

KoGPT2를 이용한 쇼핑몰 리뷰 생성기

20190431 박규현

목차


1 기법 제안

2 관련 모델 설명

3 결과

01 프로젝트 소개

Input:

옷 상태 

키워드: 소재

배송 ☒ 느려요 ☐ 빨라요

사이즈 작아요 ☒ 정사이즈 ☐ 커요

Output:

부들부들한 소재라서 괜찮았고 사이즈도 딱 맞았어요!
근데 배송이 너무 느려요 ㅠㅠ

쇼핑몰 리뷰 자동 생성기 프로젝트

KoGPT2 모델을 이용한 프로그램 제작 목표

각 카테고리별로 입력을 받게 되면 학습된 모델에
적용하여 리뷰를 생성하는 프로그램

01 기법 제안

1. DataSet 구축

- 유명 쇼핑몰(커먼유니크, 핫핑크, 브랜드) 등 쇼핑몰 웹 페이지에서 리뷰 크롤링을 통한 데이터 셋 구성

| | |
|---|----|
| 루즈핏 난방을 원했는데 일단 제품은 맘에 들어요 근데 배송이 일주일 넘게 걸렸어요 배송완료라 쓰는데 택배는 안오고,,,,,뭐 그래도 뭣은 괜찮아요 | 4점 |
| 예뻐요 저렴하게 구매해서 좋습니다 다음에 또 재구매하고싶어요 ㅎㅎ 색감도 예뻐요 | 5점 |

2. 옷 리뷰 긍/부정 분류기 구축

- BERT를 이용한 리뷰 감정 분석 이용
- 긍 / 부정 확률을 이용하여 별점 부여

| | |
|--------------------------------------|---|
| 루즈핏 난방을 원했는데 일단 제품은 맘에 들어요 | 1 |
| 근데 배송이 일주일 넘게 걸렸어요 배송완료라 쓰는데 택배는 안오고 | 0 |
| 뭐 그래도 뭣은 괜찮아요 | 1 |

3. 리뷰 생성기를 위한 학습 DataSet 구축

- 3개의 카테고리(옷 상태, 배송, 사이즈)로 나누어서 각각 학습

4. 리뷰 생성 기법을 이용한 KoGPT2 사용

- 쇼핑몰 리뷰 생성을 위한 모델 파인튜닝 진행
- 별점과 키워드를 생각한 controlling이 가능한 리뷰 생성기 구축

5. 학습

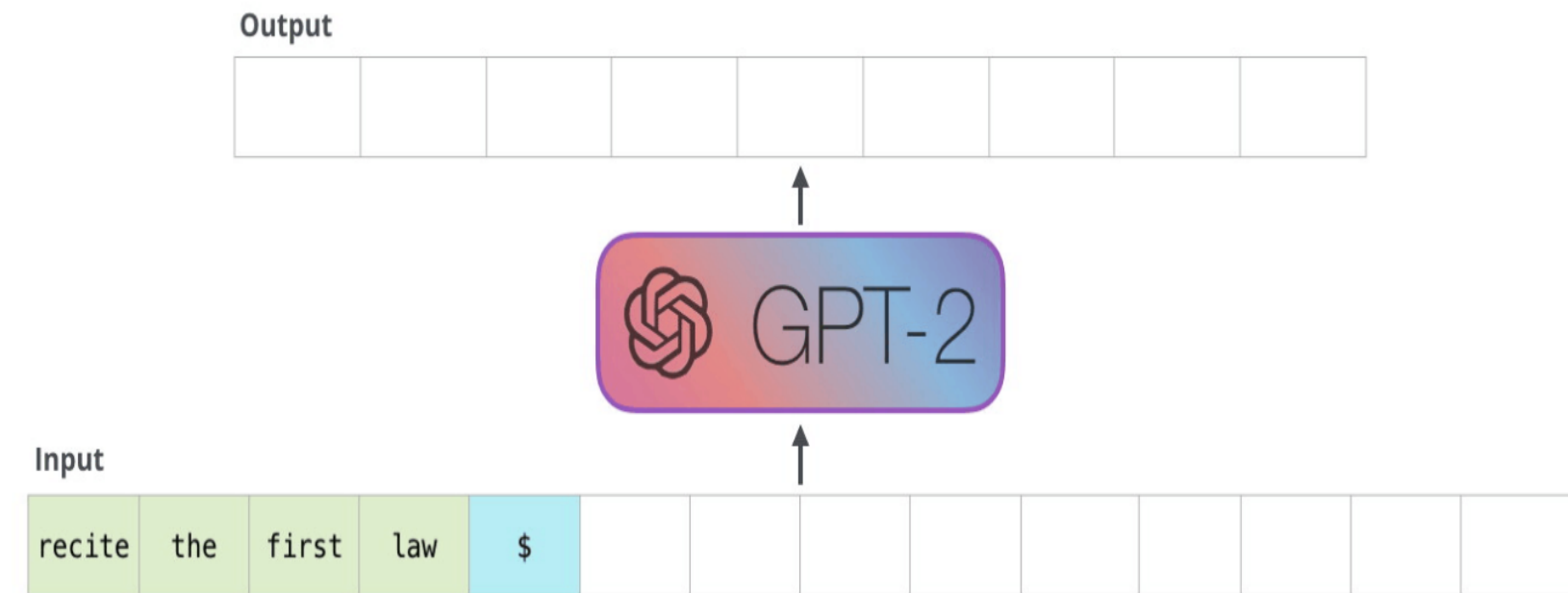
- 3개의 카테고리 별로 각각 따로 학습 진행
- 기준(별점) 별로 따로 학습 진행

