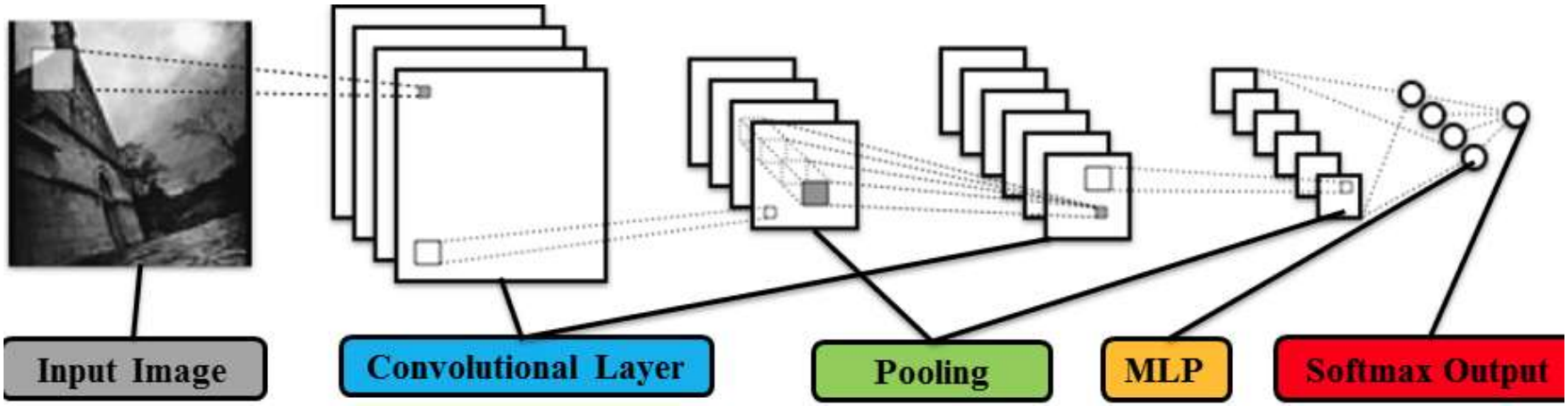
CNN 발전 과정(3/3) 2014년: VGG

- 옥스퍼드 대학교의 Visual Geometry Group이 VGG-16이라는 사전 학습된 모델로 ILSVRC 2014에 참가하여 2위를 차지 했다.
- 당시에 성능이 더 좋은 googleNet이 우승을 하였지만, 구조적인 측면에 VGG-16이 훨씬 간단한 구조로 되어 있어, 이해가 쉽고 테스트 하기에 용이하여, 더 많이 사용 되었다.
- Neural Network는 “Deep Learning”이라는 별칭을 갖게 되었다.

CNN 구조

- CNN 모델은 기본적으로 컨볼루션 계층(convolution layer), 풀링(pooling layer) 계층으로 구성 된다.

- 이 계층들을 얼마나 많이, 또 어떠한 방식으로 쌓느냐 에 따라 성능 차이는 물론 풀 수 있는 문제가 달라질 수 있다.



Convolution layer(1/4)

- 입력된 이미지에서 특징을 추출하기 위해 필터(마스크, 윈도, 커널)를 도입하는 기법이다.

  ① 입력된 이미지에 ② 2x2 필터를 맨 왼쪽 위칸 부터 적용한다.
  ③ 이미지 부분의 값에 가중치 값을 곱해 합하면 0가 된다.

0	1	0	1
1	0	0	1
1	1	0	0
1	1	0	1

① 입력된 이미지

x1	x0
x0	x1

② 2x2 필터

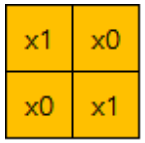
샘플 가중치 (x1, x0)

0x1	1x0	0	1
1x0	0x1	0	1
1	1	0	0
1	1	0	1

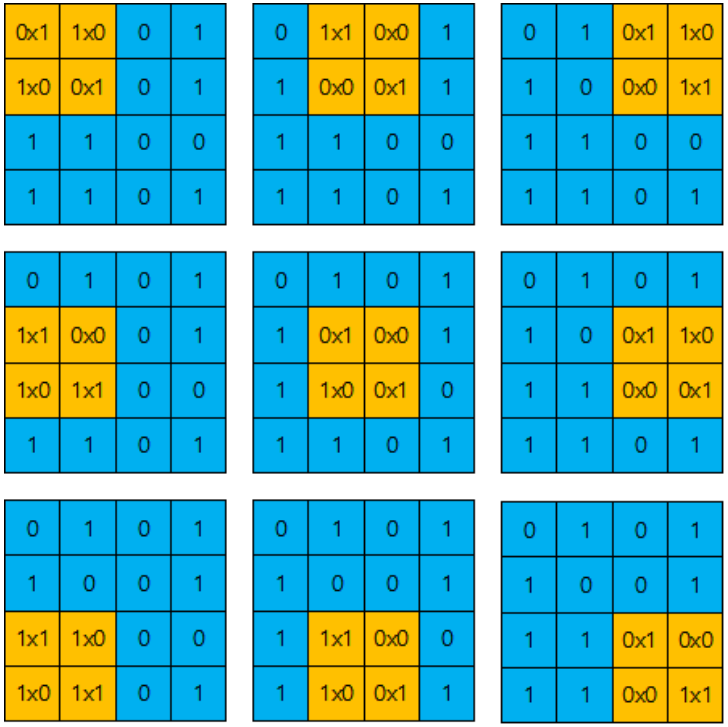
③ (0x1)+(1x0)+(1x0)+(0x1)=0

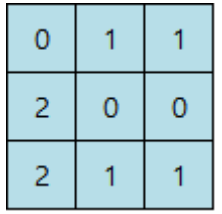
Convolution layer(2/4)

- ① 2x2 필터를 한 칸씩 옮겨 적용 하면,
  ② 컨볼루션(합성곱)이 나온다.
- output size = ((W-F+2P)/S) + 1=(4-2)/1+1=3



① 2x2필터

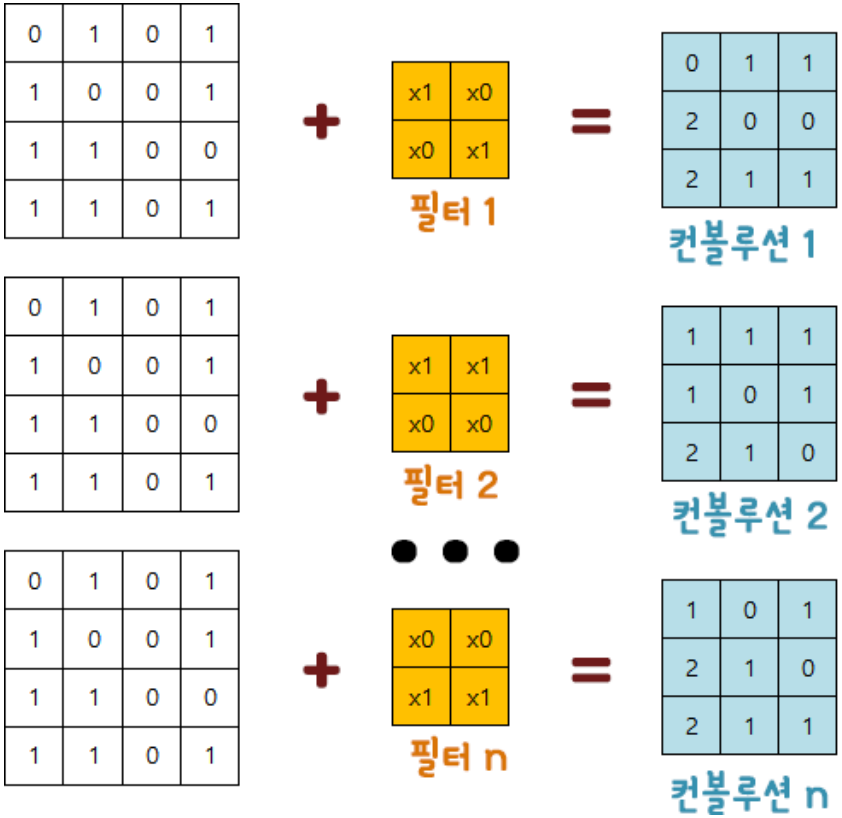




② 컨볼루션(합성곱)

Convolution layer(3/4)

- 필터를 여러 개 만들 경우 여러 개의 컨볼루션이 만들어 진다.



