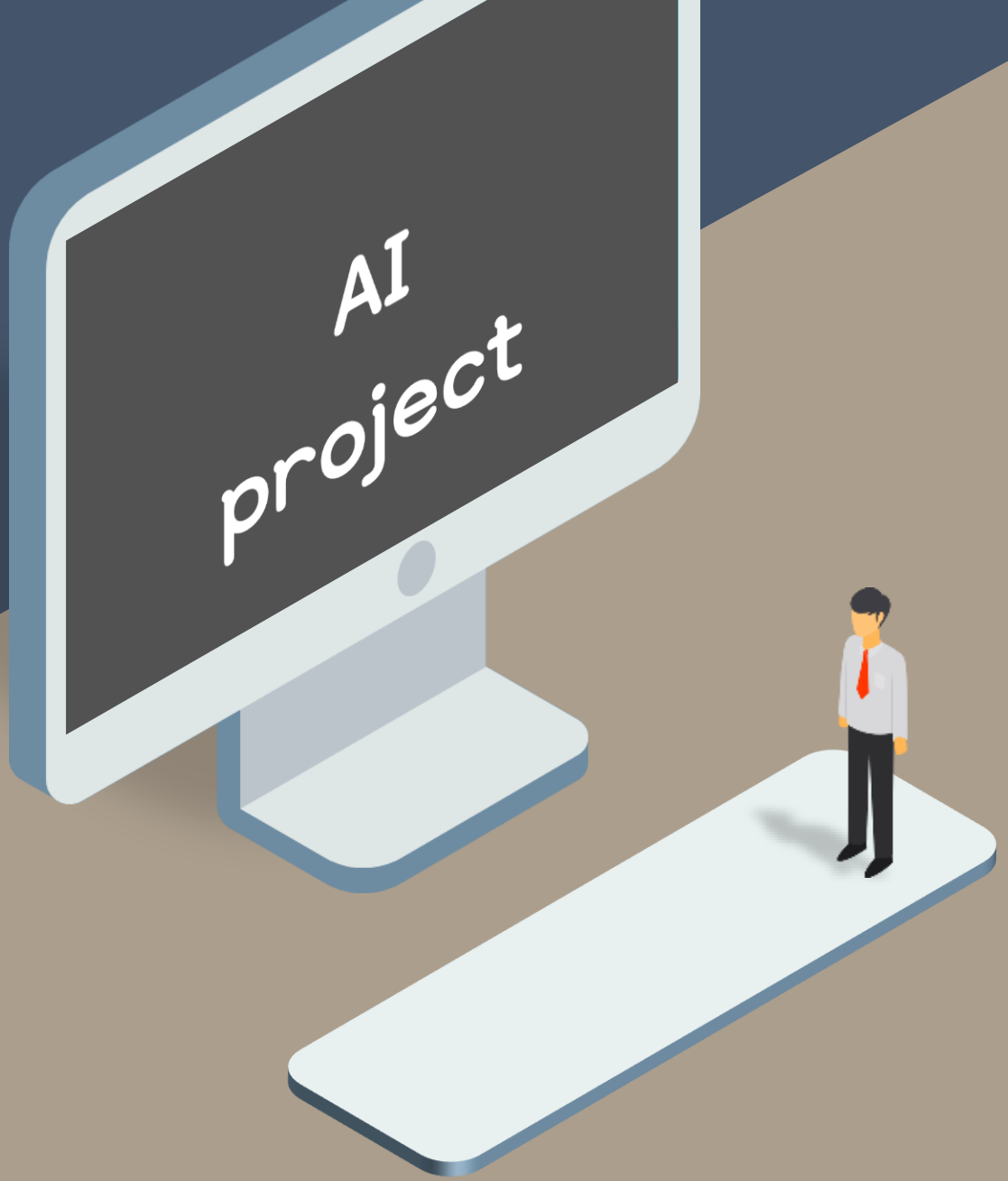


Magenta Project

고하은 장선호 정종운

Magenta Project 개요

음악을 만들거나 그림을 그리게 하는 인공지능 프로젝트

=> AI 기술을 미술과 음악 등 예술 영역에 접목시켜 새로운 예술 작품을 만들려는 시도

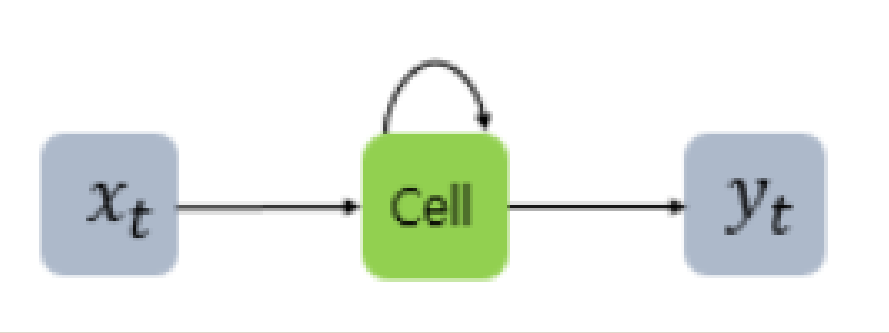
RNN 신경망으로 구현

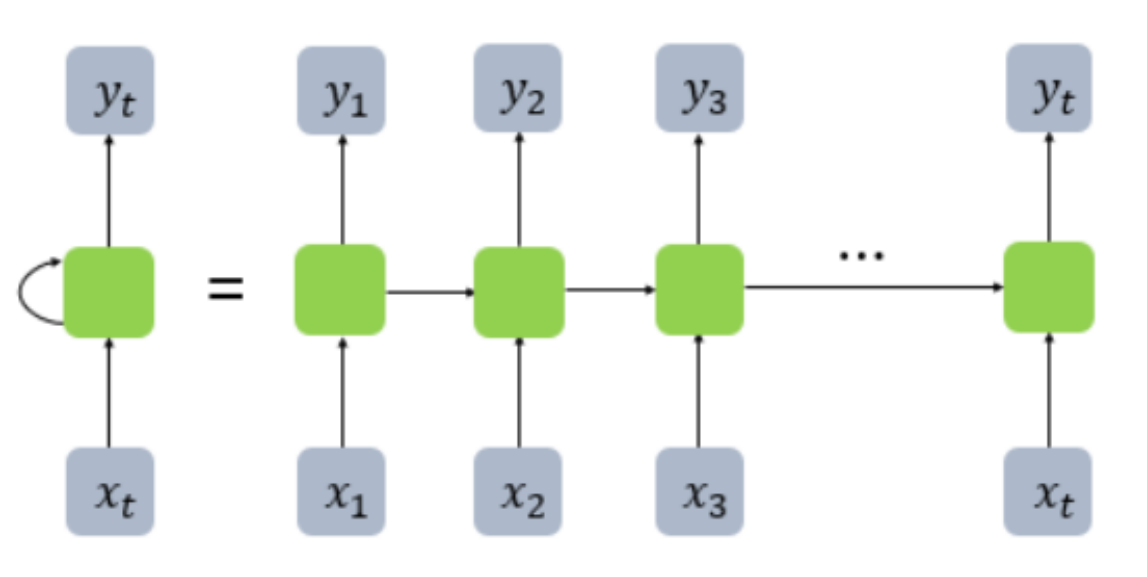
RNN

순환 계층 신경망

입력과 출력을 시퀀스 단위로 처리하는 시퀀스 모델

기본 구조





Magenta RNN layer

Make_rnn_cell 함수 : 입력받은 하이퍼 매개변수로부터 rnn cell을 만듦

```
def make_rnn_cell(rnn_layer_sizes,
                  dropout_keep_prob=1.0,
                  attn_length=0,
                  base_cell=contrib_rnn.BasicLSTMCell,
                  residual_connections=False):
```

Magenta RNN layer

Make_rnn_cell 함수

- 셀 형성
- Attention

```
cells = []
for i in range(len(rnn_layer_sizes)):
  cell = base_cell(rnn_layer_sizes[i])
  if attn_length and not cells:
    # Add attention wrapper to first layer.
    cell = contrib_rnn.AttentionCellWrapper(
        cell, attn_length, state_is_tuple=True)
  if residual_connections:
    cell = contrib_rnn.ResidualWrapper(cell)
    if i == 0 or rnn_layer_sizes[i] != rnn_layer_sizes[i - 1]:
      cell = contrib_rnn.InputProjectionWrapper(cell, rnn_layer_sizes[i])
  cell = contrib_rnn.DropoutWrapper(cell, output_keep_prob=dropout_keep_prob)
  cells.append(cell)

cell = contrib_rnn.MultiRNNCell(cells)

return cell
```