6. 결과 확인 & 성능 평가

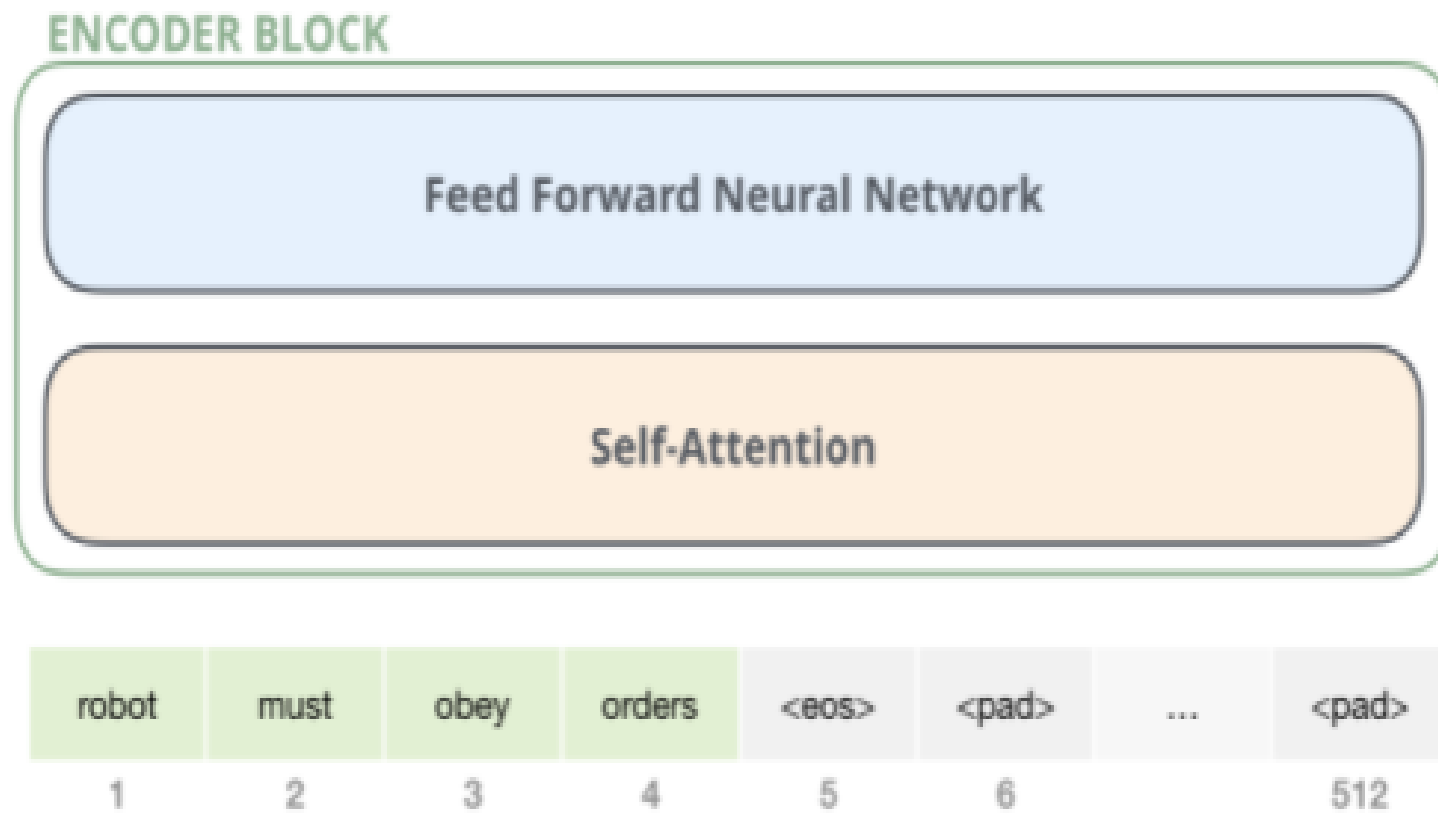
- Human Evaluation

02 --- KoGPT2

1. GPT-2는 주어진 텍스트의 다음 단어를 잘 예측할 수 있도록 학습된 언어모델
2. 어텐션만으로 구성된 트랜스포머(Transformer)를 기반으로 디코더 스택만 사용한 모델

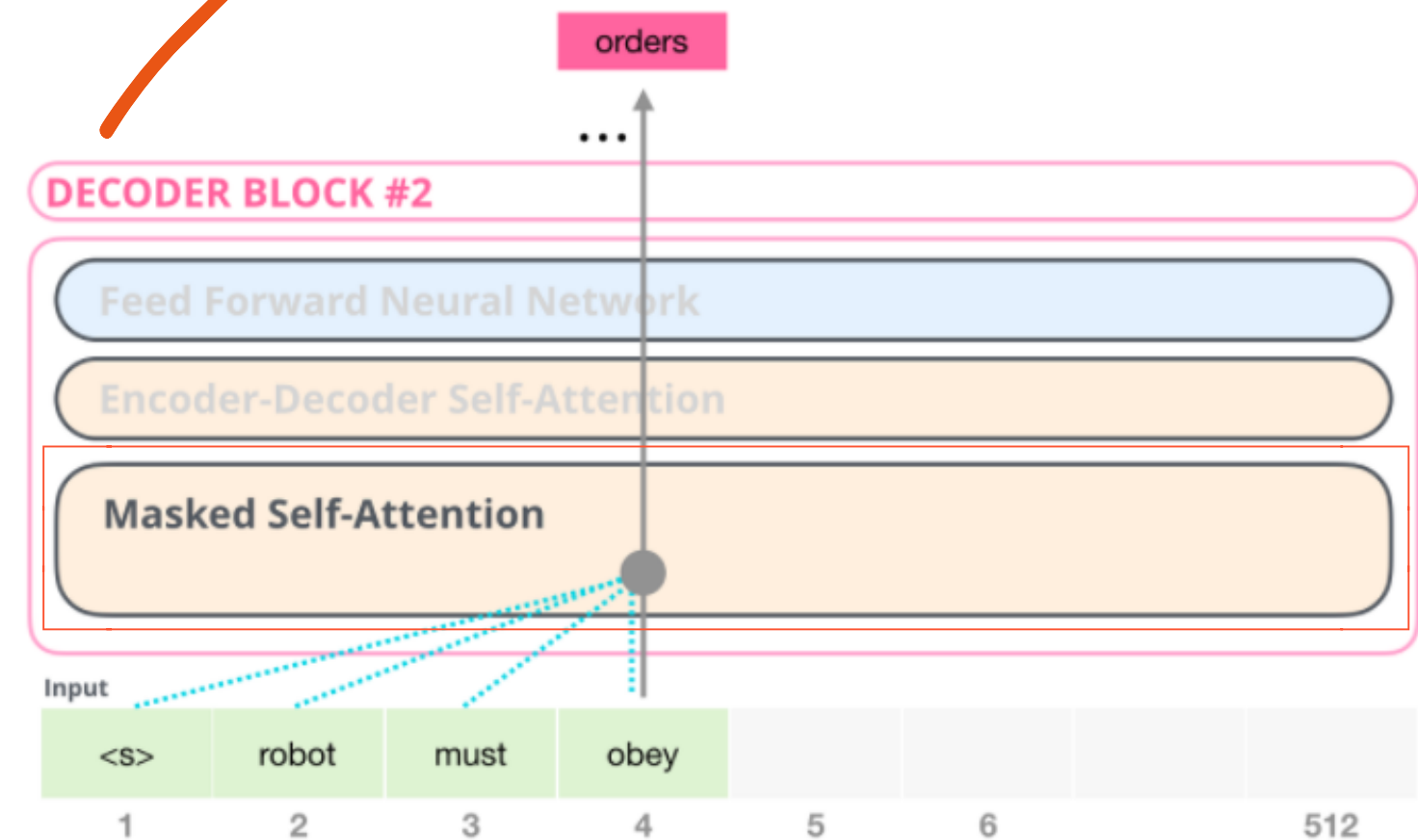


02 KoGPT2



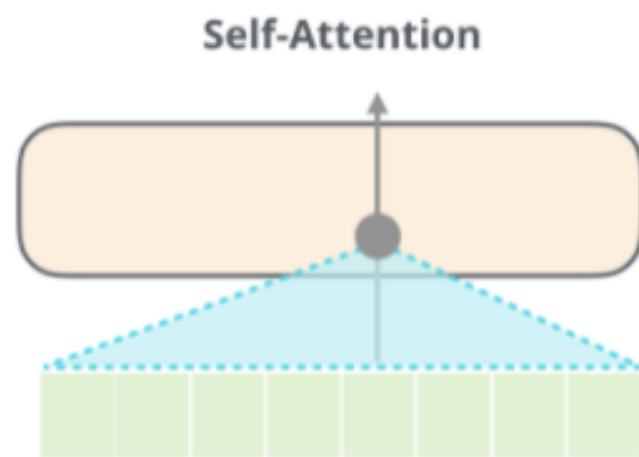
✓ Encoder

셀프 어텐션을 계산할 때 해당 스텝의 오른쪽에 있는
단어들은 고려하지 않는다는 것을 의미



✓ Decoder

02 KoGPT2



타깃 단어 뒤에 위치한 단어는
Self-Attention에 영향을 주지
않도록 마스킹 처리

Scores
(before softmax)

| | | | |
|------|------|------|------|
| 0.11 | 0.00 | 0.81 | 0.79 |
| 0.19 | 0.50 | 0.30 | 0.48 |
| 0.53 | 0.98 | 0.95 | 0.14 |
| 0.81 | 0.86 | 0.38 | 0.90 |

Apply Attention
Mask

Masked Scores
(before softmax)

| | | | |
|------|------|------|------|
| 0.11 | -inf | -inf | -inf |
| 0.19 | 0.50 | -inf | -inf |
| 0.53 | 0.98 | 0.95 | -inf |
| 0.81 | 0.86 | 0.38 | 0.90 |

