# Deep Learning based Hotel Review Classification

국민대학교 정보보호연구실
정성민

# Outline

- Introduction & Members
- Dataset
- Related Algorithm
- Model Architecture
- Experiment Result
- Q&A
- Appendix

# Introduction

**호텔 리뷰가 주어지면 평점을 예측해주는 딥러닝 분류 모델을 만들자.**

Really lovely hotel! Friendly and helpful!!

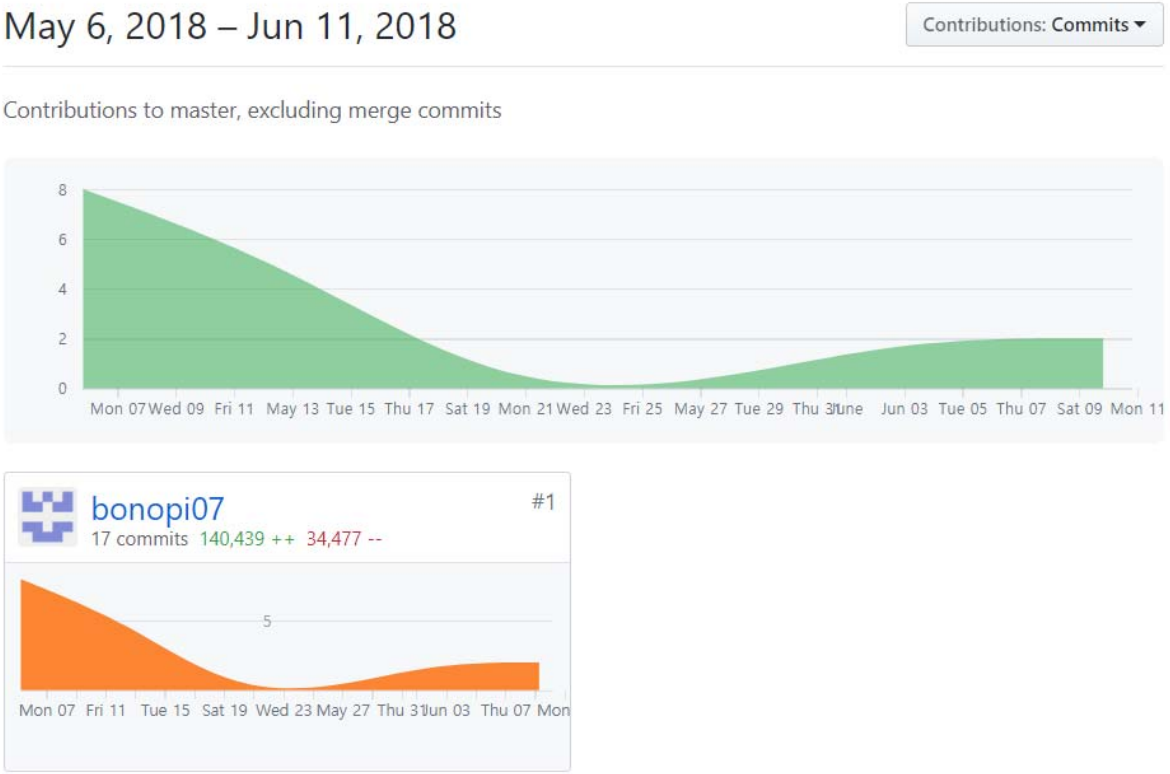
## Sentiment Analysis

Sheets were dirty. The breakfast tasted terrible!

