KU KONKUK UNIVERSITY

Transfer Learning

자연어처리연구실

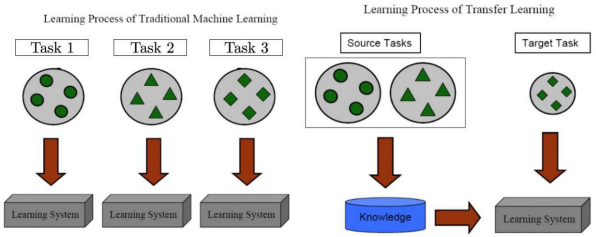
10 Trends in Deep Learning NLP

1. Previous word embedding approaches are still important
2. Recurrent Neural Networks (RNNs) are no longer an NLP standard architecture
3. The Transformer will become the dominant NLP deep learning architecture
4. Pre-trained models will develop more general linguistic skills
5. Transfer learning will play more of a role
6. Fine-tuning models will get easier
7. BERT will transform the NLP application landscape
8. Chatbots will benefit most from this phase on NLP innovation
9. Zero shot learning will become more effective
10. Discussion about the dangers of AI could start to impact NLP research and applications

출처: https://blog.floydhub.com/ten-trends-in-deep-learning-nlp/

전이 학습 (Transfer Learning)

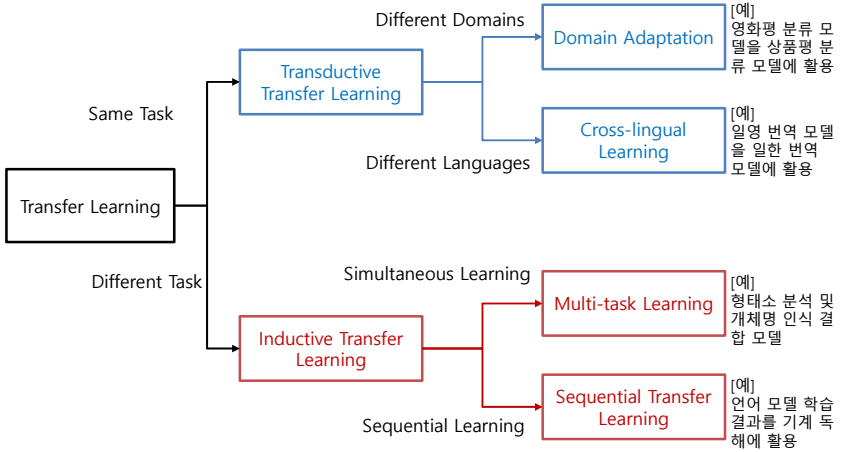
- 전이 학습
  - 사전 작업(source task)에 대하여 학습된 정보를 목표 작업(target task)에 활용하는 방법
  - 학습 데이터가 부족해도 목표 작업에 대한 수렴 속도 및 성능 향상을 꾀할 수 있음



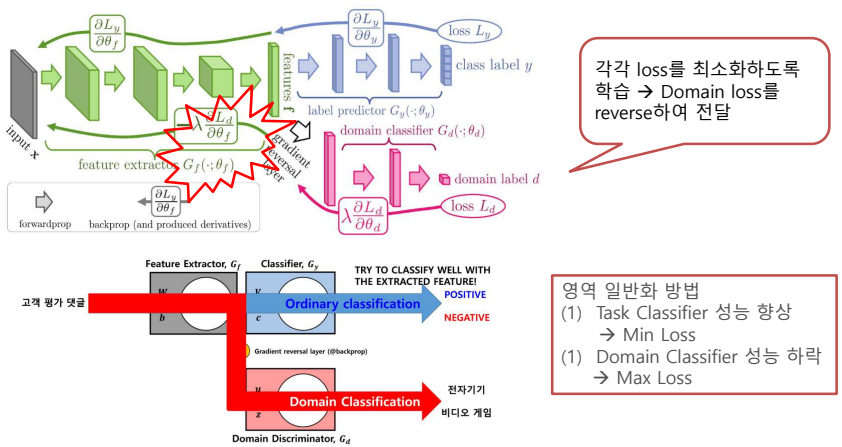
(a) Traditional Machine Learning (b) Transfer Learning

