

# awesome-spacer

케라스로 띄어쓰기 정복하기

정연준  
2019.07.04



- Keras Korea
- Deep learning
- Computer vision
- Open source
- <https://fuzzythecat.github.io>
- <https://github.com/fuzzythecat/awesome-spacer>

# Motivation



# 한국어 띄어쓰기의 어려움

- 쓰레기가 될만한 물건은 →
- 쓸만한 범퍼 →
- 아마도 카레라이스 뿐일까 한다 →
- 아이디(계정)가 기억이 안나는데요. →

예시 출처: <https://m.blog.naver.com/PostView.nhn?blogId=api0911&logNo=220916415935&proxyReferer=https%3A%2F%2Fwww.google.com%2>



# 한국어 띄어쓰기의 어려움

- 쓰레기가 될만한 물건은 → 쓰레기가 될 만한 물건은 <환경운동연합:동강 취재>
- 쓸만한 범퍼 → 쓸 만한 범퍼 <환경운동연합:생활 실천 지침>
- 아마도 카레라이스 뿐일까 한다. → 아마도 카레라이스뿐일까 한다. <아이팝콘:팝콘투어 / 세계여행>
- 아이디(계정)가 기억이 안나는데요. → 아이디(계정)가 기억이 안 나는데요. <다음·두움막>

```
60      committeeID = 0;  
61      committeeName = "";  
62      displayStipulator = true;  
63      doChanged = false;  
64      draftedStipCount = 0;  
65      editLink = true;
```

예시 출처: <https://m.blog.naver.com/PostView.nhn?blogId=api0911&logNo=220916415935&proxyReferer=https%3A%2F%2Fwww.google.com%2F>

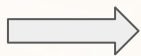
# 케라스 문서 공식 한글화

- <https://www.github.com/keras-team/keras-docs-ko>
- Watch, star, fork, contribute ★



# 한국어 띄어쓰기 교정

- 네이버 띄어쓰기 검사기



API의 부재

- 부산대 띄어쓰기 검사기

자동화의 어려움



NAVER



# 문제 정의



# 한국어 띄어쓰기 교정

- 한국어의 띄어쓰기 교정 문제는 “주어진 문장에서 띄어지지 않는 글자들 중, **본래 띄어써야 하는 부분을 교정하는 것**”으로 한정
- 한국어 띄어쓰기 교정은 연속된 글자들에서 각 글자 다음에 공백이 올지 판단하는 **이진 분류 문제**로 치환 가능
- **띄어쓰기가 잘 되어있는 데이터**로부터 띄어쓰기 패턴을 학습하여 오류 수정 가능

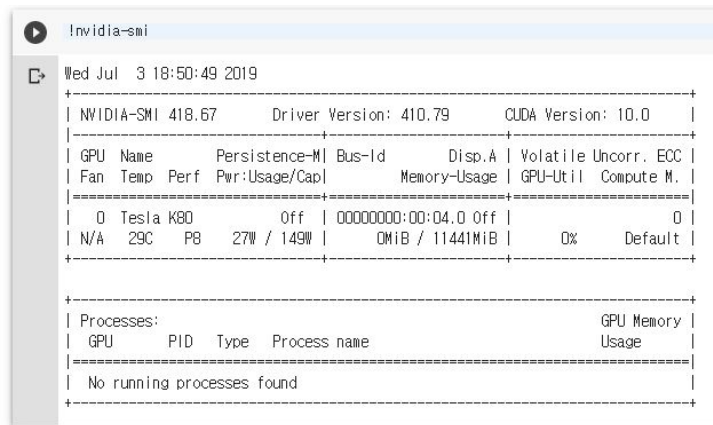
슬라이드 출처: <https://github.com/lovit/soyspacing/blob/master/tutorials/presentation.pdf>

# 환경 설정



# GPU가 없다면, 구글 Colab을

- 주피터 노트북 형식의 개발 환경 제공
- K80 GPU(11.4GB) 제공
- 단일 VM 할당 제한, 12시간 제한
  - 일일 제한이 없기 때문에 12시간 사용 후, 새로운 VM으로 다시 사용 가능
- 윈도우에서도 손쉽게 리눅스 환경 사용 가능
- 구글 드라이브와의 연동으로 데이터 업로드/다운로드 용이
- 자세한 설정은 화면으로



The screenshot shows the output of the `nvidia-smi` command in a terminal window. The window title is "Invidia-smi". The output displays system information for the NVIDIA GPU, including the driver version (410.79) and CUDA version (10.0). It also shows a table of GPU details for the Tesla K80, including fan speed, temperature, power usage, and memory usage. Finally, it shows a table of running processes, which currently lists "No running processes found".