Convolution layer(4/4)

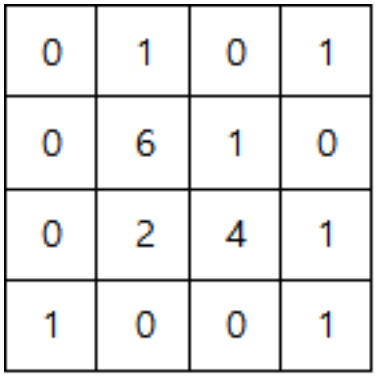
- CNN 에서는 input image가 Conv Layer를 몇 번 거치는지도 코딩으로 다 설정하기 때문에 각 Layer를 거칠 때 마다 size가 어떻게 변하는지를 아는 것도 매우 중요하다.

- 수식적으로 공식화까지 되어 있다.
- Input size가 (W), 필터 size (F), stride (S), padding의 수(P), 일 때 output size = ((W-F+2P)/S) + 1

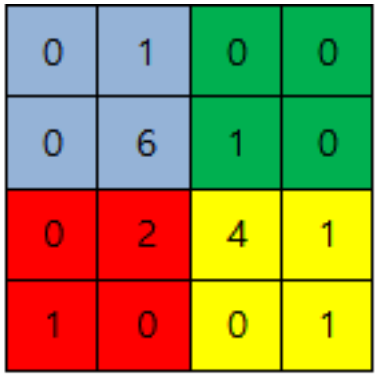
- 예를 들어 5x5x3의 input인데 패딩이 1추가되고, 3x3x3의 필터를 사용하고 stride는 2일 때, ((5-3+2X1)/2 + 1) = 3 이다.

Pooling layer

- 풀링 층은 다양한 기법 중에 맥스 풀링을 사용해서 컨볼루션 층으로 넘어온 값을 단순화 한다.
- 맥스 풀링은 ①컨볼루션에서 가장 큰 값만 남기고 나머지를 버리면, ③의 결과 값이 나온다.
- 전체 값 중 ¼만 남기고 downsampling 하는 과정이다.



① 컨볼루션 값



② 맥스 풀링

③ 결과 값

Fully-Connected Layer

- 완전 연결 계층은 Convolution layer, Pooling layer를 통해 최종적으로 도달하는 층이다.

- 필터를 따로 두는게 아닌, 전체 성분을 모두 가중합 형식으로 연결하여 output을 얻는 과정이다.

과제

1. 나중에 챗봇에 사용할 이미지 2~3가지 종류를 수집하기

2. 수집한 이미지를 식별하기 쉬운 이름으로 바꾸고 같은 크기로 resize하기
   - 예를 들면 짜장면 0_*.jpg, 짬뽕 1_*.jpg, 탕수육 2_*.jpg

6일차

CNN

- Image Classification

목차

실습

실습 환경 구축

데이터 다운로드

트레이닝

테스트

실습 환경 구축 케라스(Keras)란?

- 케라스(Keras)는 파이썬으로 작성된 오픈 소스 신경망 라이브러리다
- MXNet, Deeplearning4j, TensorFlow, Microsoft Cognitive Toolkit, Theano 위에서 수행할 수 있다.
- 딥 신경망과의 빠른 실험을 가능케 하도록 설계되었으며, 최소한의 모듈 방식의 확장 가능성에 초점을 둔다.

실습 환경 구축 케라스(Keras) 설치하기

- 설치방법 : pip install keras

```
(venv) C:₩CNN>pip install keras
```

- 설치확인

```
(venv) C:₩CNN>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>>
```

실습 환경 구축 PIL 소개 및 설치

- PIL은 파이썬으로 작성된 Imaging Library이고 현재 Pillow에 포함되고 있다.
  - 설치방법 : pip install Pillow

```
(venv) C:₩CNN>pip install Pillow
```

  - 설치확인

```
(venv) C:₩CNN>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from PIL import Image
>>>
```

데이터 다운로드(1/3)

- CIFAR-10데이터를 사용한다.
- CIFAR-10에는 10개 class에 대한 이미지(32x32)들이 있고 각 class 당6000장씩 들어있다. 그중 5000장은 트레이닝, 1000장은 테스트에 사용 된다.
- Airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- From keras.datasets import cifar10으로 쉽게 접근 할 수 있다.

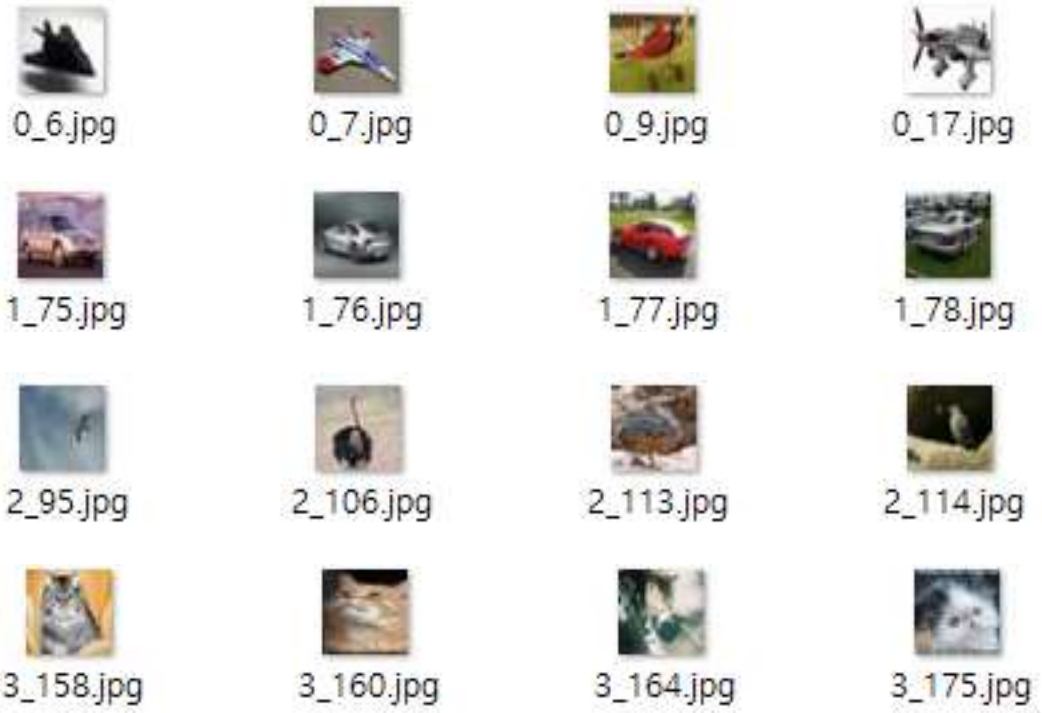
데이터 다운로드(2/3)

- Cifar10으로 비행기, 자동차, 새, 고양이의 이미지를 각각 60장(train 50장, test 10장)씩 내려 받고, data와 datat 폴더에 저장한다.

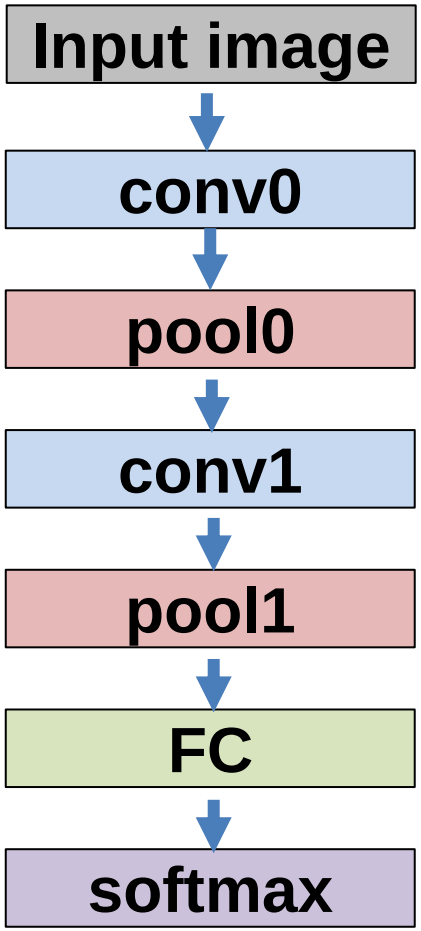
```
from keras.datasets import cifar10
from PIL import Image
import numpy as np
import os

(x_train, y_train), (x_test, y_test) = cifar10.load_data()

max_num_datas = 50 # 트레이닝 이미지 개수
test_num_datas=10 # 테스트 이미지 개수
num_classes = 3
num_datas_list = np.zeros(num_classes)

img_dir = "data" # 트레이닝 이미지 저장 폴더
img_dirt = "datat" # 테스트 이미지 저장 폴더
```

데이터 다운로드(3/3)

- 다운로드하면 비행기는 0_xx.jpg, 자동차는 1_xx.jpg, 새는 2_xx.jpg 형태로 저장한다.