그림 출처: Pan and Yang 발표자료

전이 학습 유형



Transductive TL: Domain Adaptation

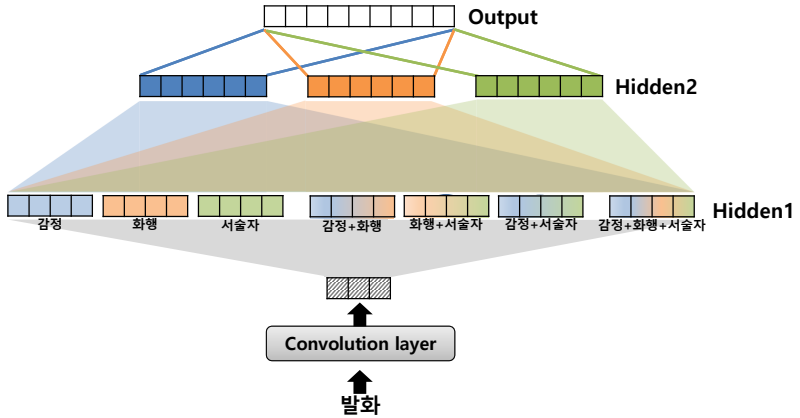


각각 loss를 최소화하도록 학습 → Domain loss를 reverse하여 전달

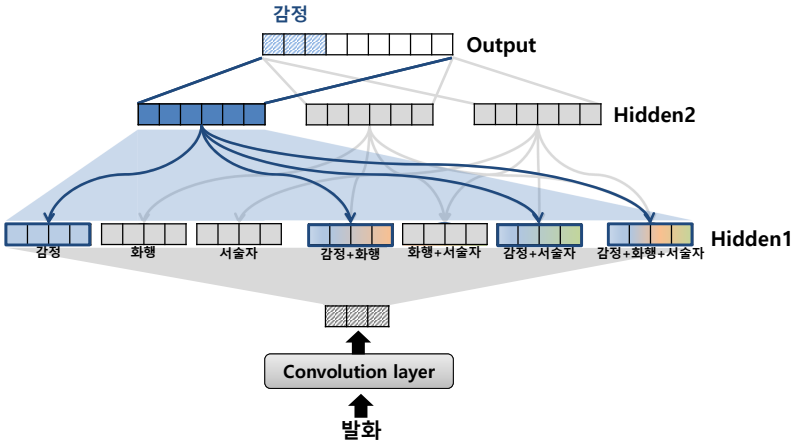
영역 일반화 방법
(1) Task Classifier 성능 향상
→ Min Loss
(1) Domain Classifier 성능 하락
→ Max Loss

참조: Ganin & Lempitsky, "Unsupervised Domain Adaptation by Backpropagation"