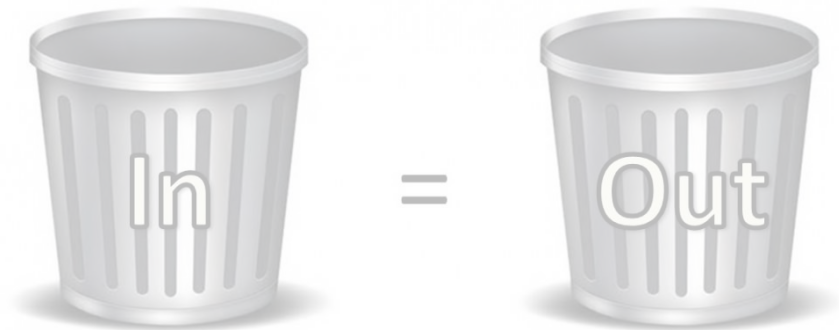
```
Wed Jul 3 18:50:49 2019
+-----+
| NVIDIA-SMI 418.67      Driver Version: 410.79      CUDA Version: 10.0      |
+-----+-----+
| GPU   Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan   Temp   Perf   Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
| 0   Tesla K80           Off | 00000000:00:04:0 Off |             0         |
| N/A   29C    P8      27W / 149W |  0MiB / 11441MiB |           0%    Default |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+
| No running processes found                                     |
+-----+
```

# 데이터셋 준비

# 어떤 데이터셋을 사용해야 할까?

- 정답 데이터 확보의 중요성
  - Garbage in, garbage out



- 어떤 데이터가 있나?
  1. 인터넷 뉴스 크롤링
  2. 위키피디아 한글 문서
  3. 세종 말뭉치 데이터
- 데이터셋 선택이 갖는 의미
  - 트위터 데이터로 학습한 모델을 신문 기사에 적용할 수 있을까?

# 세종 코퍼스 사용하기

- <https://ithub.korean.go.kr/user/guide/corpus/guide1.do>
  - 말뭉치들이 여러 게시판에 나뉘어 저장돼있어 사용이 까다롭다
- 전체 데이터가 필요하다면
  - `$ git clone https://github.com/coolengineer/sejong-corpus`
  - `$ cd sejong-corpus`
  - `$ make`
  - `$ mae dic`
  - `$mae diff`

감사!

압도적  
감사..!

# 세종 코퍼스 사용하기

- [ISSUE] Mac/Windows 환경에서는 **sejong-corpus** 실행 불가
- 샘플 다운로드 링크(150개의 파일, 18MB)
  - <https://drive.google.com/file/d/1dmATco3eysvKZ3EtuTgzPT0bgmoL-ymo/view?usp=sharing>

감사!

압도적  
감사..!

# 샘플 데이터 살펴보기

- 모든 텍스트 파일에서 텍스트만 얻으려면 어떻게 해야할까?
- sejong-corpus 텍스트 파일 예시
  - sejong-corpus/corpus-utf8/2BA90A03.txt
- HTML 형식
  - 파싱 필요
- 텍스트는 <p></p> 태그로 추출
- 파일별로 추출한 텍스트를 하나로 머지

```
filenames = glob('data/*.txt')
print('Total {} text files found.\n'.format(len(filenames)))

for p in filenames:
    with open(p, 'r', encoding='utf-8') as f:
        html = f.readlines()
        # Make it a single string
        html = ''.join(html)
        print(html[1950:2500])
        break
```

Total 150 text files found.

date>1990/01/10</date>

<page>01</page>

</source>

<head>만물상</head>

<p>일본 회사와 거래가 있는 우리나라 회사서 업무 협의차 직원 한 명이 동경 출장을 가면 상대방 회사의 중역들이 공항까지 나와 정중하게 영접을 한다. 그러나 두 명 이상의 한국인들이 일본에 가면 일본 회사 쪽에서는 아무도 공항에 나오지 않는다.</p>

<p>단 한 명의 한국인은 능력도 있고 순발력도 있지만 두 명 이상이 되면 서로 자기만을 내세워서 그냥 내버려 두어도 일본 회사가 득을 볼 수 있기 때문이다. 누군가가 만들어낸 말이 분명하지만 개개인으로는 누구에게도 뒤지지 않는 한국인들의 특성을 잘 비유한 일화다.</p>

<p>외국인 직업관광 안내원들은 일본인 단체와 한국인 단체는 같은 유교문화권 사람들 같지 않다고 평한다. 일본인들은 항상 리더가 있어야 무슨 일이 있을 때는 리더와 상의하면 모든 일이 잘 풀리는데 한국인들은 모든 사람이 리더라는 이야기다. 누가 한마디 하면 다른 사람들은 요새 유행어로 "잘났어, 정말"이라는 반응만을 보인다.</p>



# 샘플 데이터 살펴보기

- <p> 태그를 일괄적으로 지우기 위한 정규표현식
  - <p>임의의 문장. 하나. 두개.</p>

```
filenames = glob('data/*.txt')
print('Total {} text files found.\n'.format(len(filenames)))

data = []

# Compiled regex to filter tags
treg = re.compile('<\w*>|</\w*>')

for p in filenames:
    with open(p, 'r', encoding='utf-8') as f:
        html = f.readlines()
        # Make it a single string
        html = ''.join(html)

        # Parse HTML file
        soup = BeautifulSoup(html, 'html.parser')

        # Use paragraphs only
        texts = soup.find_all('p')
        texts = [treg.sub('', str(t)).strip() for t in texts]

        # Filter zero length input sentences
        # and convert to lowercase
        texts = [t.lower() for t in texts if t]

    data.extend(texts)
```

Total 150 text files found.