트레이닝(1/7)



트레이닝(2/7)

- 첫번째 Convolutional Layer0

```
# input은 이미지=32x32x3, filters 필터개수 =20, kernal_size 필터사이즈=5x5, stride 이동=1x1
conv0 = tf.layers.conv2d(datas_placeholder, 20, 5, activation=tf.nn.relu)
print(conv0)
# output size= [(W - F + 2P)/S] +1 = (32-5)/1+1 =28
# shape=(?, 28, 28, 20)
```

- 첫번째 pooling Layer0 (사이즈를 ¼로 줄인다)

```
# input=28x28x20, pooling size=2x2, stride이동=2x2
pool0 = tf.layers.max_pooling2d(conv0, [2, 2], [2, 2])
print(pool0)
# output size= W = [(W - F )/S] +1 = [(28-2)/2]+1 =14
# shape=(?, 14, 14, 20)
```

트레이닝(3/7)

- 두번째 Convolutional Layer1

```
# input=14x14x20, filters 필터개수=40, kernal_size 필터사이즈=4x4, stride 이동=1x1
conv1 = tf.layers.conv2d(pool0, 40, 4, activation=tf.nn.relu)
print(conv1)
# output size= [(W - F + 2P)/S] +1 = (14-4)/1+1 =11
# shape=(?, 11, 11, 40)
```

- 두번째 pooling Layer1 (사이즈를 ¼로 줄인다)

```
# input=11x11x40, pooling size=2x2, stride 이동=2x2
pool1 = tf.layers.max_pooling2d(conv1, [2, 2], [2, 2])
print(pool1)
# output size= W = [(W - F )/S] +1 = [(11-2)/2]+1 =5.5
# shape=(?, 5, 5, 40)
```

트레이닝(4/7)

- FC

```
# fully-connected layer
fc = tf.layers.dense(flatten, 400, activation=tf.nn.relu)
print(fc)
# shape=(?, 400)

# DropOut를 추가하여 overfitting을 방지한다.
dropout_fc = tf.layers.dropout(fc, dropout_placeholdr)
print(dropout_fc)
# shape=(?, 400)
```

트레이닝(5/7)

- softmax

```
# DropOut를 추가하여 overfitting을 방지한다.
dropout_fc = tf.layers.dropout(fc, dropout_placeholdr)
print(dropout_fc)
# shape=(?, 400)

# output=4개 class
logits = tf.layers.dense(dropout_fc, num_classes)
print(logits)
# shape=(?, 4)
```

트레이닝(6/7)

- Loss

```
# loss를 정의한다.
losses = tf.nn.softmax_cross_entropy_with_logits(
    labels=tf.one_hot(labels_placeholder, num_classes), #실제 label
    logits=logits #트레이닝 output label
)
```

- Optimizer

```
# optimizer를 정의한다.
optimizer = tf.train.AdamOptimizer(learning_rate=1e-2).minimize(losses)
```

트레이닝(7/7)

- 실행: train=True

```
(venv) C:₩CNN>python cnn.py
```

```
training
step = 0        mean loss = 1.3929905891418457
step = 10       mean loss = 1.3828598260879517
step = 20       mean loss = 1.3281848430633545
step = 30       mean loss = 1.0992988348007202
step = 40       mean loss = 1.0461777448654175
step = 50       mean loss = 0.757497251033783
step = 60       mean loss = 0.43973755836486816
step = 70       mean loss = 0.17067275941371918
step = 80       mean loss = 0.042258091270923615
step = 90       mean loss = 0.008834458887577057
step = 100      mean loss = 0.0023835503961890936
step = 110      mean loss = 0.0010368013754487038
step = 120      mean loss = 0.0006441258010454476
step = 130      mean loss = 0.0004793642438016832
step = 140      mean loss = 0.0003921742318198085
training done! saved in model/image_model
```

테스트

- 실행: train=False

```
(venv) C:₩CNN>python cnn.py
```

```
datat₩0_198.jpg airplane => airplane
datat₩0_201.jpg airplane => cat
datat₩0_203.jpg airplane => bird
datat₩0_208.jpg airplane => car
datat₩0_218.jpg airplane => cat
datat₩0_219.jpg airplane => airplane
datat₩0_223.jpg airplane => airplane
datat₩0_227.jpg airplane => airplane
datat₩0_228.jpg airplane => airplane
datat₩0_230.jpg airplane => airplane
datat₩1_136.jpg car => car
datat₩1_148.jpg car => car
datat₩1_151.jpg car => car
datat₩1_156.jpg car => car
datat₩1_161.jpg car => airplane
datat₩1_163.jpg car => airplane
datat₩1_165.jpg car => car
datat₩1_180.jpg car => car
datat₩1_181.jpg car => car
datat₩1_188.jpg car => car
```

```
datat₩2_212.jpg bird => bird
datat₩2_213.jpg bird => cat
datat₩2_214.jpg bird => bird
datat₩2_220.jpg bird => airplane
datat₩2_221.jpg bird => airplane
datat₩2_222.jpg bird => bird
datat₩2_225.jpg bird => car
datat₩2_233.jpg bird => bird
datat₩2_236.jpg bird => airplane
datat₩2_237.jpg bird => cat
datat₩3_217.jpg cat => cat
datat₩3_224.jpg cat => cat
datat₩3_226.jpg cat => bird
datat₩3_229.jpg cat => bird
datat₩3_231.jpg cat => cat
datat₩3_232.jpg cat => car
datat₩3_234.jpg cat => cat
datat₩3_235.jpg cat => airplane
datat₩3_238.jpg cat => airplane
datat₩3_239.jpg cat => car
```

과제

1. CIFAR-10데이터를 사용하여 10개의 class를 분류하는 model을 트레이닝하기

2. 5일차에 모았던 이미지를 가지고 분류해 보기

6일차 부록-과제 설명

- CIFAR-10데이터로 10개의 class를 분류(1/2)

```
conv0 = tf.layers.conv2d(datas_placeholder, 64, 5, activation=tf.nn.relu, padding='SAME')
pool0 = tf.layers.max_pooling2d(conv0, [3, 3], [2, 2], padding='SAME')
print(conv0) #shape=(?, 32, 32, 64)
print(pool0) #(?, 16, 16, 64)

conv1 = tf.layers.conv2d(pool0, 64, 5, activation=tf.nn.relu, padding='SAME')
pool1 = tf.layers.max_pooling2d(conv1, [3, 3], [2, 2], padding='SAME')
print(conv1) #shape=(?, 16, 16, 64)
print(pool1) #shape=(?, 8, 8, 64)

conv2 = tf.layers.conv2d(pool1, 128, 3, activation=tf.nn.relu)
print(conv2) #shape=(?, 6, 6, 128)

conv3 = tf.layers.conv2d(conv2, 128, 3, activation=tf.nn.relu)
print(conv3) #shape=(?, 4, 4, 128)

# 3-D vector를 1-D vector로 변환한다.
flatten = tf.layers.flatten(conv3)
# fully-connected layer
print(flatten)

fc = tf.layers.dense(flatten, 512, activation=tf.nn.relu)
```

6일차 부록-과제 설명

- CIFAR-10데이터로 10개의 class를 분류(2/2)

```
datat₩0_1008.jpg airplane => cat
datat₩0_1019.jpg airplane => ship
datat₩0_974.jpg airplane => airplane
datat₩0_980.jpg airplane => ship
datat₩0_985.jpg airplane => airplane
datat₩1_875.jpg automobile => automobile
datat₩1_880.jpg automobile => automobile
datat₩1_894.jpg automobile => truck
datat₩1_918.jpg automobile => truck
datat₩1_938.jpg automobile => automobile
datat₩2_1003.jpg bird => dog
datat₩2_1004.jpg bird => bird
datat₩2_1013.jpg bird => bird
datat₩2_1020.jpg bird => dog
datat₩2_999.jpg bird => bird
datat₩3_1027.jpg cat => cat
datat₩3_1028.jpg cat => cat
datat₩3_1031.jpg cat => deer
datat₩3_1033.jpg cat => bird
datat₩3_1034.jpg cat => cat
datat₩4_1000.jpg deer => cat
datat₩4_1001.jpg deer => frog
datat₩4_1002.jpg deer => bird
datat₩4_1011.jpg deer => deer
datat₩4_996.jpg deer => deer
```

```
datat₩5_1045.jpg dog => dog
datat₩5_1046.jpg dog => truck
datat₩5_1047.jpg dog => frog
datat₩5_1048.jpg dog => dog
datat₩5_1049.jpg dog => cat
datat₩6_925.jpg frog => frog
datat₩6_931.jpg frog => frog
datat₩6_933.jpg frog => frog
datat₩6_937.jpg frog => truck
datat₩6_939.jpg frog => frog
datat₩7_1012.jpg horse => horse
datat₩7_1022.jpg horse => horse
datat₩7_971.jpg horse => horse
datat₩7_972.jpg horse => horse
datat₩7_989.jpg horse => horse
datat₩8_1007.jpg ship => ship
datat₩8_1009.jpg ship => airplane
datat₩8_1016.jpg ship => ship
datat₩8_1018.jpg ship => truck
datat₩8_1025.jpg ship => ship
datat₩9_951.jpg truck => truck
datat₩9_954.jpg truck => truck
datat₩9_962.jpg truck => automobile
datat₩9_968.jpg truck => truck
datat₩9_994.jpg truck => truck
```