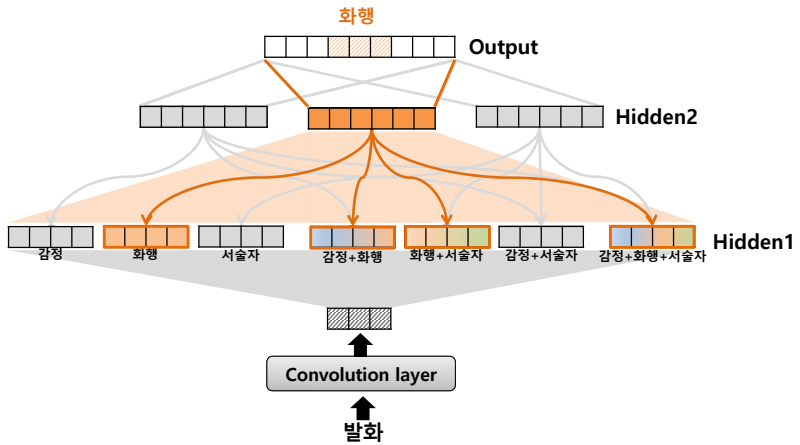
Inductive TL: Multi-task Learning



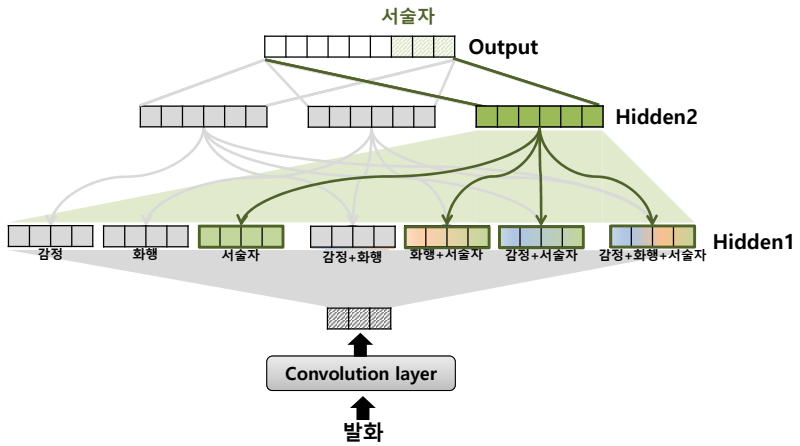
Inductive TL: Multi-task Learning



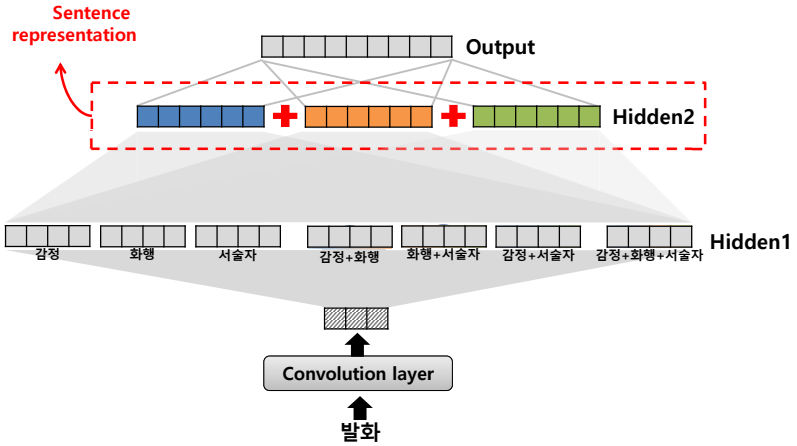
Inductive TL: Multi-task Learning



Inductive TL: Multi-task Learning



Inductive TL: Multi-task Learning



Inductive TL: Sequential Transfer Learning

- 순차 전이 학습
  - 사전 작업과 목표 작업이 다르고 각 작업에 대하여 순차적으로 학습을 수행하는 전이 학습 방법
    - 사전 작업과 목표 작업에 대한 데이터를 동시에 사용할 수 없는 경우
    - 사전 작업 학습 데이터가 목표 작업 학습 데이터보다 많은 경우
    - 여러 목표 작업에 대한 적용이 필요한 경우
- 대용량 사전 학습 언어 모델
  - Large-scaled pre-trained language model
    - BERT, GPT, ALBERT, RoBERTa, ELECTRA, XLNet 등
  - 다양한 NLP 문제에서 SOTA (State-Of-The-Art) 성능 획득

OpenAI GPT vs. Google BERT

Pre-training a Transformer Decoder for Language Modeling

Pre-training a Transformer Encoder for Language Modeling

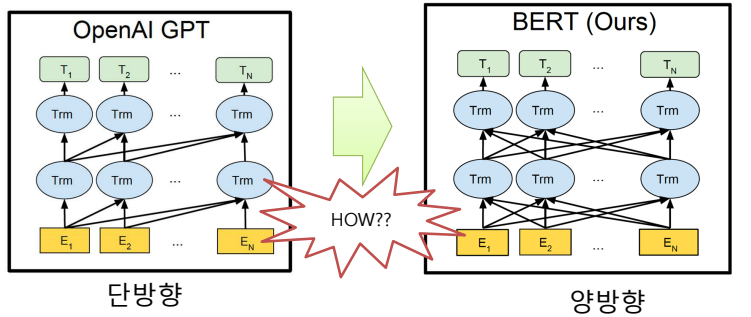


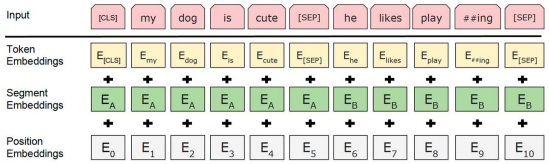
그림 출처: Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