# 샘플 데이터 살펴보기

1. HTML 파일 파싱
2. <p> 태그 추출
3. 태그 제거
4. 길이가 0인 문장 제거
  - a. <p>\n</p>
5. 모든 영문자는 소문자로 대체

```
filenames = glob('data/*.txt')
print('Total {} text files found.\n'.format(len(filenames)))

data = []

# Compiled regex to filter tags
treg = re.compile('<\w*>|</\w*>')

for p in filenames:
    with open(p, 'r', encoding='utf-8') as f:
        html = f.readlines()
        # Make it a single string
        html = ''.join(html)

        # Parse HTML file
        soup = BeautifulSoup(html, 'html.parser')

        # Use paragraphs only
        texts = soup.find_all('p')
        texts = [treg.sub('', str(t)).strip() for t in texts]

        # Filter zero length input sentences
        # and convert to lowercase
        texts = [t.lower() for t in texts if t]

    data.extend(texts)
```

Total 150 text files found.

# 샘플 데이터 살펴보기

- 그냥 찍어도 77%
- 평균 122 글자

```
print('Total {} sentences.\n'.format(len(data)))

size = sum([len(t) for t in data])
print('Total {} characters.\n'.format(size))

avg_len = size // len(data)
print('{} characters/sentence avg.\n'.format(avg_len))

whitespaces = [c for t in data for c in t if c == ' ']
ratio = len(whitespaces) / size * 100
print('Of {} characters, {}% are white spaces.\n'.format(size, ratio))

print('Baseline accuracy @ {}%.\n'.format((100 - ratio)))

print('1: {}\n2: {}'.format(data[0], data[1]))
```

Total 169022 sentences.

Total 20646352 characters.

122 characters/sentence avg.

Of 20646352 characters, 22.7164585782515% are white spaces.

Baseline accuracy @ 77.2835414217485%.

1: 일본 회사와 거래가 있는 우리나라 회사서 업무 협의차 직원 한 명이 동경 출장을 가면 상대방 회사의 중역들이 공항까지 나와 정중하게 영접을 한다. 그러나 두 명 이상의 한국인들이 일본에 가면 일본 회사 쪽에서는 아무도 공항에 나오지 않는다.  
2: 단 한 명의 한국인은 능력도 있고 순발력도 있지만 두 명 이상이 되면 서로 자기만을 내세워서 그냥 내버려 두어도 일본 회사가 득을 볼 수 있기 때문이다. 누군가가 만들어낸 말이 분명하지만 개개인으로는 누구에게도 뒤지지 않는 한국인들의 특성을 잘 비유한 일화다.