# Members

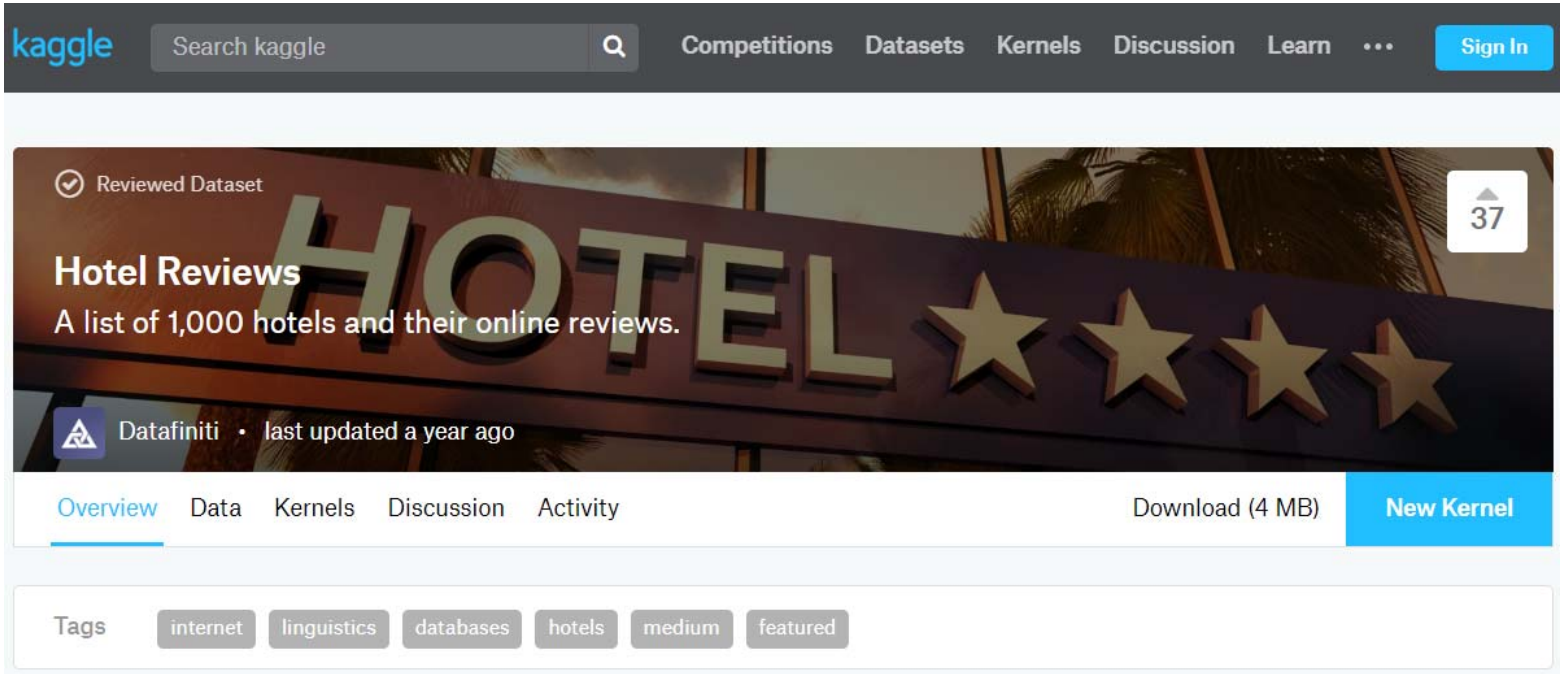
- Seongmin Jeong (Only One)
  - 정보보호연구실 석사과정

- Member's(=My) role
  - 데이터 수집 및 가공
  - Deep Learning 학습 모델 설계
  - 모델 검증 실험
  - hyper parameter 튜닝



# Dataset

- Kaggle Datasets 에서 무료 제공 (Datafiniti 회사)
- 1,000 개의 Hotel에 대한 Review 데이터 (data: 약 36,000개, 19 columns)




출처: https://www.kaggle.com/datafiniti/hotel-reviews

# Dataset

- Kaggle Datasets 에서 무료 제공 (Datafiniti 회사)
- 1,000 개의 Hotel에 대한 Review 데이터 (data: 약 36,000개, 19 columns)

Preview (first 100 rows) | Column Metadata | Column Metrics

| reviews.rating | reviews.text | reviews.title |
|---|---|---|
| 4 | Pleasant 10 min walk along the sea front to the Water Bus. restaurants etc. Hotel was comfortable breakfast was good - quite a variety. Room aircon didn't work very well. Take mosquito repelant! | Good location away from the crouds |
| 5 | Really lovely hotel. Stayed on the very top floor and were surprised by a Jacuzzi bath we didn't know we were getting! Staff were friendly and helpful and the included breakfast was great! Great location and great value for money. Didn't want to leave! | Great hotel with Jacuzzi bath! |
| 5 | Ett mycket bra hotell. Det som drog ner betyget var att vi fick ett rum under taksarna dr det endast var full sthjd i 80 av rummets yta. | Lugnt l��ge |