BERT

- BERT Models
  - BERT-Base
    - L=12, A=12, H=768
    - Total Parameters=110M
  - BERT-Large
    - L=24, A=16, H=1024
    - Total Parameters=340M

L : # of layers
A : # of heads
H : Hidden size of FFN

- Input Representation



Pre-Training

- 학습 방법의 차별화

How to learn LM?
→ Word2Vec의 CBOW처럼

- Masked LM
  - 문장에 존재하는 단어를 마스킹 후 예측하게 함

- Next Sentence Prediction
  - 다음 문장 여부를 학습

BERT 목적 → Transfer Learning
(QA, NLI 등)

Masked LM

- 무작위 토큰 중 15%만 아래의 내용 적용
  - 80% of the time: Replace the word with the [MASK] token
    - my dog is hairy -> my dog is [MASK]
  - 10% of the time: Replace the word with a random word
    - my dog is hairy -> My dog is apple
  - 10% of the time: Keep the word unchanged,
    - my dog is hairy -> my dog is hairy

Next Sentence Prediction

- 주어진 문장이 다음 문장 관계인지 예측
  - 50%는 연속된 문장 학습
  - 50%는 랜덤으로 다른 문장 가져와서 학습

$\text{Input} =$ `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
$\text{Label} =$ `IsNext`

$\text{Input} =$ `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
$\text{Label} =$ `NotNext`

Datasets and Experimental Settings

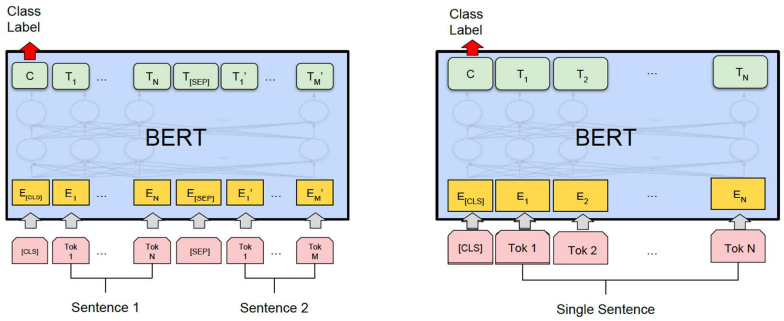
- Datasets
  - Book Corpus (800M word)
  - English Wikipedia (2,500M word)
- Training
  - BERT-Base: Cloud TPU 16개로 4일
  - BERT-Large: Cloud TPU 64개로 4일

GPU(Titan X) → 375일

Experiments on Down-Stream Tasks

- Fine-tuning 방법으로 학습
- Classification task
- Span prediction task
- Sequence labeling task

Classification Task



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

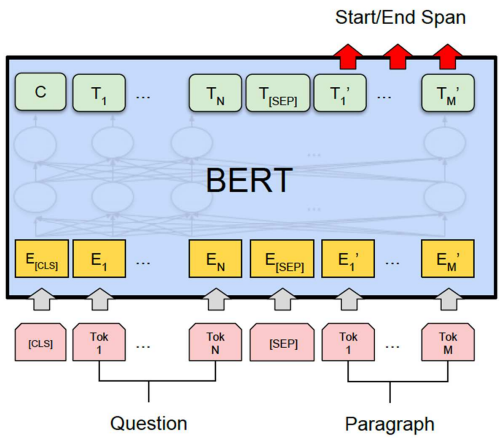
(b) Single Sentence Classification Tasks: SST-2, CoLA