# tf2.0-beta + keras



# 어떻게 해결할까?

- 딥러닝 프레임워크
  - Keras + tf2.0-beta
  - Keras
  - PyTorch
  - CNTK

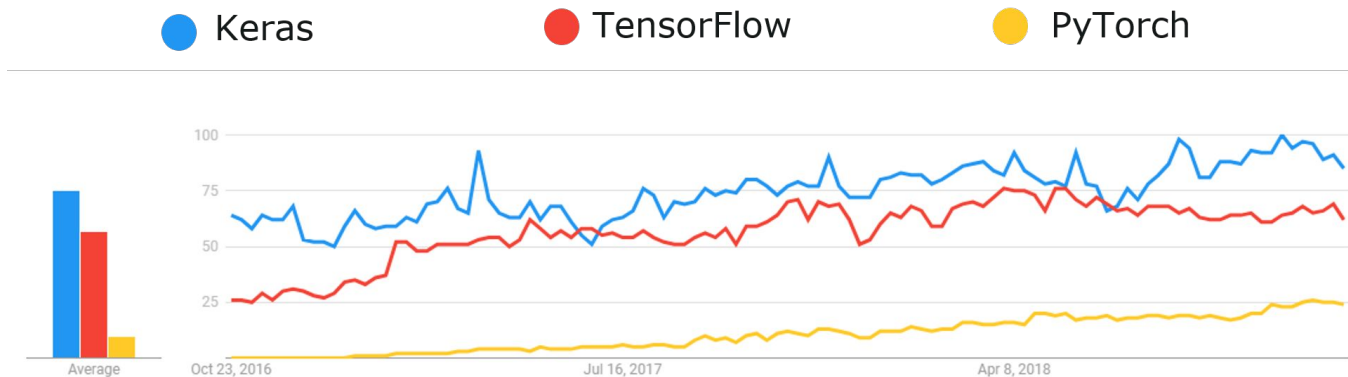
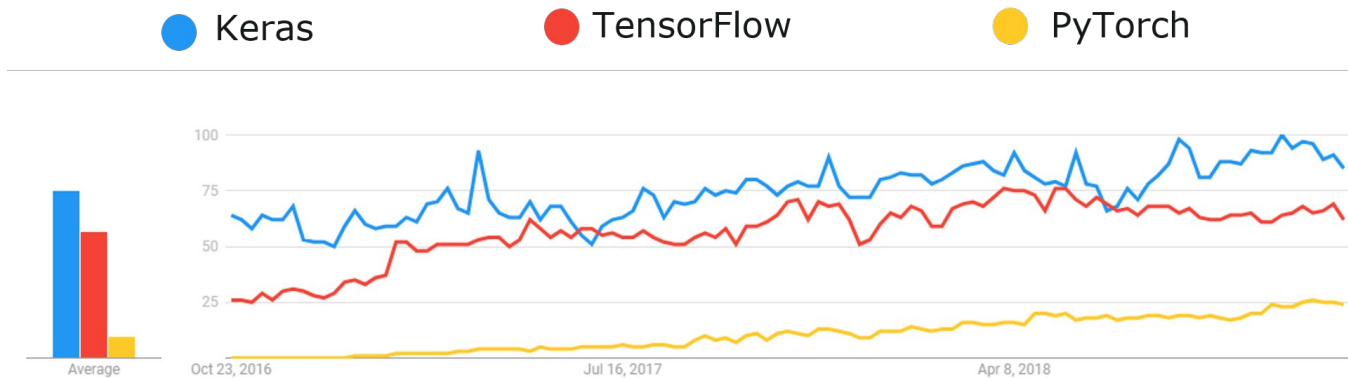
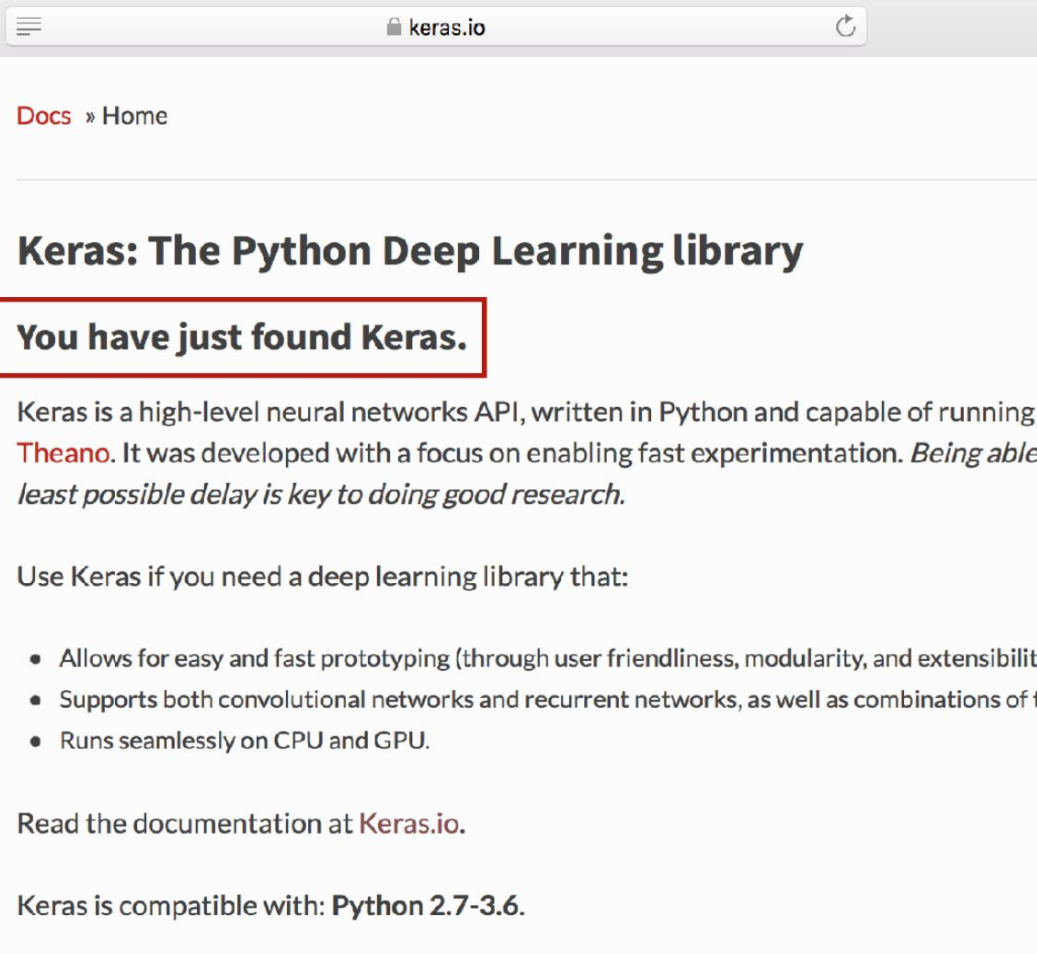
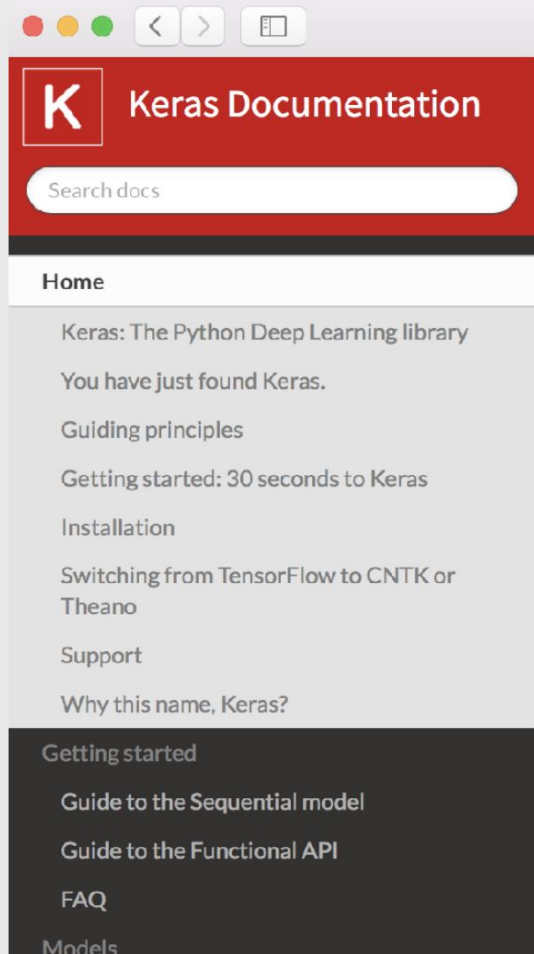