-무한대의 값에 해당하는
수를 더해준다

02 KoGPT2

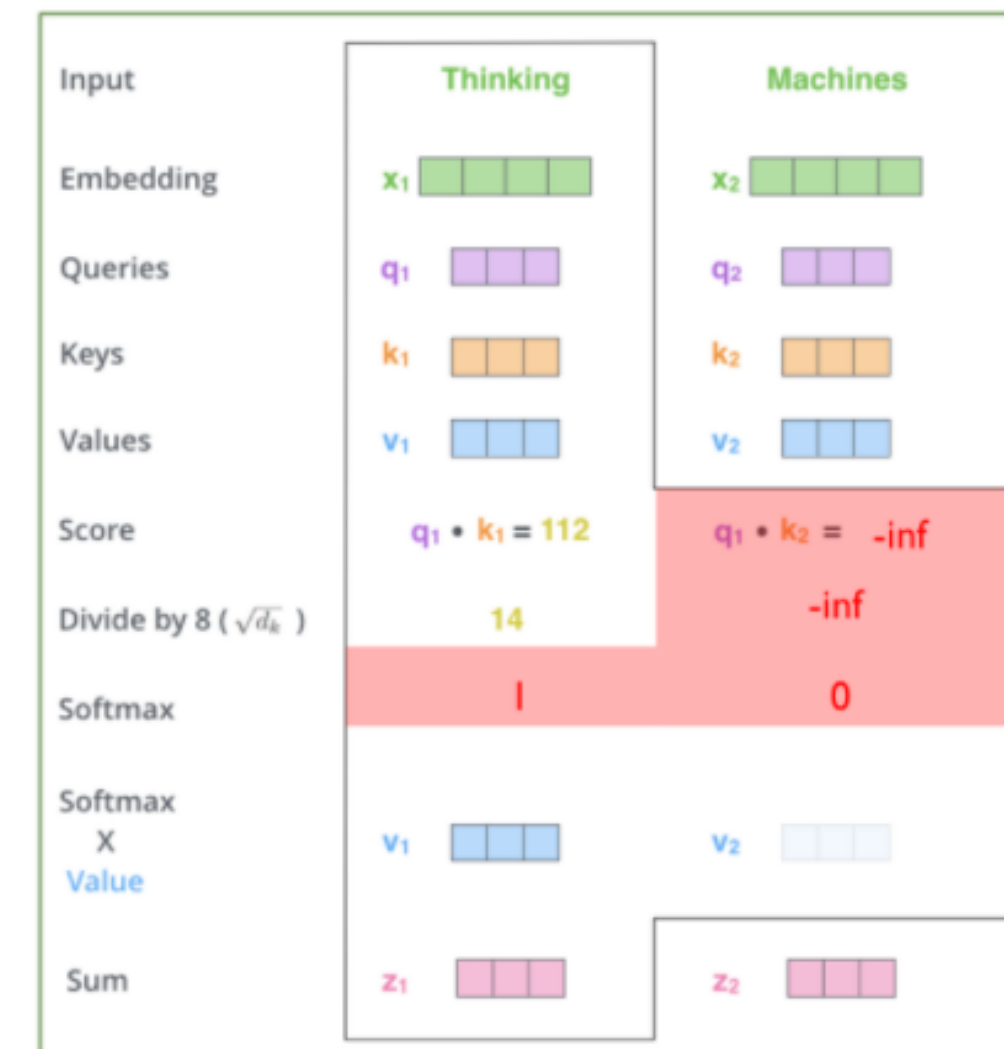
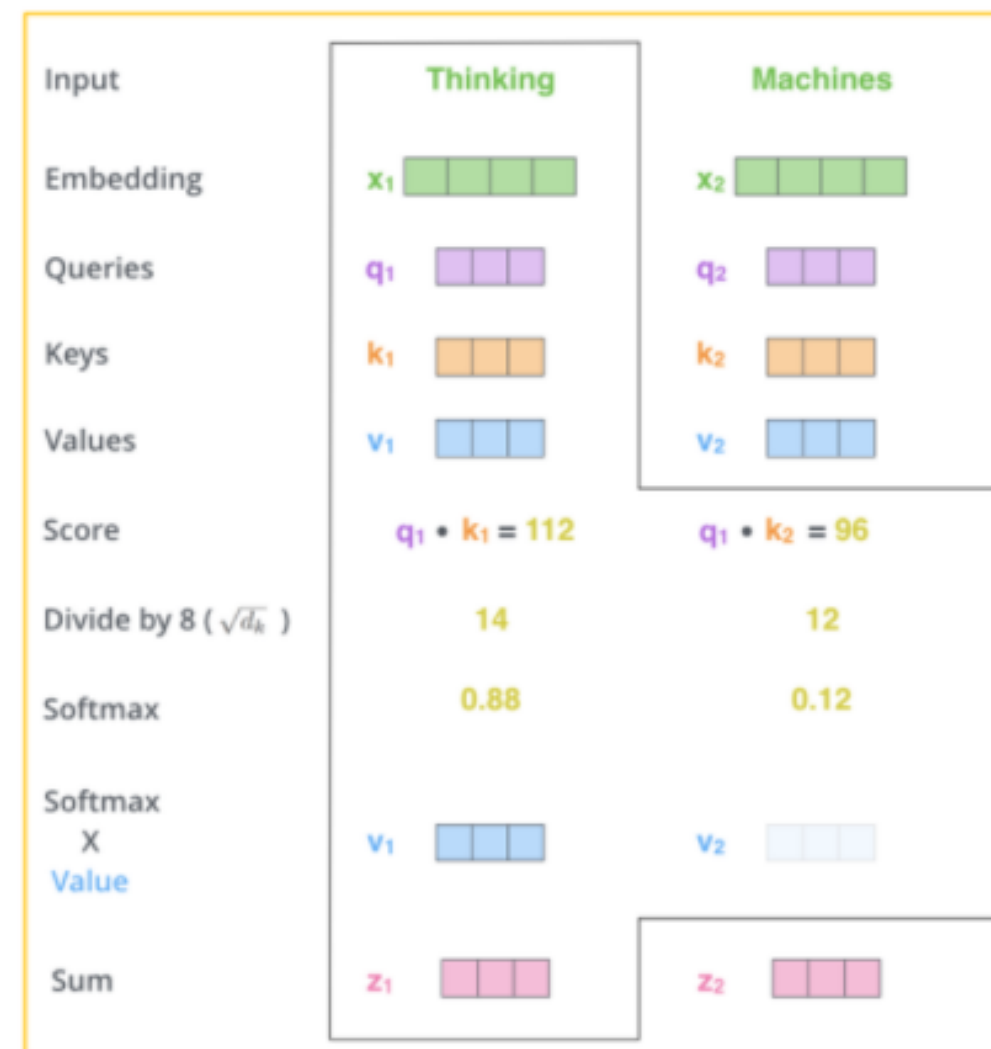
Masked Scores
(before softmax)

| | | | |
|------|------|------|------|
| 0.11 | -inf | -inf | -inf |
| 0.19 | 0.50 | -inf | -inf |
| 0.53 | 0.98 | 0.95 | -inf |
| 0.81 | 0.86 | 0.38 | 0.90 |