**리뷰 평점, 리뷰 텍스트 두 정보만 사용**

출처: https://www.kaggle.com/datafiniti/hotel-reviews

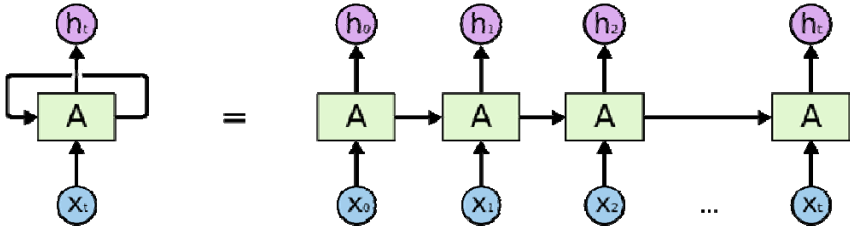
# Related Algorithm

- Data: Text (length: dynamic)
- Type: English, Spanish

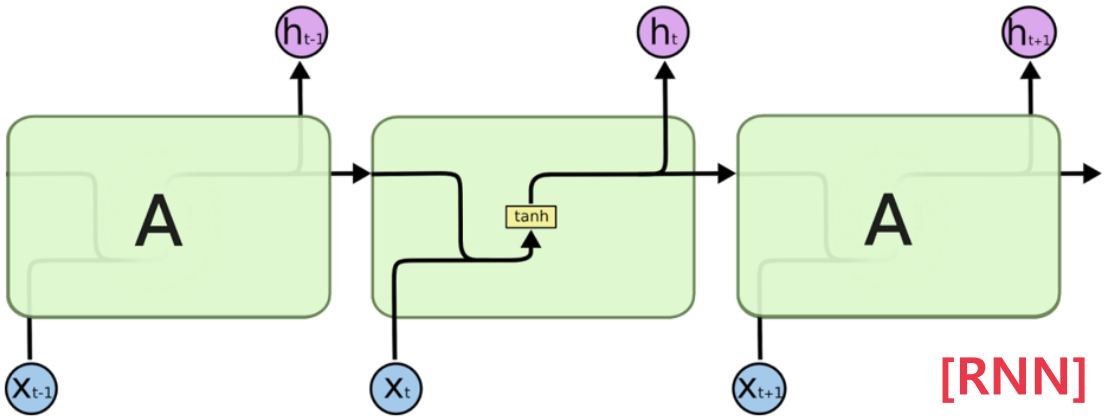
**리뷰 데이터를 이해한 후 평점을 예측해야 한다.**

- Model: Many to one
- Type: Classification

**LSTM 기반의 Many-to-one 학습 모델!**



# Related Algorithm

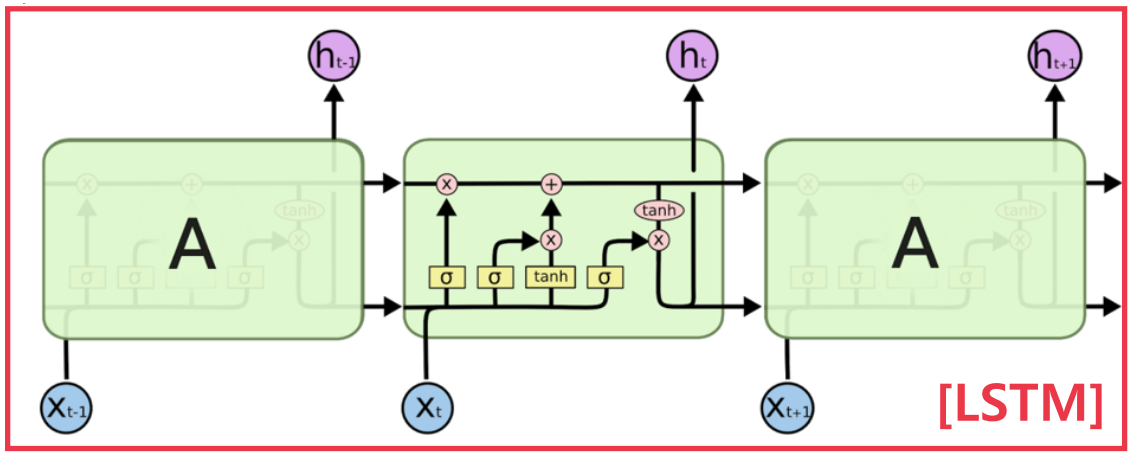



[RNN]