차트 출처: <https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch>

# 어떻게 해결할까?

- 딥러닝 프레임워크
  - Keras + tf2.0-beta
  - Keras
  - PyTorch
  - CNTK





# 그래서, Keras 뭐예요?

- 구글 엔지니어 **Francois Chollet**가 개발
- 파이썬으로 구현된 쉽고 간결한 딥러닝 API
- 내부엔진으로 **TesnsorFlow, Theano, CNTK** 사용

슬라이드 출처: [https://www.slideshare.net/MljeongJeon1/keras-81639526?from\\_action=save](https://www.slideshare.net/MljeongJeon1/keras-81639526?from_action=save)





# 코라서, Keras 왜 써요?

**TensorFlow**

```
#####
##CNN in TensorFlow Only##
#####

import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

###1. Load data set, and split it if necessary
mnist = input_data.read_data_sets("MNIST_data/")

###2. We create a holder, a container to place the computation activities in tensorflow
###identifying format and tensor's r/c, null means any kind
VISIBLE_NODES = 784
HIDDEN_NODES = 400
x = tf.placeholder("float", shape=(None, VISIBLE_NODES))
y = tf.placeholder("float", shape=(None, 10))

###3. We identify weights and biases with tensor shape, start with 0
weights = tf.Variable(tf.random_normal([VISIBLE_NODES, HIDDEN_NODES],
mean=0.0, stddev=1. / VISIBLE_NODES))
hidden_bias = tf.Variable(tf.zeros([HIDDEN_NODES]))
visible_bias = tf.Variable(tf.zeros([VISIBLE_NODES]))

###4. set up the sigmoid model and multiply x and W with matmul function, building the
###hidden layer and reconstruction layer
hidden_activation = tf.nn.sigmoid(tf.matmul(x, weights) + hidden_bias)
visible_reconstruction = tf.nn.sigmoid(tf.matmul(hidden_activation, tf.transpose(weights))
+ visible_bias)
final_hidden_activation = tf.nn.sigmoid(tf.matmul(visible_reconstruction, weights)
+ hidden_bias)

###5. This process can be understood as being two phases of learning
###positive and negative or, more poetically, waking and sleeping
positive_phase = tf.matmul(tf.transpose(x), hidden_activation)
negative_phase = tf.matmul(tf.transpose(visible_reconstruction), final_hidden_activation)
LEARNING_RATE = 0.01
weight_update = weights.assign_add(LEARNING_RATE *
(positive_phase - negative_phase))
visible_bias_update = visible_bias.assign_add(LEARNING_RATE *
tf.reduce_mean(x - visible_reconstruction, 0))
hidden_bias_update = hidden_bias.assign_add(LEARNING_RATE *
tf.reduce_mean(hidden_activation - final_hidden_activation, 0))
###6. Now we create the operations for scaling the hidden and visible biases, with loss
###function feedback
train_op = tf.group(weight_update, visible_bias_update, hidden_bias_update)
loss_op = tf.reduce_sum(tf.square(x - visible_reconstruction))

###7. We start the session
session = tf.Session()
session.run(tf.global_variables_initializer())
current_epochs = 0

###8. Run the session
for i in range(20):
    total_loss = 0
    while mnist.train.epochs_completed <= current_epochs:
        batch_inputs, batch_labels = mnist.train.next_batch(100)
        reconstruction_loss = session.run([train_op, loss_op], feed_dict={input_placeholder: batch_inputs})
        total_loss += reconstruction_loss
    print("epochs %s loss %s" % (current_epochs, reconstruction_loss))
    current_epochs = mnist.train.epochs_completed
```

VS

**Keras**

```
#####
##CNN in TensorFlow Keras##
#####

###1. Load Data and Split Data
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.utils import np_utils

(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

###2. Preprocess
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
classes = 10
Y_train = np_utils.to_categorical(Y_train, classes)
Y_test = np_utils.to_categorical(Y_test, classes)

###3. Set up parameters
input_size = 784
batch_size = 100
hidden_neurons = 400
epochs = 30

###4. Build the model
model = Sequential()
model.add(Dense(hidden_neurons, input_dim=input_size))
model.add(Activation('relu'))
model.add(Dense(classes, input_dim=hidden_neurons))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
metric='accuracy', optimizer='adadelta')
model.fit(X_train, Y_train, batch_size=batch_size, epochs=epochs, verbose=1)

###5. Test
score = model.evaluate(X_test, Y_test, verbose=1)
print("\nTest accuracy:", score[1])
#Test accuracy: 0.983
```

<https://charleshalao.wordpress.com/2017/06/19/cnn-and-cnn-of-tensorflowkeras-with-mnist-data/>

슬라이드 출처: [https://www.slideshare.net/MljeongJeon1/keras-81639526?from\\_action=save](https://www.slideshare.net/MljeongJeon1/keras-81639526?from_action=save)



# 1. Word Vectors



# One-hot Encoding

- 딥러닝은 결국 복잡한 행렬 연산의 집합
  - 텍스트를 수치화할 수 있어야한다.
  - 문자간에  $+$ ,  $-$ ,  $*$  등 연산이 정의되어있지 않다.
- 텍스트  $\rightarrow$  벡터