Softmax
(along rows)

Scores

| | | | |
|------|------|------|------|
| 1 | 0 | 0 | 0 |
| 0.48 | 0.52 | 0 | 0 |
| 0.31 | 0.35 | 0.34 | 0 |
| 0.25 | 0.26 | 0.23 | 0.26 |

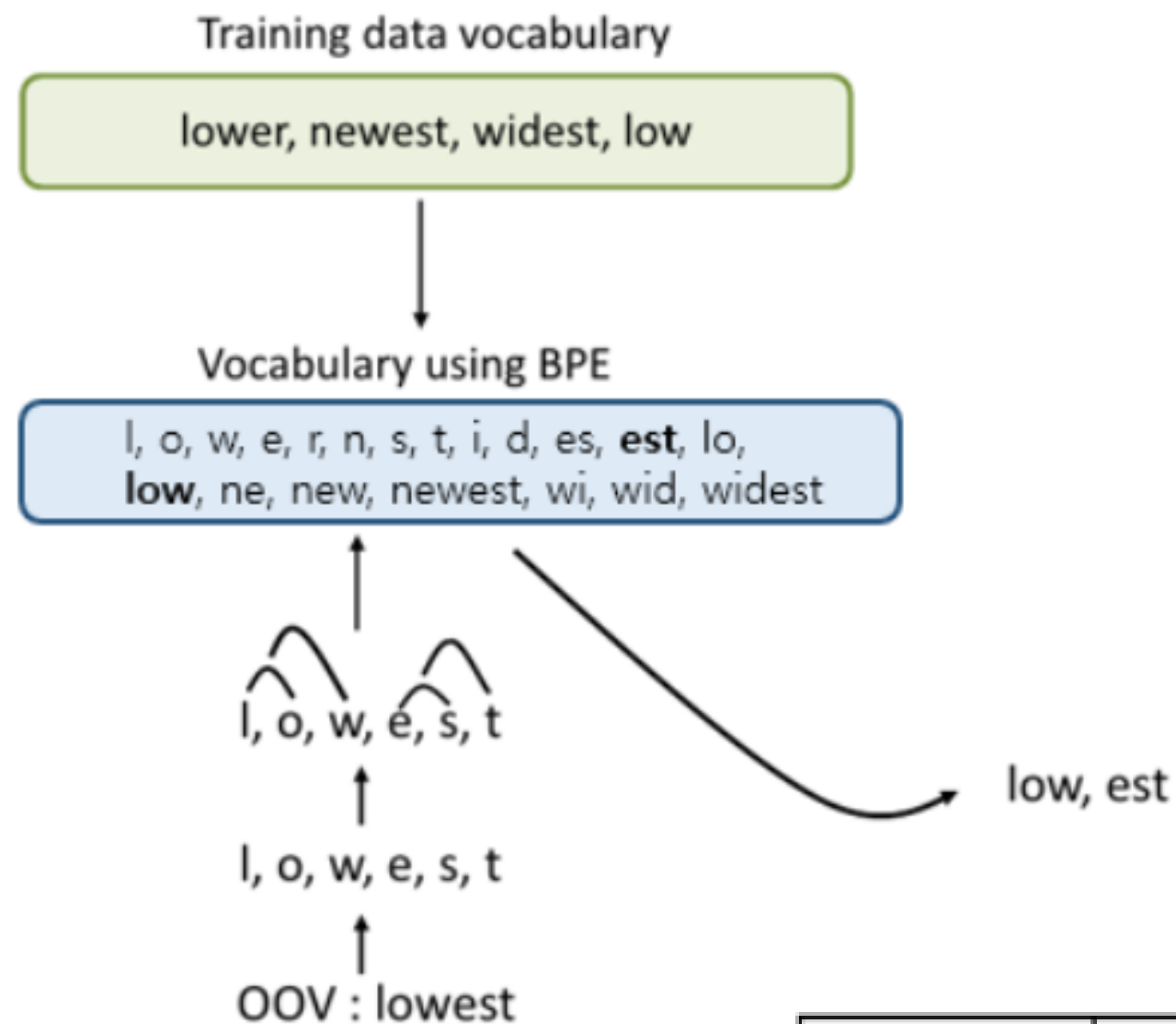


다음 단어의 예측 능력이 뛰어남 !