Classification Performance

- 문장/문서 분류

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

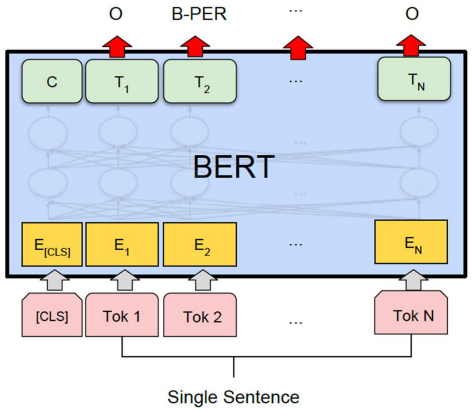
Span Prediction Task



SQuAD v1.1 Performance

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Sequence Labeling Task



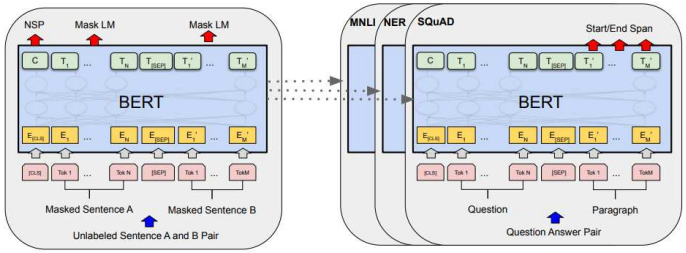
NER Performance

- 개체명 인식

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Self-Supervised Pre-training

- 사전 작업으로부터 저수준의 자질을 학습하고 이를 활용할 수 있는 여러 목표 작업에 학습과 성능 향상에 영향을 줄 수 있음
- 목표 작업에 대하여 적은 시간과 자원을 사용하여 비교적 높은 성능을 보일 수 있음



Pre-trained LM: BERT → ALBERT, RoBERTa → XLNet, ELECTRA, ...

실습

실습 코드 다운로드: https://github.com/KUNLP/Lecture

- 대용량 언어 모델인 ELECTRA를 이용하여 기계 독해(MRC; Machine Reading Comprehension) 시스템을 구현 하시오.

질의와 문서를 입력으로 받아 질의에 대한 답변의 위치를 문서에서 찾아주는 인공지능 모델

대한민국 서쪽에는 어느 나라가 있나?

대한민국(大韓民國, 영어: Republic of Korea; ROK, 문화어: 남조선; 南朝鮮), 약칭으로 한국(韓國), 남한(南韓)은 동아시아의 한반도 남부에 있는 공화국이다. 서쪽으로는 서해를 사이에 두고 중화인민공화국이, 동쪽으로는 동해를 사이에 두고 일본이 있으며 북쪽으로는 조선민주주의인민공화국과 맞닿아 있다. 수도는 서울특별시이며, 국기는 태극기, 국가는 애국가, 공용어는 한국어이다.

데이터 구성

질의, 문서, 정답 구분자

정답 위치 (시작, 끝)

mrc_train.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

[CLS] 서 주 에 는 _ 무 엇 이 _ 암 시 되 어 _ 있 는 가 ? [SEP] 이 _ 작 품 은 _ 라 ... 90 95

[CLS] 첫 부 분 에 는 _ 어 떤 _ 악 기 를 _ 사 용 해 _ 더 욱 _ 명 확 하 게 _ 나 타 ... 181 186

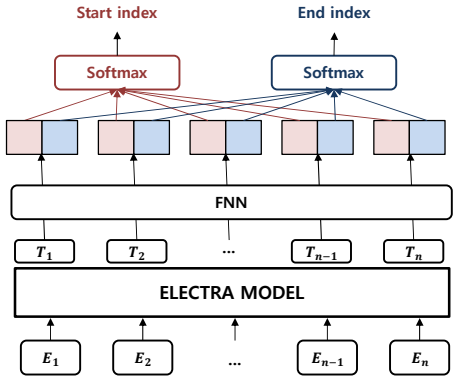
[CLS] 주 요 부 는 _ 어 떤 _ 형 식 으 로 _ 되 어 _ 있 는 가 ? [SEP] 이 _ 작 품 은 조 용 히 _ 마 친 다 . [SEP] 294 299

언어모델 시작

음절(공백은 _로 변환)

실습

모델 구조



실습

라이브러리 설치

```
!pip install tokenizers
!pip install transformers==2.11.0
```

```
Collecting tokenizers
  Downloading https://files.pythonhosted.org/packages
     |████████████████████████████████| 3.3MB 8.7MB/s
Installing collected packages: tokenizers
Successfully installed tokenizers-0.10.3
Collecting transformers==2.11.0
  Downloading https://files.pythonhosted.org/packages
     |████████████████████████████████| 675kB 8.7MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/loc
```