6일차 부록-과제 설명

- 짜장면, 짬뽕, 탕수육 분류하기 (1/2)

```
conv0 = tf.layers.conv2d(datas_placeholder, 20, 5, activation=tf.nn.relu)
pool0 = tf.layers.max_pooling2d(conv0, [2, 2], [2, 2])
print(conv0) #shape=(?, 28, 28, 20)
print(pool0) #shape=(?, 14, 14, 20)

conv1 = tf.layers.conv2d(pool0, 40, 3, activation=tf.nn.relu)
pool1 = tf.layers.max_pooling2d(conv1, [2, 2], [2, 2])
print(conv1) #shape=(?, 12, 12, 40)
print(pool1) #shape=(?, 6, 6, 40),

conv2 = tf.layers.conv2d(pool1, 40, 3, activation=tf.nn.relu)
pool2 = tf.layers.max_pooling2d(conv2, [2, 2], [2, 2])
print(conv2) #shape=(?, 4, 4, 40),
print(pool2) #shape=(?, 2, 2, 40),

# 3-D vector을 1-D vector로 변환한다.
flatten = tf.layers.flatten(pool2)
print(flatten) #shape=(?, 1440)

# fully-connected layer
fc = tf.layers.dense(flatten, 400, activation=tf.nn.relu)
print(fc) #shape=(?, 400)
```

6일차 부록-과제 설명

- 짜장면, 짬뽕, 탕수육 분류하기 (2/2)


```
test_s₩0_1.jpg  Zzazang => Zzazang
test_s₩0_2.jpg  Zzazang => Zzazang
test_s₩0_3.jpg  Zzazang => Zzazang
test_s₩1_4.jpg  Zzambbong => Zzambbong
test_s₩1_5.jpg  Zzambbong => Zzambbong
test_s₩1_6.jpg  Zzambbong => Zzambbong
test_s₩2_7.jpg  Tangsu6 => Tangsu6
test_s₩2_8.jpg  Tangsu6 => Tangsu6
test_s₩2_9.jpg  Tangsu6 => Tangsu6
```

7일차

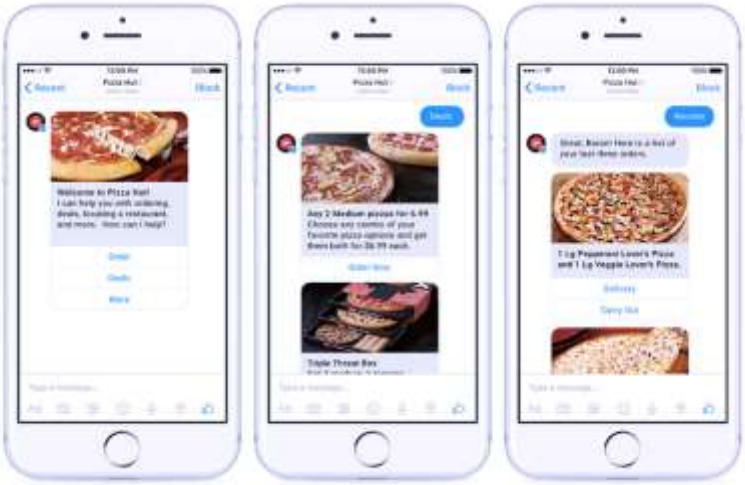
챗봇과 머신러닝

목차

- **챗봇이란?**
- **챗봇 사례?**
  1) 스타벅스의 챗봇 Barista
  2) 의류브랜드 버버리의 챗봇
- **개발 방식 비교**
- **챗봇 빌더**
- **주요 챗봇 빌더**
  1) Chatfuel(facebook)
  2) Dialogflow(google)
  3) Kakao i(kakao)
- **챗봇의 두 가지 모델**
  1) 검색모델
  2) 생성모델
- **기계학습**
  1) 지도학습
  2) 비지도학습
  3) 강화학습
- **과제**

챗봇이란?

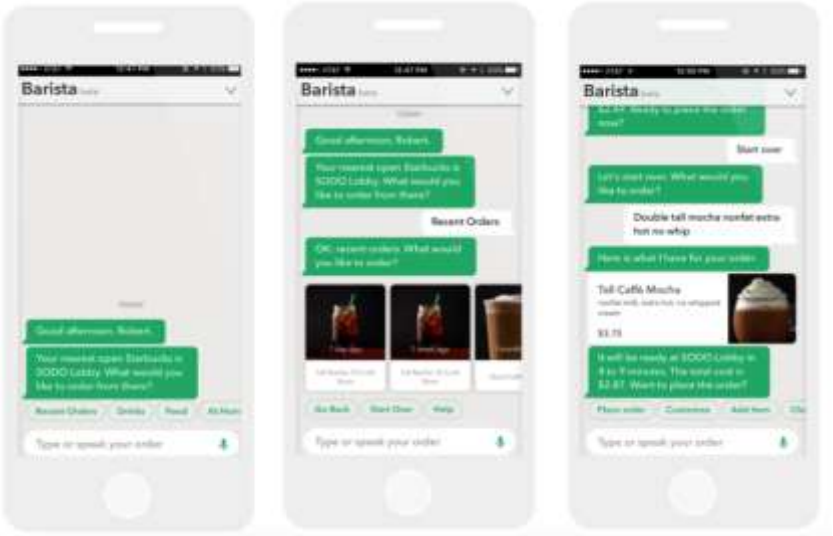
- 챗봇(ChatBot)이란 대화형 인터페이스로 기계가 사용자와 상호작용하는 서비스를 말한다.
- 쉽게 말하자면 채팅 화면에서 사람을 대신해서 대화하는 기계이다.



피자헛 페이스북 챗봇

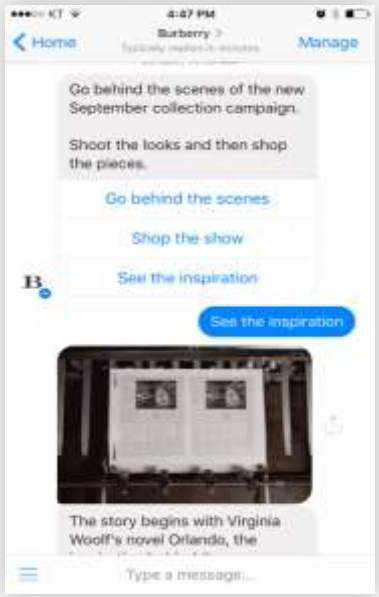
챗봇 사례 스타벅스의 챗봇 Barista

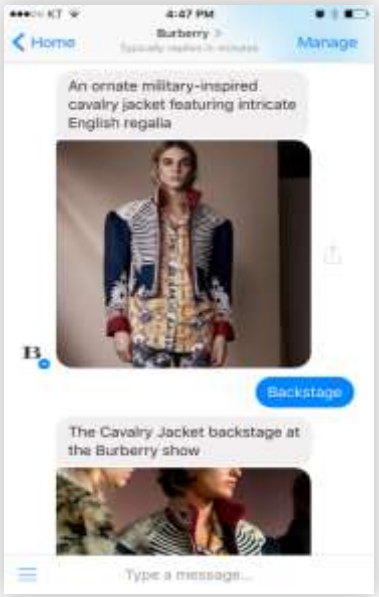
- 커피 주문을 도와주는 챗봇으로 스타벅스 앱 안에서 구동된다.
- 고객의 주문 히스토리와 대화를 통해 빠르고 개인화된 주문을 도와준다.