02 KoGPT2

1. BPE(Byte Pair Encoding) 사용

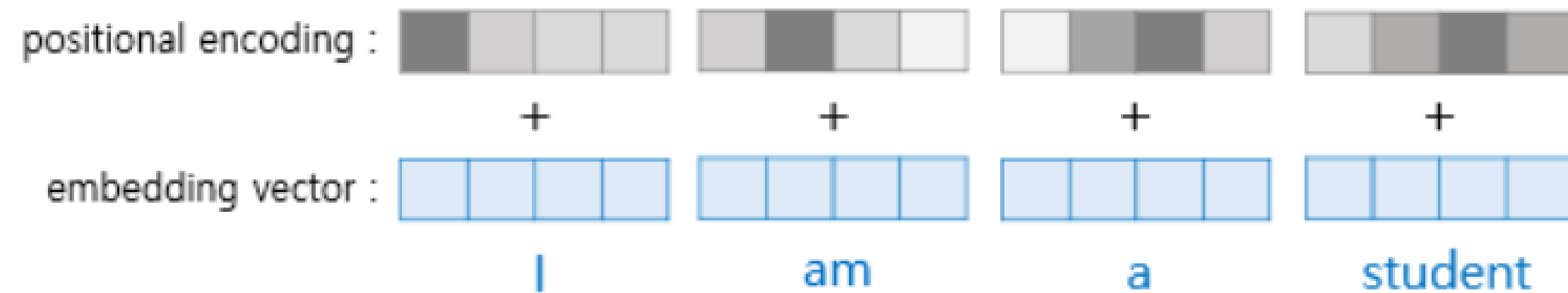
- BPE는 서브워드를 분리하는 알고리즘
- 가장 빈도수가 높은 쌍을 하나로 통합하는 과정을 반복하여
토큰 디셔너리 생성
- GPT2는 Byte Pair Encoding를 거친 토큰을 입력 단위로
사용을 하지만 바로 입력으로 넣는 것이 아니라, Positional
Encoding을 추가적으로 거쳐야함



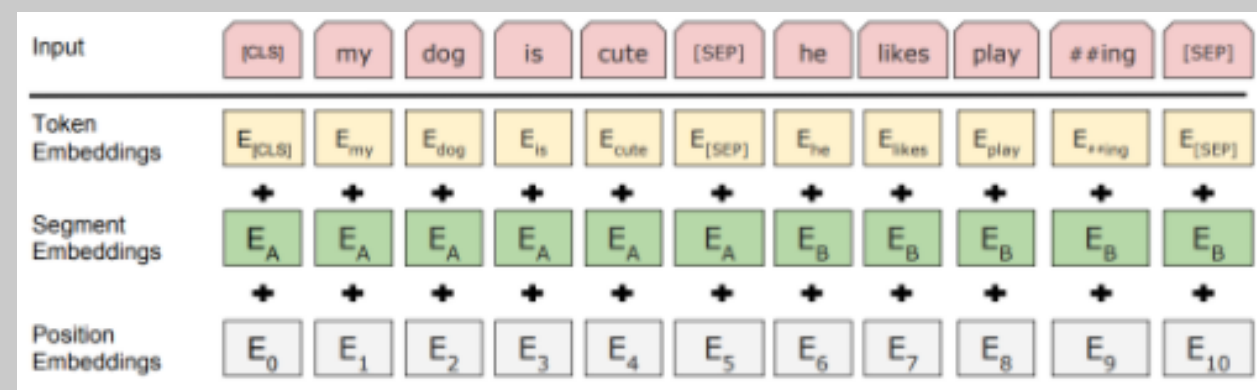
| Input | Output |
|-------|------------|
| 찰떡 | 옷이 찰찰떡 맞아요 |

02

KoGPT2



BERT는 Position Embeddings 사용 !