**Vanishing Gradient Problem**

출처: https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

# Related Algorithm




+ TensorFlow

출처: https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

# Model Architecture

- LSTM Layer #
  - Single (1 Layer) → Stacked (e.g. 3 layer)
- Data Dimension
  - char-level : 44차원
  - word-level : 300차원
- Loss Function
  - Softmax cross entropy (with Adam Optimizer)

```
def inference_LSTM(x, x_seq_length, max_seq_length, data_dim, hidden_size, num_rnn_layers,
                   output_size, num_words, prob=1.0, train_flag=False):
    # x_one_hot = tf.one_hot(x, data_dim, name='x_one_hot')  # (batch size, seq_length, data_dim)
    word_embedding = tf.get_variable('word_embedding', [num_words, data_dim])
    x_embedding_vector = tf.nn.embedding_lookup(word_embedding, x)  # [input word #, embedding size]

    cell = rnn.BasicLSTMCell(num_units=hidden_size, state_is_tuple=True,
                             activation=tf.tanh)
    cells = [cell for _ in range(num_rnn_layers)]
    cells = rnn.MultiRNNCell(cells)
    # cell = rnn.DropoutWrapper(cell=cell, output_keep_prob=prob)

    outputs, state = tf.nn.dynamic_rnn(cells, x_embedding_vector,
                                       sequence_length=x_seq_length,
                                       dtype=tf.float32)  # state: usually use final values
    # outputs : (?, ?, 300)
    # state   : (3, ?, 300) ?

    y_reshape = tf.reshape(outputs, [-1, max_seq_length * hidden_size])  # 3278
    # y_reshape = tf.reshape(state, [-1, 6*hidden_size])  # why 6?

    y_fc = layers.fully_connected(inputs=y_reshape,
                                  num_outputs=output_size,
                                  activation_fn=None)

    return y_fc
```

LSTM inference 소스코드
출처:https://github.com/bonopi07/2018-1_advML_project/blob/master/src/network.py

# Experiment Result (Accuracy)

- K-fold Cross Validation (K =5)
  - 모델 정확도 및 데이터셋 검증
- LSTM Layer #
  - Stacked (e.g. 3 layer)
- Data Dimension
  - word-level : 300차원 (word embedding vector)
- Loss Function
  - Softmax cross entropy (with Adam Optimizer)

| 실험 step | 정확도 (accuracy) |
|---|---|
| 1 | 89.17 |
| 2 | 88.42 |
| 3 | 91.03 |
| 4 | 84.98 |
| 5 | 86.11 |
| 평균 | 87.94 |