챗봇 사례 의류브랜드 버버리의 챗봇

- 페이스북 메신저에서 대화하며 다양한 이미지와 제품을 보여주고 실제로 구매도 할 수 있도록 프로모션용으로 만들어진 챗봇이다.





개발 방식 비교

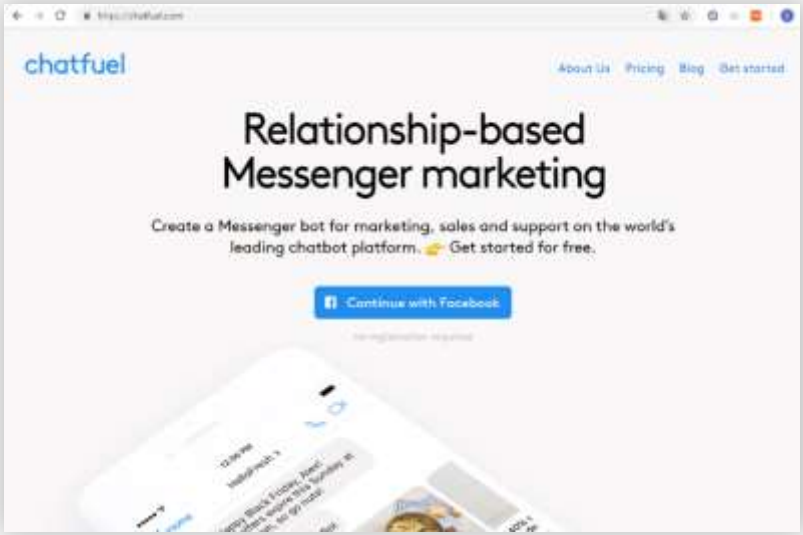
챗봇 빌더 사용		챗봇 엔진 개발
✓ 챗봇 빌더에서 분석 결과를 받아 대답 생성 ✓ 대부분 통계 기반	자연어 처리	✓ 직접 구현 ✓ 규칙 기반/통계 기반/딥러닝 기반
✓ 개발이 상대적으로 쉬움	장점	✓ 자유도 높음 ✓ 무료
✓ 지원하는 기능만 사용 가능 ✓ 유료	단점	✓ 개발에 많은 시간과 노력 필요

챗봇 빌더

- 챗봇 빌더란 기본 템플릿이 존재하고 사용자가 커스터마이즈 할 수 있도록 기능을 제공하는 서비스를 말한다.
- 일반인도 코딩없이 챗봇을 만들어 볼 수있는 서비스도 있고 개발자가 기능을 확장할 수 있도록 API를 지원해주는 서비스도 있다.
- 해외에는 꽤 많은 챗봇 빌더가 나와 있다. (Api.ai, wit.ai Chatfuel등은 코딩 없이도 챗봇을 만들 수 있다)
- 국내에서 개발된 챗봇 빌더는 카카오I, 클로바, 마인드맵이 있다.

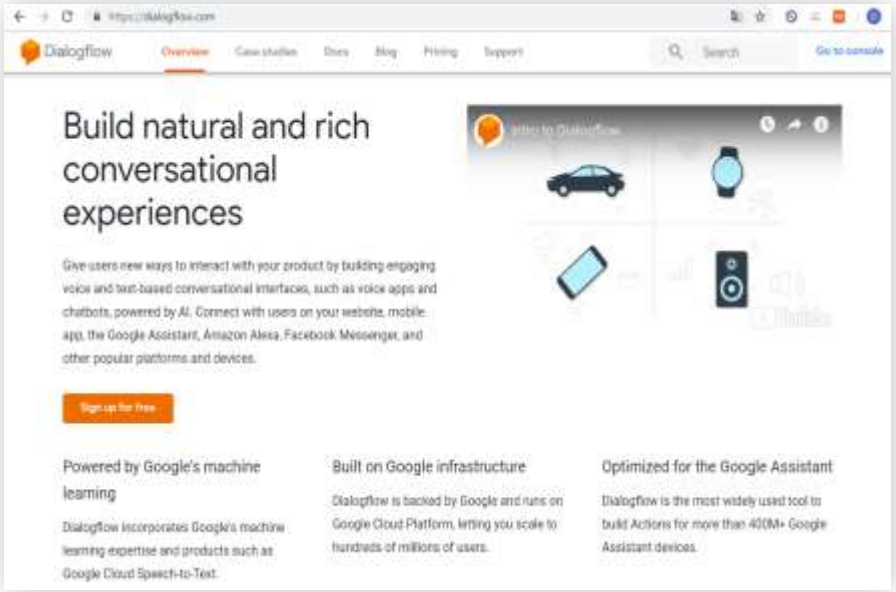
주요 챗봇 빌더 Chatfuel(facebook)

- 페이스북에서 지원하는 챗봇 플랫폼이다.
- 비개발자도 챗봇을 직접 만들어 볼 수 있을 정도로 사용법이 간단하다.
- 클릭 몇 번으로 페이스북과 연결이 가능하다.



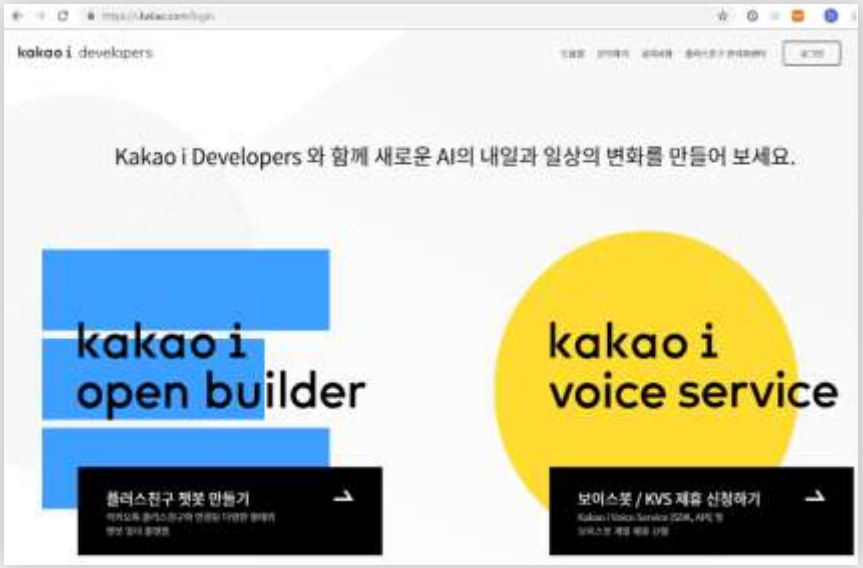
주요 챗봇 빌더 Dialogflow(google)

- 구글이 제공하는 챗봇 플랫폼이다.
- 의도분석(Intent)과 성분분석(Entity)를 지원한다.
- 메신저 연동이 쉽고, 외부 서버와도 연결이 가능하다.



주요 챗봇 빌더 Kakao i(kakao)

- 카카오톡 플러스 친구에 적용이 가능한 텍스트형 챗봇 개발이 가능하다.
- 카카오미니에 적용된 음성 인터페이스와 호환되어 음성형 챗봇도 개발 가능하다.



챗봇의 두 가지 모델

검색 모델(Retrieval Based Model)

생성 모델(Generative Model)

검색 모델(Retrieval Based Model)

- 사용자의 질문 내용과 문맥에 기반해서 의도를 파악하고 적절한 응답을 선택(검색)하는 방식을 말한다.
- 새로운 텍스트를 생성하지 않고 미리 만들어진 답변에서 답을 고르는 방식이다.
- 답변의 내용이 정확하고 문법에 맞지 않는 답변을 하지 않는다.

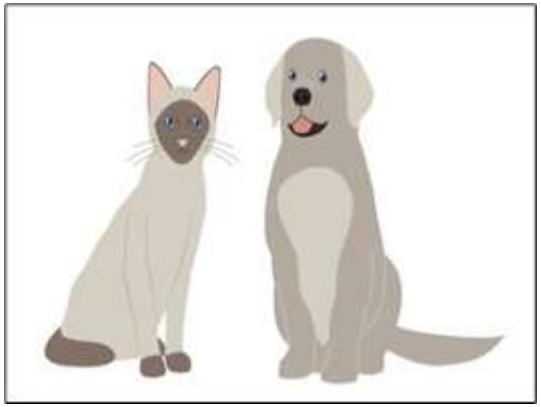
생성 모델(Generative Model)

- 대화 상대의 질문 의도를 파악하기 보다는 사용자의 질문에 대한 최선의 답변을 생성한다.
- 사용자의 질문에 대한 새로운 답변을 만들어 내기 때문에 마치 사람과 이야기하는 듯한 인상을 주기도 한다.
- 가벼운 일상 대화를 제공하는 챗봇에서 주로 사용된다.
- 기계 학습을 활용하여 답변을 생성한다.