# One-hot Encoding

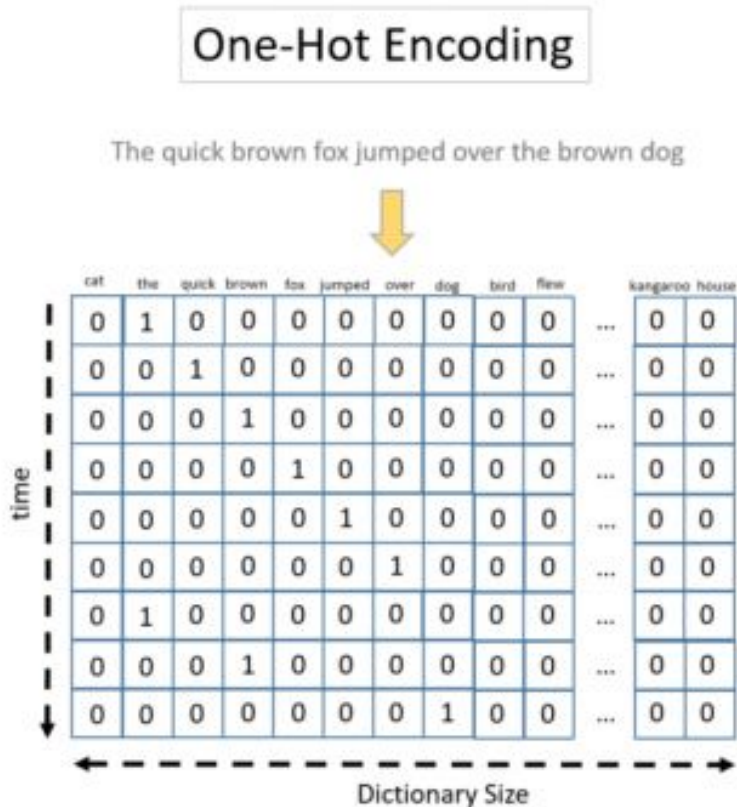
- 딥러닝은 결국 복잡한 행렬 연산의 집합
  - 텍스트를 수치화할 수 있어야한다.
  - 문자간에 +, -, \* 등 연산이 정의되어있지 않다.
- 텍스트 → 벡터

## One-hot encoding

	cat	mat	on	sat	the
<b>the</b> =>	0	0	0	0	1
<b>cat</b> =>	1	0	0	0	0
<b>sat</b> =>	0	0	0	1	0
...					

# One-hot Encoding

- 딥러닝은 결국 복잡한 행렬 연산의 집합
  - 텍스트를 수치화할 수 있어야한다.
  - 문자간에 +, -, \* 등 연산이 정의되어있지 않다.
- 텍스트 → 벡터
- 단어의 수가 많아지면?



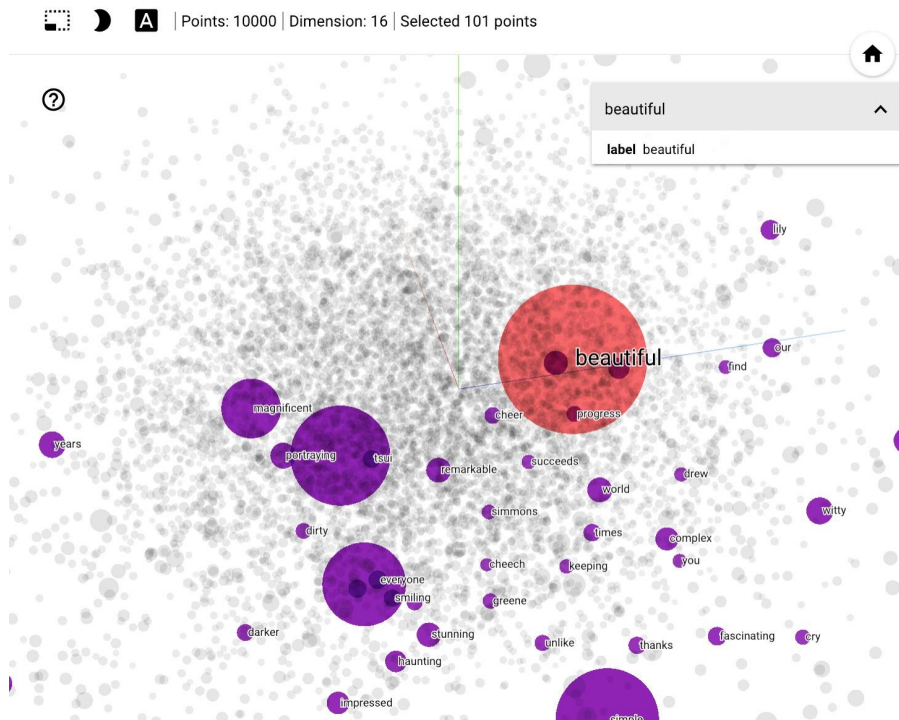
# Word Embedding

## A 4-dimensional embedding

cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4

...

...