라이브러리 import

```
import sys
sys.path.append("/gdrive/MyDrive/colab/transfer_learn/mrc")
from tokenization_kocharelectra import KoCharElectraTokenizer
```

```
from transformers.configuration_electra import ElectraConfig
from transformers.modeling_electra import ElectraPreTrainedModel, ElectraModel
from transformers.optimization import AdamW

from tokenization_kocharelectra import KoCharElectraTokenizer
```

실습

모델 설계

```python
class ElectraMRC(ElectraPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)

        # 분류할 라벨의 개수
        self.num_labels = config.num_labels

        # ELECTRA 모델
        self.electra = ElectraModel(config)

        # Span 범위 예측을 위한 linear
        self.projection_layer = nn.Linear(config.hidden_size, self.num_labels)

    def forward(self, input_ids=None, attention_mask=None, token_type_ids=None):
        # input_ids, attention_mask, token_type_ids 형태: [batch, seq_len]
        # electra_outputs 형태: [1, batch, seq_len, hidden]
        electra_outputs = self.electra(input_ids, attention_mask, token_type_ids)

        # hypothesis 형태: [batch, seq_len, 2 (start/end)]
        hypothesis = self.projection_layer(electra_outputs[0])

        # start, end 형태: [batch, seq_len, 1] -> [batch, seq_len]
        p_start, p_end = hypothesis.split(1, dim=-1)
        p_start = p_start.squeeze(-1)
        p_end = p_end.squeeze(-1)

        return p_start, p_end
```

Edited by Harksoo Kim

실습

데이터 읽기

```python
def convert_data2feature(config, input_sequence, tokenizer):
    # 고정 길이 벡터 생성
    input_ids = np.zeros(config["max_length"], dtype=np.int)
    attention_mask = np.zeros(config["max_length"], dtype=np.int)
    segment_ids = np.zeros(config["max_length"], dtype=np.int)

    is_context = False
    for idx, token in enumerate(input_sequence.split()):
        input_ids[idx] = tokenizer._convert_token_to_id(token)
        attention_mask[idx] = 1
    return input_ids, attention_mask, segment_ids
```

어텐션을 계산할 부분만 1
(질의와 문서)

[SEP] 이후만 1
(질의와 문서 구분, 문서 부분만 1)

주요 부분
일부 캡쳐

```python
def read_data(file_path, tokenizer):
    all_input_ids, all_attention_mask, all_segment_ids, start_indexes, end_indexes = [], [], [], [], []
    for idx, line in enumerate(lines):
        input_sequence, start_idx, end_idx = line.strip().split("₩t")
        input_ids, attention_mask, segment_ids = convert_data2feature(config, input_sequence, tokenizer)

    all_input_ids = torch.tensor(all_input_ids, dtype=torch.long)
    all_attention_mask = torch.tensor(all_attention_mask, dtype=torch.long)
    all_segment_ids = torch.tensor(all_segment_ids, dtype=torch.long)
    start_indexes = torch.tensor(start_indexes, dtype=torch.long)
    end_indexes = torch.tensor(end_indexes, dtype=torch.long)

    return all_input_ids, all_attention_mask, all_segment_ids, start_indexes, end_indexes
```


Edited by Harksoo Kim

실습

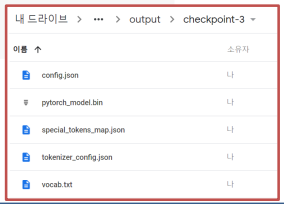
Test

```python
def test(config):
    # electra config 객체 생성
    electra_config = ElectraConfig.from_pretrained(
        os.path.join(config["output_dir"], "checkpoint-{0:d}".format(config["checkpoint"])),
        num_labels=config["num_labels"])

    # electra tokenizer 객체 생성
    electra_tokenizer = KoCharElectraTokenizer.from_pretrained(
        os.path.join(config["output_dir"], "checkpoint-{0:d}".format(config["checkpoint"])),
        do_lower_case=False)

    # electra model 객체 생성
    model = ElectraMRC.from_pretrained(
        os.path.join(config["output_dir"], "checkpoint-{0:d}".format(config["checkpoint"])),
        config=electra_config).cuda()

    do_test(model=model, tokenizer=electra_tokenizer)
```