기계학습(machine learning)

- 기계학습이란 문제를 해결하기 위한 맞춤 코드를 작성하지 않고도 데이터로부터 **학습**하고 일반화된 **알고리즘**을 이용하여 문제를 해결하는 것이다.

- **알파고**와 **개와 고양이 분류하기**는 가장 널리 알려진 기계학습 사례이다.



기계학습(machine learning)

지도학습(Supervised Learning)

비지도학습(Unsupervised Learning)

강화학습(Reinforcement Learning)

지도학습(Supervised Learning)

- 데이터에 대한 레이블(Label)이 주어진 상태로 기계를 학습시키는 방법이다.
- 분류 문제를 해결하는데 유용하게 사용된다.

사자인가, 코알라인가,
아니면 고양이인가?

분류 알고리즘으로
이 문제를 해결할 수 있다.

비지도학습(Unsupervised Learning)

- 데이터에 대한 레이블 없이 순수 데이터만 가지고 기계를 학습시킨다.
- 딥러닝 모델에 뭘 할지에 대한 명확한 지시 없이 데이터가 주어진다.
- 알고리즘은 유용한 특징을 추출하고 구조를 분석함으로써 자동적으로 데이터 안에서 구조를 찾는다.

강화학습(Reinforcement Learning)

- 행동을 하고 시행착오를 겪으면서 학습하는 것이다.
- 어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동 중에서 보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법이다.
- 별도의 데이터를 제공하지 않고 환경과 행동에 따른 보상만 주어진다.

과제

1. 텐서플로를 이용하여
   MNIST 문자 인식 프로그램을 만들어라.

   - MNIST란 28×28크기의 0~9사이의 숫자 이미지와 이에 해당하는 레이블(Label)로 구성된 데이터셋이다
   - MNIST는 텐서플로에서 기본으로 제공한다.

8일차

챗봇실습 및 종합

목차

순환 신경망(Recurrent Neural Networks, RNN)이란?

- 연속성이 있는 데이터를 다루는 신경망 모델이다.
- 음성인식이나 자연어 처리 분야에서 많이 사용된다.
- 자연어의 경우 단어 하나만으로 처리가 어렵고 앞뒤 문맥을 함께 이해해야 해석이 가능하다(RNN 모델이 사용되는 이유).
- LSTM(Long-Short Term Memory) 알고리즘이 주로 사용된다.

Sequence2Sequence

- 인코더와 디코더 두 개의 RNN을 이용하여
  문장을 학습하는 기법이다.

- 기계번역 연구를 통해 나온 기술이며
  주로 번역 기술에 사용된다.

- 번역이 언어를 또다른 언어로 바꿔주는 것인데
  이것을 변형하여 다른 언어가 아니라 응답으로 변환해주는
  방식으로 챗봇에 응용이 가능하다.

기계학습 기반 챗봇 만들기

- 기계 학습을 활용하여 답변을 생성해내는
  생성 모델 기반의 챗봇이다.

- 구글에서 공개한 NMT(Neural Machine Translation)
  Tutorial을 기반으로 한다.

기계학습 기반 챗봇 만들기

실습 환경 설정

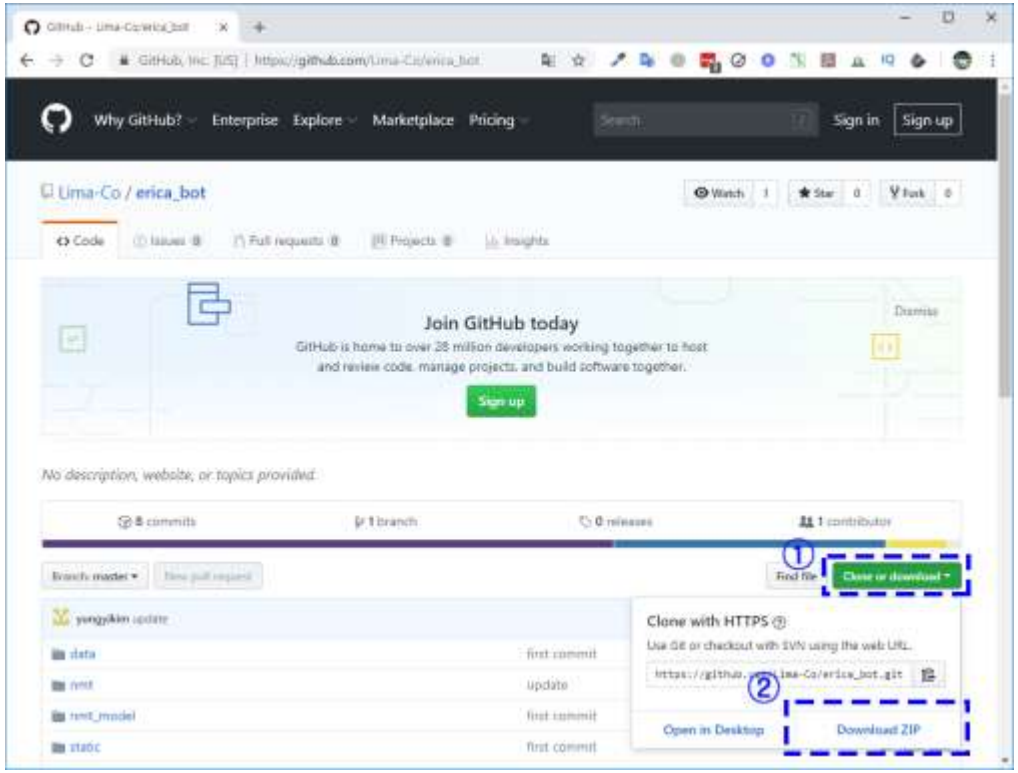
학습 데이터 소개

데이터 학습

챗봇 실행

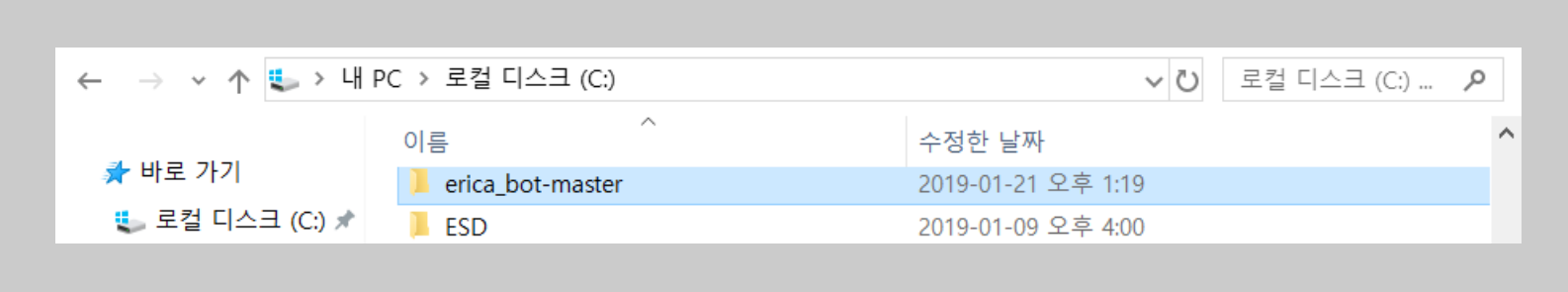
실습 환경 설정(1/4) 챗봇 코드 다운로드

- https://github.com/Lima-Co/erica_bot



실습 환경 설정(2/4) 챗봇 코드 설치

- 압축을 풀어 C:₩로 복사 붙여 넣기를 한다.