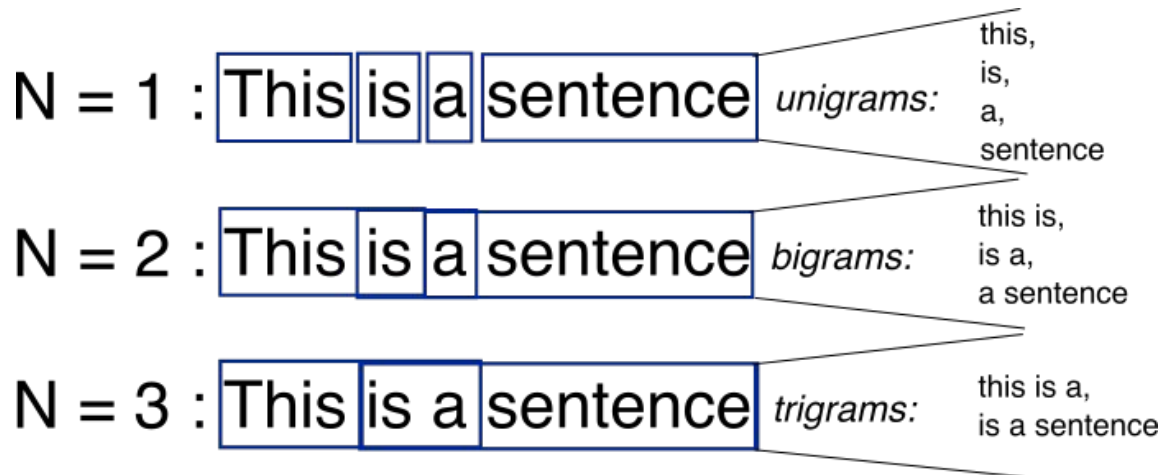


## 2. N-gram



# N-gram

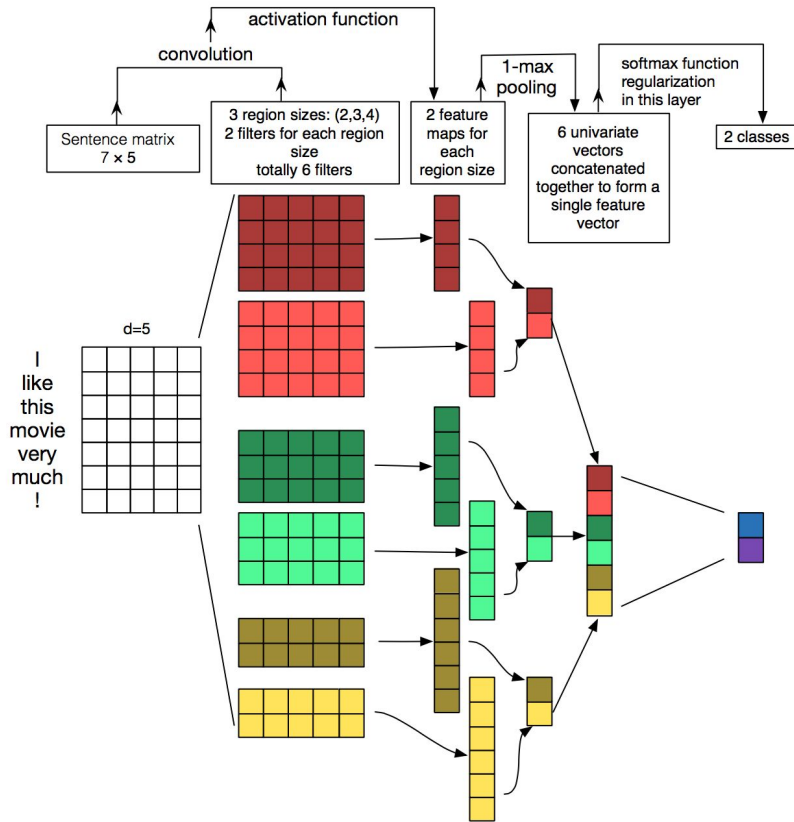
- 아버지가방에들어가셨다
  1. 아버지가 방에 *if* ‘가 방에’ > ‘가방에’
  2. 아버지 가방에 *if* ‘가 방에’ < ‘가방에’





# N-gram

- 1-D Convolution으로 N-gram 학습 가능
- 필터의 크기 : N
  - $N = 1$  : (아), (버), (지), (가)
  - $N = 2$  : (아,버), (버,지), (지,가)
  - $N = 3$  : (아, 버, 지), (버, 지, 가)
- 필터의 개수 : 학습된 N-gram의 개수
  - (아,버), (버,지), (지,가), ...

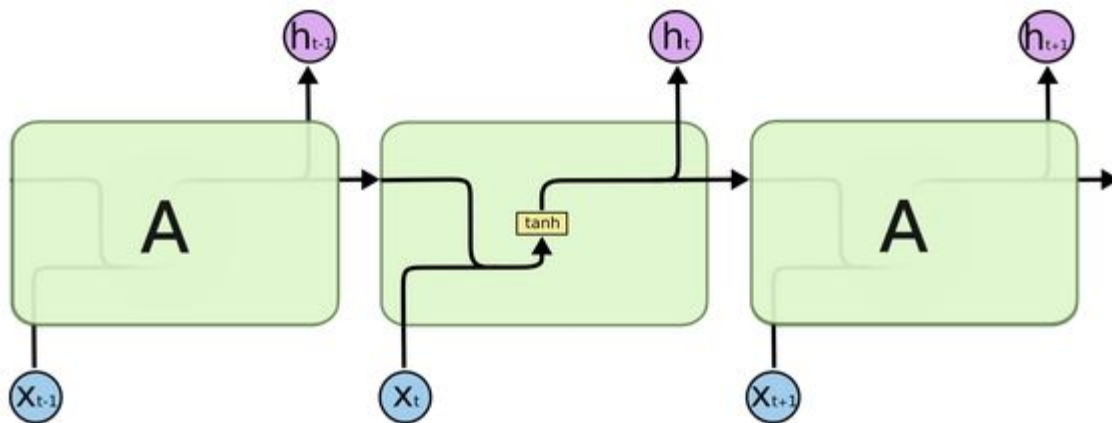


# 3. LSTM



# LSTM

- 시퀀스 데이터 처리에 강하다
- 그런데 느리다
- 많이 느리다



The repeating module in a standard RNN contains a single layer.