Edited by Harksoo Kim

실습

Test

```python
def do_test(model, tokenizer):
    # 평가 모드 셋팅
    model.eval()

    # 평가 데이터 Load
    all_input_ids, all_attention_mask, all_segment_ids, start_indexes, end_indexes = ₩
        read_data(tokenizer=tokenizer, file_path=config["test_data_path"])

    for step, batch in enumerate(test_dataloader):
        batch = tuple(t.cuda() for t in batch)
        input_ids, attention_mask, segment_ids, a_start, a_end = batch

        # 입력 데이터에 대한 출력과 loss 생성
        # p_start, p_end 형태:[1, seq_len]
        p_start, p_end = model(input_ids, attention_mask, segment_ids)

        p_start = p_start.argmax(dim=-1)
        p_start = tensor2list(p_start)[0]
        p_end = p_end.argmax(dim=-1)
        p_end_ = tensor2list(p_end)[0]

        a_start = tensor2list(a_start)[0]
        a_end = tensor2list(a_end)[0]

        # 입력 Text 생성
        input_token_ids = tensor2list(input_ids)[0]
        input_tokens = [tokenizer._convert_id_to_token(e) for e in input_token_ids]

        # 입력 Text에서 예측/정답 Span 추출
        predict_span = input_tokens[p_start:p_end+1]
        answer_span = input_tokens[a_start:a_end+1]
```


Edited by Harksoo Kim

실습

Train

```
def train(config):
    # electra config 객체 생성
    electra_config = ElectraConfig.from_pretrained("monologg/kocharelectra-base-discriminator",
                                                   num_labels=config["num_labels"])
    # electra tokenizer 객체 생성
    electra_tokenizer = KoCharElectraTokenizer.from_pretrained("monologg/kocharelectra-base-discriminator",
                                                               do_lower_case=False)
    # electra model 객체 생성
    model = ElectraMRC.from_pretrained("monologg/kocharelectra-base-discriminator",
                                                                    config=electra_config).cuda()
```

```
    for epoch in range(config["epoch"]):
        for step, batch in enumerate(train_dataloader):
            # 학습 모드 셋팅
            model.train()
```

```
            # p_start, p_end 형식: [batch, seq_len]
            # a_start, a_end 형식: [batch]
            input_ids, attention_mask, segment_ids, a_start, a_end = batch
            p_start, p_end = model(input_ids, attention_mask, segment_ids)

            start_loss = loss_func(p_start, a_start)
            end_loss = loss_func(p_end, a_end)

            total_loss = start_loss + end_loss

            # 손실 역전파 수행
            total_loss.backward()
            optimizer.step()
```

Edited by Harksoo Kim

실습

Main

```
if __name__ == "__main__":
    root_dir = "/gdrive/MyDrive/colab/transfer_learn/mrc"
    output_dir = os.path.join(root_dir, "output")

    if (not os.path.exists(output_dir)):
        os.makedirs(output_dir)

    config = {"mode": "test",
              "train_data_path": os.path.join(root_dir, "mrc_train.txt"),
              "test_data_path": os.path.join(root_dir, "mrc_dev.txt"),
              "output_dir": output_dir,
              "checkpoint": 3,
              "epoch": 3,
              "learning_rate": 5e-5,
              "batch_size": 16,
              "max_length": 512,
              "num_labels": 2,
              }

    if (config["mode"] == "train"):
        train(config)
    else:
        test(config)
```

학습에 5시간 이상 소요

```
########################################
Context :  2008년 2월 25일 이명박은 취임식과 함께 업무 수행을 시작
Question :  이명박이 취임식과 함께 업무수행을 시작한 해는?
Answer Span :  2008년
Predict Span :  2008년


########################################
Context :  부모는 사건 이후 일을 그만두고 딸의 치료에만 매달렸다.
Question :  조두순 사건에서 보험사는 부모에게 얼마를 지급했는가?
Answer Span :  4000만 원
Predict Span :  4000만 원
```

Edited by Harksoo Kim

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr

Edited by Harksoo Kim