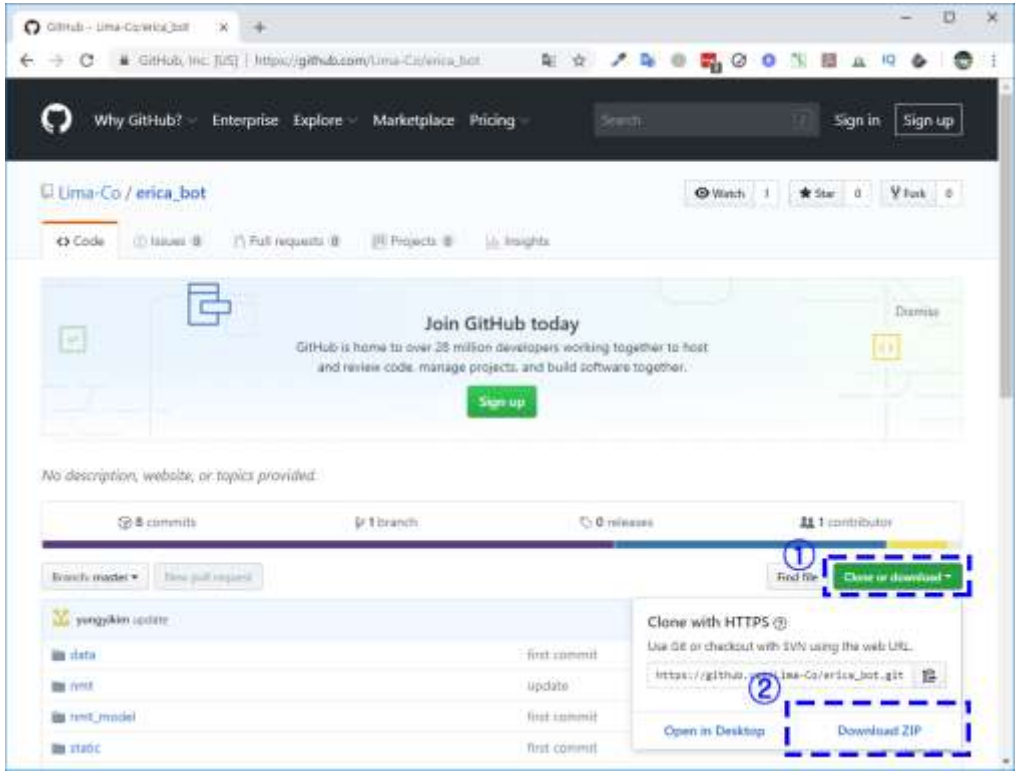


- 가상 환경 활성화 후
- C:₩erica_bot-master 디렉터리로 이동
- 패키지 설치(pip install -r requirements.txt)

```
C:₩>venv₩scripts₩activate
(venv) C:₩>cd erica_bot-master

(venv) C:₩erica_bot-master>pip install -r requirements.txt
```

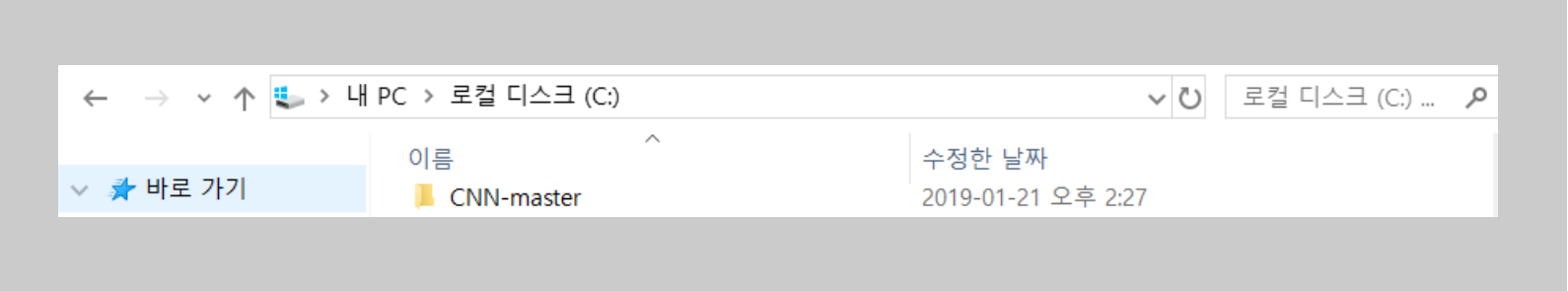
실습 환경 설정(3/4) CNN 코드 다운로드

- https://github.com/Lima-Co/CNN



실습 환경 설정(4/4) CNN 코드 설치

- 압축을 풀어 C:₩로 복사 붙여 넣기를 한다.



- 가상 환경 활성화 후
- C:₩CNN-master 디렉터리로 이동
- 패키지 설치(pip install -r requirements.txt)

```
C:₩>venv₩scripts₩activate
(venv) C:₩>cd CNN-master

(venv) C:₩CNN-master>pip install -r requirements.txt
```

학습 데이터 소개(1/4)

- 학습 데이터 경로 : data/

- **Train set**
  - 학습용 데이터
  - 데이터가 많을수록 챗봇의 질이 좋아진다.
  - train.req(질문 문장), train.rep(답변 문장)

- **Test set**
  - 학습의 정확도를 평가하기 위한 테스트 데이터
  - 학습용 데이터를 대표할 수 있어야함
  - test.req(질문 문장), test.rep(답변 문장)

학습 데이터 소개(2/4)

- **Dev set**
  - 성능 개선을 위한 데이터
  - 테스트 데이터를 대표할 수 있어야함
  - dev.req(질문 문장), dev.rep(답변 문장)

- **Vocabulary**
  - 학습과 테스트시에 사용되는 단어집
  - train.req와 train.rep 파일 각각에 대한 단어집이 필요함
  - vocab.req(질문 문장), vocab.rep(답변 문장)

학습 데이터 소개(3/4)

- 단어집 생성(Vocabulary)
  - 질문 단어집(data/vocab.req)

```
python generate_vocab.py < data₩train.req > data₩vocab.req
```

  - 답변 단어집(data/vocab.rep)

```
python generate_vocab.py < data₩train.rep > data₩vocab.rep
```

학습 데이터 소개(4/4)

- 학습 데이터

train.req

```
1 안녕하세요
2 직업이 뭐에요?
3 이름이 뭐에요?
4 반가워요
5
```

train.rep

```
1 네 안녕하세요
2 학생이에요
3 내 이름은 에리카에요
4 나도 반가워요
5
```

- 단어집

vocab.req

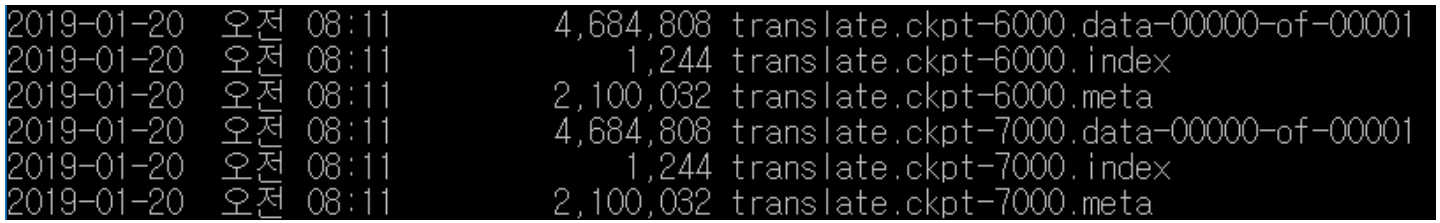
```
1 <unk>
2 <s>
3 </s>
4 뭐에요?
5 직업이
6 이름이
7 안녕하세요
8 반가워요
9
```

vocab.rep

```
1 <unk>
2 <s>
3 </s>
4 학생이에요
5 이름은
6 에리카에요
7 안녕하세요
8 네
9 내
10 나도
11 반가워요
```

데이터 학습

- 데이터 학습은 구글에서 제공하는 NMT를 활용
- 데이터 학습 명령어
  - python -m nmt.nmt --attention=scaled_luong --src=req --tgt=rep --vocab_prefix=data/vocab --train_prefix=data/train --dev_prefix=data/dev --test_prefix=data/test --out_dir=nmt_model --num_train_steps=12000 --steps_per_stats=100 --num_layers=4 --num_units=128 --dropout=0.2 --metrics=bleu
- 데이터 학습이 완료되면 **nmt_model** 디렉토리에 모델 파일들이 생성됨



챗봇 실행

- 챗봇 실행(python chat.py)

```
(venv) C:₩erica_bot-master>python chat.py
# Loading hparams from nmt_model₩hparams
# Updating hparams.override_loaded_hparams: False -> True
# Vocab file data/vocab.req exists
# Vocab file data/vocab.rep exists
```

- 입력 테스트

```
2019-01-21 13:49:55.588673: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports
s TensorFlow binary was not compiled to use: AVX2
  loaded infer model parameters from nmt_model₩translate.ckpt-7000, time 0.09s
> 안녕하세요
네 안녕하세요
> 넌 이름이 뭐니
내 이름은 에리카에요
> 직업은 뭐니
학생이에요 이름은 에리카에요
>
```

과제

1. 데이터를 변경한 후 학습시켜서 자신만의 주제를 가지는 챗봇을 만들어보세요.