Magenta RNN layer

Make_cdnn 함수 : 주어진 하이퍼 매개 변수를 통해 cuDNN LSTM 층을 조직

```
def make_cudnn(inputs, rnn_layer_sizes, batch_size, mode,
               dropout_keep_prob=1.0, residual_connections=False):
  """Builds a sequence of cuDNN LSTM layers from the given hyperparameters.
```

Magenta RNN layer

Build 함수 : 텐서 플로우 그래프를 만듦

```
def build():
  """Builds the Tensorflow graph."""
  inputs, labels, lengths = None, None, None

  if mode in ('train', 'eval'):
    if isinstance(no_event_label, numbers.Number):
      label_shape = []
    else:
      label_shape = [len(no_event_label)]
    inputs, labels, lengths = magenta.common.get_padded_batch(
        sequence_example_file_paths, hparams.batch_size, input_size,
        label_shape=label_shape, shuffle=mode == 'train')
```

Event sequence RNN model

학습모델 만들기

```
class EventSequenceRnnModel(mm.BaseModel):
  """Class for RNN event sequence generation models.
```

```
def __init__(self, config):
  """Initialize the EventSequenceRnnModel.

  Args:
    config: An EventSequenceRnnConfig containing the encoder/decoder and
      HParams to use.
  """
  super(EventSequenceRnnModel, self).__init__()
  self._config = config

def _build_graph_for_generation(self):
  events_rnn_graph.get_build_graph_fn('generate', self._config)()

def _batch_size(self):
  """Extracts the batch size from the graph."""
  return self._session.graph.get_collection('inputs')[0].shape[0].value
```

Event sequence RNN model

Generate_step : 시행중인 event sequence를 수정함. 수정된 event sequence, 업데이트된 모델의 상태 및 log-likelihoo를 return

```
def _generate_step(self, event_sequences, model_states, logliks, temperature,
                   extend_control_events_callback=None,
                   modify_events_callback=None):
```

Event sequence RNN model

이전의 sequence에서 event sequence를 생성함

```
def _generate_events(self, num_steps, primer_events, temperature=1.0,
                     beam_size=1, branch_factor=1, steps_per_iteration=1,
                     control_events=None, control_state=None,
                     extend_control_events_callback=(
                         _extend_control_events_default),
                     modify_events_callback=None):
```

Train

Run_Training 함수를 통해
Event sequence RNN
Model을 훈련함

훈련 루프 실행

```
with tf.Graph().as_default():
  with tf.device(tf.train.replica_device_setter(num_ps_tasks)):
    build_graph_fn()

    global_step = tf.train.get_or_create_global_step()
    loss = tf.get_collection('loss')[0]
    perplexity = tf.get_collection('metrics/perplexity')[0]
    accuracy = tf.get_collection('metrics/accuracy')[0]
    train_op = tf.get_collection('train_op')[0]

    logging_dict = {
        'Global Step': global_step,
        'Loss': loss,
        'Perplexity': perplexity,
        'Accuracy': accuracy
    }
    hooks = [
        tf.train.NanTensorHook(loss),
        tf.train.LoggingTensorHook(
            logging_dict, every_n_iter=summary_frequency),
        tf.train.StepCounterHook(
            output_dir=train_dir, every_n_steps=summary_frequency)
    ]
```

Train

Contrib_training.train
으로 훈련 진행

Tensorflow.contrib의
Training 모듈의 train 함수

```
if num_training_steps:
  hooks.append(tf.train.StopAtStepHook(num_training_steps))

scaffold = tf.train.Scaffold(
    saver=tf.train.Saver(
        max_to_keep=checkpoints_to_keep,
        keep_checkpoint_every_n_hours=keep_checkpoint_every_n_hours))

tf.logging.info('Starting training loop...')
contrib_training.train(
    train_op=train_op,
    logdir=train_dir,
    scaffold=scaffold,
    hooks=hooks,
    save_checkpoint_secs=save_checkpoint_secs,
    save_summaries_steps=summary_frequency,
    master=master,
    is_chief=task == 0)
tf.logging.info('Training complete.')
```

References

https://github.com/tensorflow/magenta/blob/master/magenta/models/shared/events_rnn_train.py
- 마젠타 프로젝트 깃헙

https://wikidocs.net/22886 - 순환 신경망 위키독스

https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/10/06/attention/ - 어텐션 매커니즘

Thank you