# Contact us

Address:
**7-613, Information Security Lab,**
**Kookmin University**

Phone Number:
**(010) 6667 - 5204**

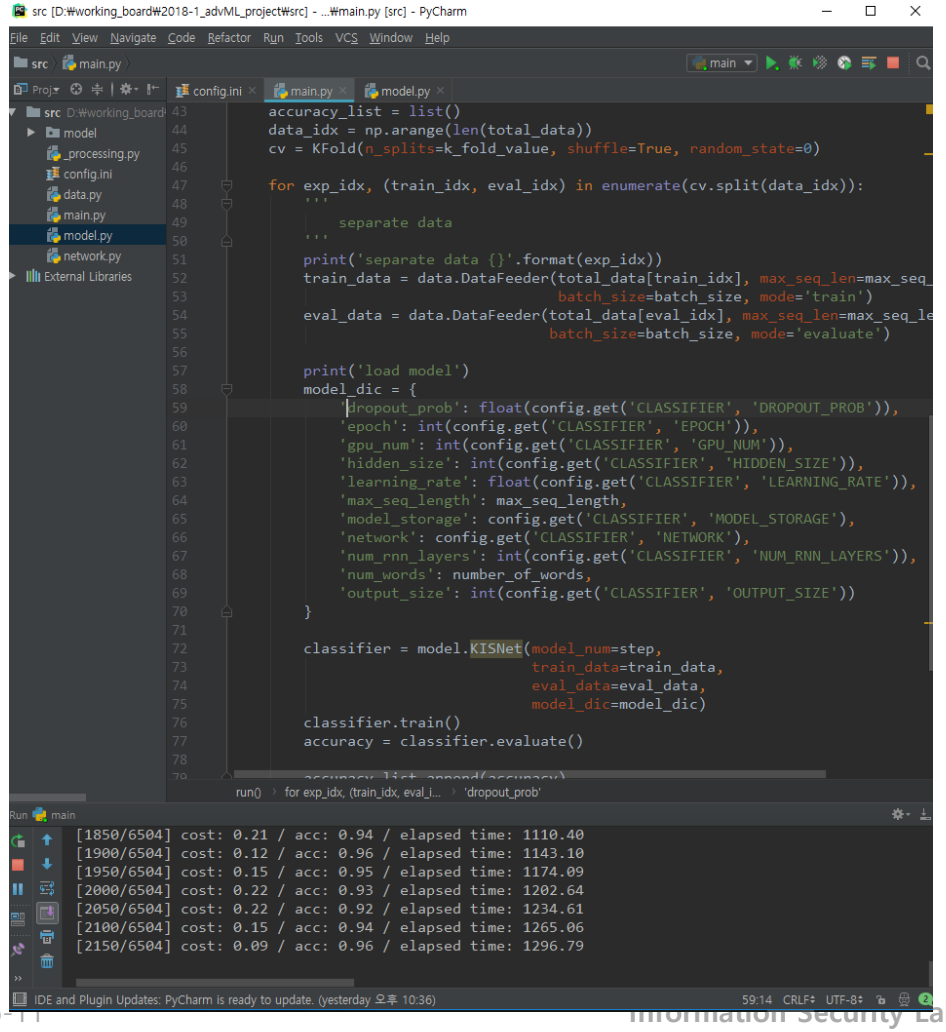
E-Mail
**bonopi07@kookmin.ac.kr**

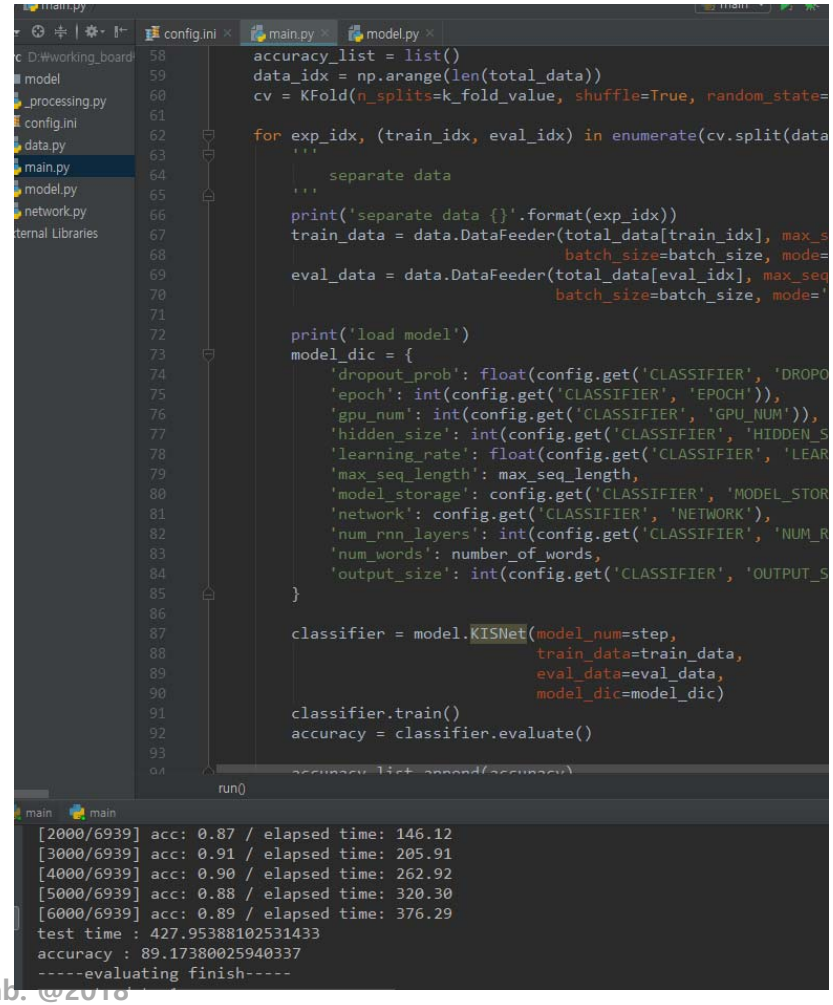
Github
**https://github.com/bonopi07**

## Any Questions?

## Thank you for listening!

# 시연 장면


src [D:₩working_board₩2018-1_advML_project₩src] - ...₩main.py [src] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
config.ini
main.py
model.py
accuracy_list = list()
data_idx = np.arange(len(total_data))
cv = KFold(n_splits=k_fold_value, shuffle=True, random_state=0)
for exp_idx, (train_idx, eval_idx) in enumerate(cv.split(data_idx)):
separate data
print('separate data {}'.format(exp_idx))
train_data = data.DataFeeder(total_data[train_idx], max_seq_len=max_seq_
batch_size=batch_size, mode='train')
eval_data = data.DataFeeder(total_data[eval_idx], max_seq_len=max_seq_le
batch_size=batch_size, mode='evaluate')
print('load model')
model_dic = {
'dropout_prob': float(config.get('CLASSIFIER', 'DROPOUT_PROB')),
'epoch': int(config.get('CLASSIFIER', 'EPOCH')),
'gpu_num': int(config.get('CLASSIFIER', 'GPU_NUM')),
'hidden_size': int(config.get('CLASSIFIER', 'HIDDEN_SIZE')),
'learning_rate': float(config.get('CLASSIFIER', 'LEARNING_RATE')),
'max_seq_length': max_seq_length,
'model_storage': config.get('CLASSIFIER', 'MODEL_STORAGE'),
'network': config.get('CLASSIFIER', 'NETWORK'),