2. 웹서버를 붙여서 GUI를 지원하는 챗봇을 만들어보세요. (선택)

종합

GUI를 지원하는 챗봇

CNN과 통합

GUI를 지원하는 챗봇(1/2)

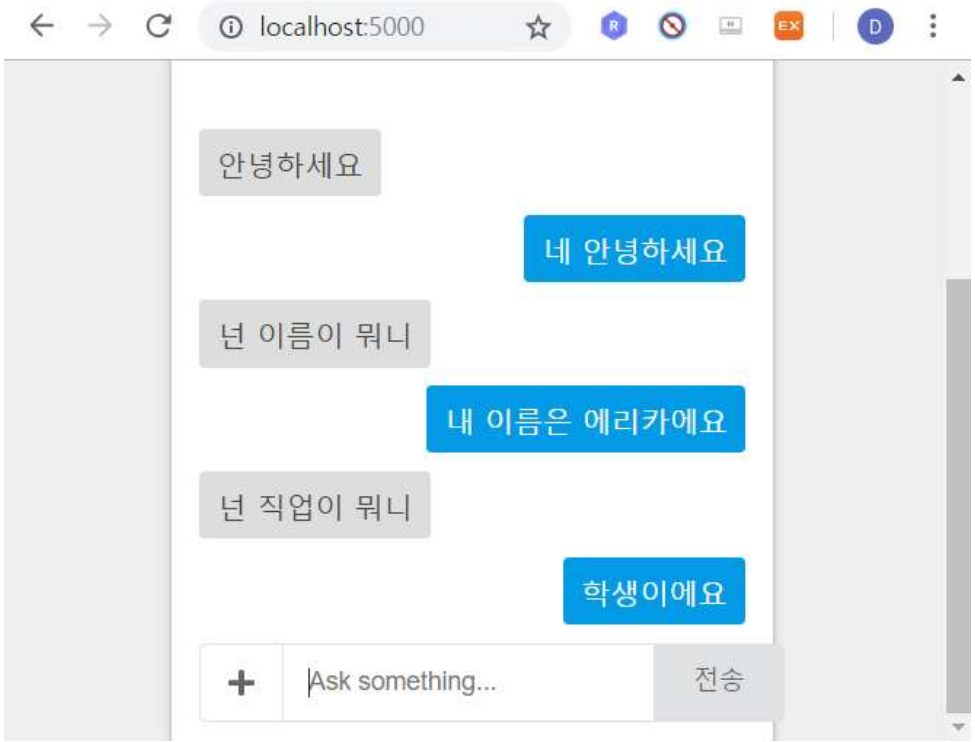
- Flask 기반의 웹서버
- 실행 명령어
  - **python server.py**

```
(venv) C:₩erica_bot-master>python server.py
# Loading hparams from nmt_model/hparams
# Updating hparams.override_loaded_hparams: False -> True
# Updating hparams.out_dir: nmt_model -> nmt_model/
# Vocab file data/vocab.req exists
# Vocab file data/vocab.rep exists
```

```
2019-01-21 14:10:15.454101: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: AVX2
  loaded infer model parameters from nmt_model/translate.ckpt-7000, time 0.09s
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

GUI를 지원하는 챗봇(2/2)

- 웹브라우저 실행 후 http://localhost:5000으로 연결



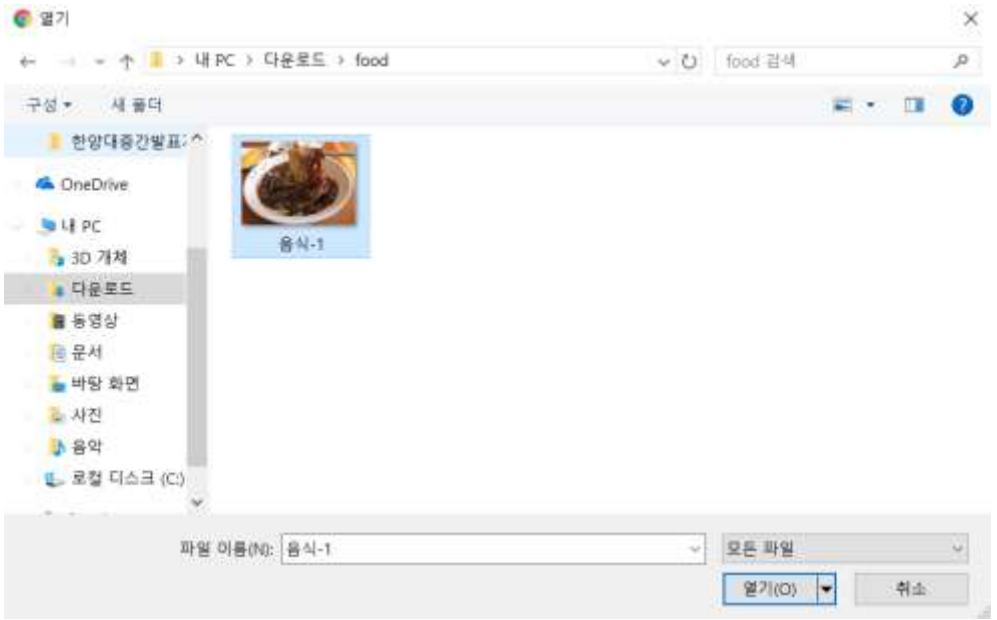
CNN과 통합(1/3)

- Flask 기반의 웹서버
- 실행 명령어
  - **python server.py**

```
(venv) C:₩CNN-master>python server.py
curses is not supported on this machine (please install/reinstall curses for an optimal exp
erience)
WARNING:tensorflow:From C:₩venv₩lib₩site-packages₩tflearn₩initializations.py:119: UniformUn
itScaling.__init__ (from tensorflow.python.ops.init_ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Use tf.initializers.variance_scaling instead with distribution=uniform to get equivalent be
havior.
WARNING:tensorflow:From C:₩venv₩lib₩site-packages₩tflearn₩objectives.py:66: calling reduce_
sum (from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed
in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
2019-01-21 14:33:56.945239: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU s
upports instructions that this TensorFlow binary was not compiled to use: AVX2
model loaded!
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5050/ (Press CTRL+C to quit)
```

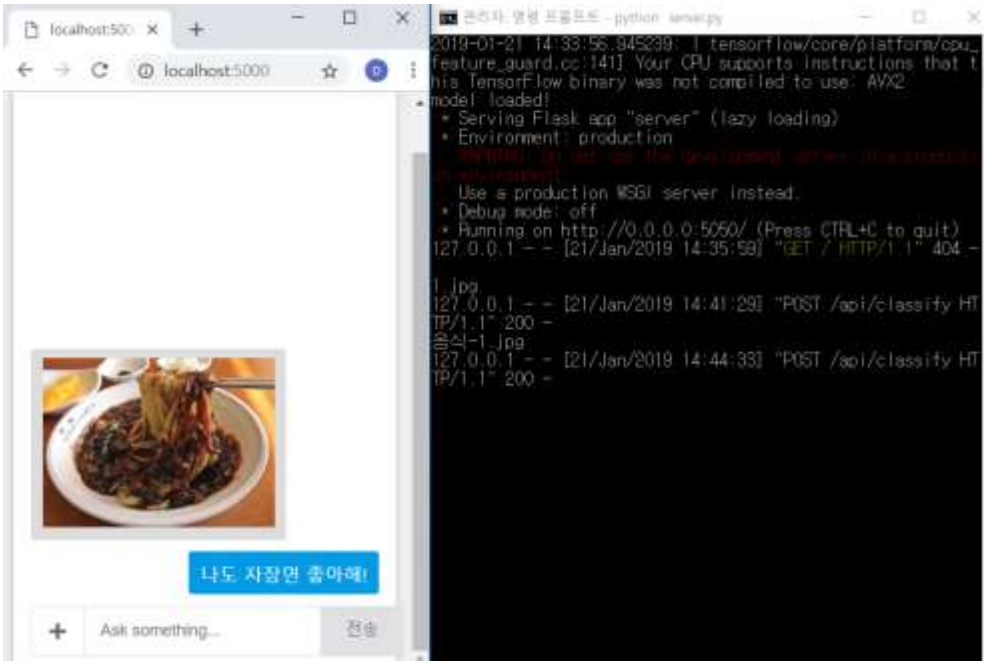
CNN과 통합(2/3)

- 챗봇 창에서 +버튼을 클릭하여 이미지를 추가한다.



CNN과 통합(3/3)

- 추가된 이미지가 CNN 서버에 전달되어 분석 후 자장면이라는 결과를 응답한다.



감사합니다

LIMA
이준영

문의: lima@storyfac.com