# 결론

- N-gram을 학습할 수 있다
- 글자의 연속 -> 학습된 N-gram의 시퀀스(최소 단위 변환)
- LSTM으로 시퀀스 데이터 학습
- 각 글자(N-gram)에 대해 이진분류 진행

# 4. Implementation

# 5. Training

# 6. Test → .ipynb를 보며 실습

코드

- [www.github.com/fuzzythecat/awesome-spacer](https://www.github.com/fuzzythecat/awesome-spacer)

학습된 모델

- <https://drive.google.com/file/d/1vIKa7A0F3W3cAAi9E1V6ARXiDeDtHiiW/view?usp=sharing>



# 데이터 입력

- 모든 텍스트 파일에서 텍스트만 얻으려면 어떻게 해야할까?
- **sejong-corpus** 텍스트 파일 예시
  - sejong-corpus/corpus-utf8/2BA90A03.txt
- HTML 형식
  - 파싱 필요
- 텍스트는 `<p></p>` 태그로 추출
- 파일별로 추출한 텍스트를 하나로 머지

```
filenames = glob('data/*.txt')
print('Total {} text files found.\n'.format(len(filenames)))

for p in filenames:
    with open(p, 'r', encoding='utf-8') as f:
        html = f.readlines()
        # Make it a single string
        html = ''.join(html)
        print(html[1950:2500])
        break
```

Total 150 text files found.

date>1990/01/10</date>

<page>01</page>

</source>

<head>만물상</head>

<p>일본 회사와 거래가 있는 우리나라 회사서 업무 협의차 직원 한 명이 동경 출장을 가면 상대방 회사의 중역들이 공항까지 나와 정중하게 영접을 한다. 그러나 두 명 이상의 한국인들이 일본에 가면 일본 회사 쪽에서는 아무도 공항에 나오지 않는다.</p>

<p>단 한 명의 한국인은 능력도 있고 순발력도 있지만 두 명 이상이 되면 서로 자기만을 내세워서 그냥 내버려 두어도 일본 회사가 득을 볼 수 있기 때문이다. 누군가가 만들어낸 말이 분명하지만 개개인으로는 누구에게도 뒤지지 않는 한국인들의 특성을 잘 비유한 일화다.</p>

<p>외국인 직업관광 안내원들은 일본인 단체와 한국인 단체는 같은 유교문화권 사람들 같지 않다고 평한다. 일본인들은 항상 리더가 있어야 무슨 일이 있을 때는 리더와 상의하면 모든 일이 잘 풀리는데 한국인들은 모든 사람이 리더라는 이야기다. 누가 한마디 하면 다른 사람들은 요새 유행어로 "잘났어, 정말"이라는 반응만을 보인다.</p>

# Future work



# 성능을 어떻게 높일까?

1. Colab 기준 400Mb(약 1억 글자)를 학습 시, 1 epoch에 약 2시간
2. 도메인에 특정한 데이터셋 확보
  - a. 위키 데이터로는 구어체에 대한 분류는 부정확할 것
3. LSTM → Bidirectional
4. Alphabet → <w>
5. Digits → <d>
6. Class별 Weight(0.85 : 0.15)
7. Etc...



# References

[https://pdfs.semanticscholar.org/7a10/2ac662b3446830a80284c11d9b69ff1417d2.pdf?\\_ga=2.62396460.167641117.1561461262-789995954.1561461262](https://pdfs.semanticscholar.org/7a10/2ac662b3446830a80284c11d9b69ff1417d2.pdf?_ga=2.62396460.167641117.1561461262-789995954.1561461262)

[https://www.tensorflow.org/beta/tutorials/text/word\\_embeddings?hl=ko](https://www.tensorflow.org/beta/tutorials/text/word_embeddings?hl=ko)

[https://www.tensorflow.org/beta/tutorials/keras/basic\\_text\\_classification?hl=ko](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification?hl=ko)

[https://www.tensorflow.org/beta/tutorials/text/text\\_classification\\_rnn?hl=ko](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn?hl=ko)

[https://wenku.baidu.com/view/7bbc2f6825c52cc58bd6bec6\\_](https://wenku.baidu.com/view/7bbc2f6825c52cc58bd6bec6_)

<https://github.com/lovit/soyspacing/blob/master/tutorials/presentation.pdf>

<https://github.com/coolengineer/sejong-corpus>

<https://www.slideshare.net/MijeongJeon1/ios-103594100>

<https://www.slideshare.net/MijeongJeon1/ios-84097139>

# Call for contribution

All contributions are welcome!

Submit PR @ <https://github.com/fuzzythecat/awesome-spacer>