'num_rnn_layers': int(config.get('CLASSIFIER', 'NUM_RNN_LAYERS')),
'num_words': number_of_words,
'output_size': int(config.get('CLASSIFIER', 'OUTPUT_SIZE'))
}
classifier = model.KISNet(model_num=step,
train_data=train_data,
eval_data=eval_data,
model_dic=model_dic)
classifier.train()
accuracy = classifier.evaluate()
[1850/6504] cost: 0.21 / acc: 0.94 / elapsed time: 1110.40
[1900/6504] cost: 0.12 / acc: 0.96 / elapsed time: 1143.10
[1950/6504] cost: 0.15 / acc: 0.95 / elapsed time: 1174.09
[2000/6504] cost: 0.22 / acc: 0.93 / elapsed time: 1202.64
[2050/6504] cost: 0.22 / acc: 0.92 / elapsed time: 1234.61
[2100/6504] cost: 0.15 / acc: 0.94 / elapsed time: 1265.06
[2150/6504] cost: 0.09 / acc: 0.96 / elapsed time: 1296.79


config.ini
main.py
model.py
accuracy_list = list()
data_idx = np.arange(len(total_data))
for exp_idx, (train_idx, eval_idx) in enumerate(cv.split(data
separate data
print('separate data {}'.format(exp_idx))
print('load model')
model_dic = {
classifier = model.KISNet(model_num=step,
train_data=train_data,
eval_data=eval_data,
model_dic=model_dic)
classifier.train()
accuracy = classifier.evaluate()
[2000/6939] acc: 0.87 / elapsed time: 146.12
[3000/6939] acc: 0.91 / elapsed time: 205.91
[4000/6939] acc: 0.90 / elapsed time: 262.92
[5000/6939] acc: 0.88 / elapsed time: 320.30
[6000/6939] acc: 0.89 / elapsed time: 376.29
test time : 427.95388102531433
accuracy : 89.17380025940337
-----evaluating finish-----