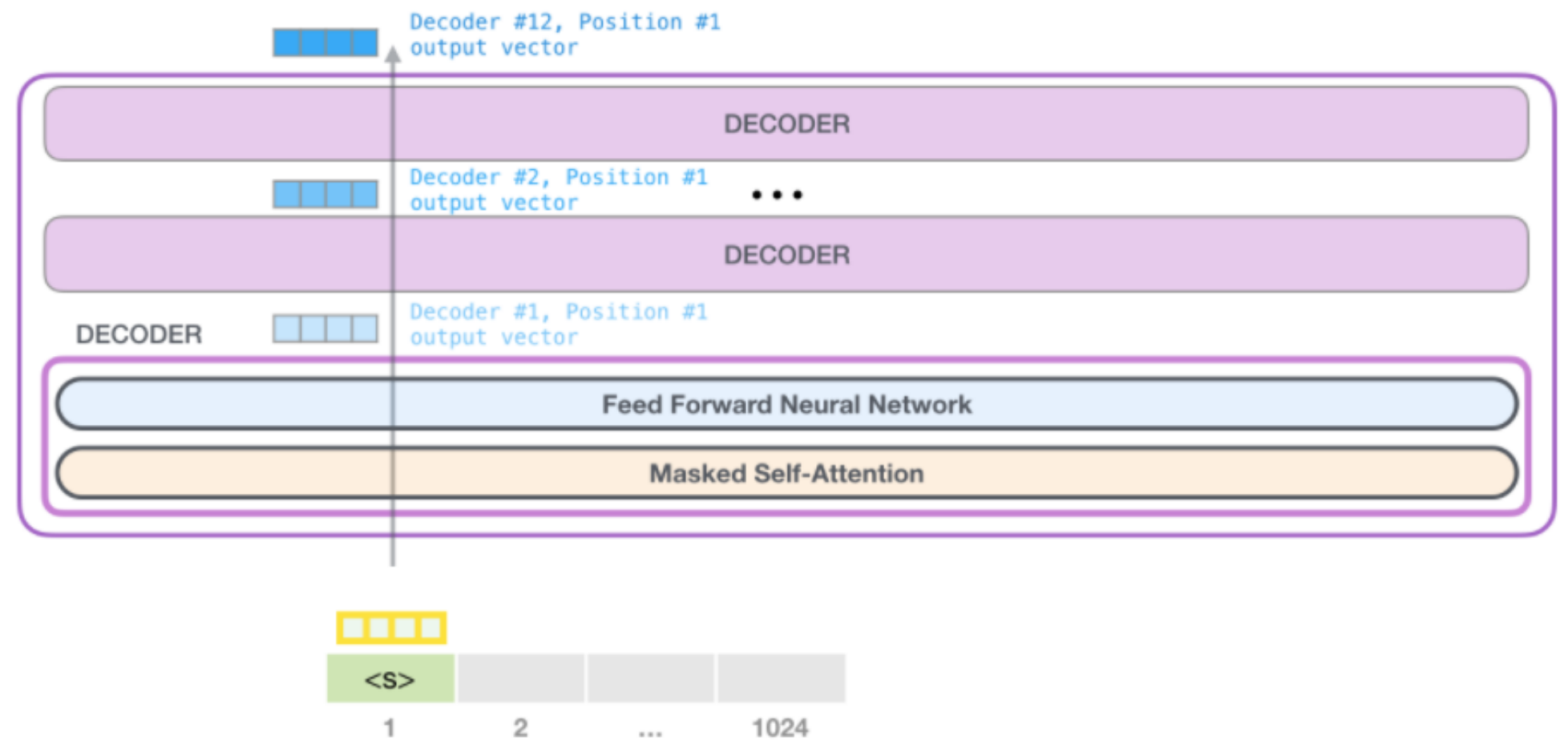


2. Positional Encoding 사용

- 각 단어에 순서 정보를 추가하는 것을 의미
- 순서 정보의 고려
- **input token = embedded token + positional encoding**

02 KoGPT2

입력 벡터는 각 디코더셀의
self-attention 과정을 거친 뒤,
신경망 레이어를 통해 출력



03 실행 결과

이전 사용 방법

```
pytorch_kogpt2 = {  
    'url':  
        'checkpoint/pytorch_kogpt2_676e9bcfa7.params',  
    'fname': 'pytorch_kogpt2_676e9bcfa7.params',  
    'chksum': '676e9bcfa7'  
}
```

모델의 매개변수 ←

불러오기

```
kogpt2_config = {  
    "initializer_range": 0.02,  
    "layer_norm_epsilon": 1e-05,  
    "n_ctx": 1024,  
    "n_embd": 768,  
    "n_head": 12,  
    "n_layer": 12,  
    "n_positions": 1024,  
    "vocab_size": 50000  
}
```

← 학습 파라미터 설정

```
# KoGPT-2 언어 모델 학습을 위한 GPT2LMHeadModel 선언  
kogpt2model= GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')
```

```
device = torch.device(ctx)  
kogpt2model.to(device)
```

```
# 불러오기 부분
```

```
try:  
    checkpoint = torch.load(load_path, map_location=device)
```

```
# KoGPT-2 언어 모델 학습을 위한 GPT2LMHeadModel 선언
```

```
kogpt2model = GPT2LMHeadModel(config=GPT2Config.from_dict(kogpt2_config))
```

```
kogpt2model.load_state_dict(checkpoint['model_state_dict'],strict=False)
```

```
kogpt2model.eval()
```

```
except:
```

```
    count = 0
```

```
else:
```

```
    count = int(re.findall("\d+", load_path)[1])
```

03

실행 결과

모델을 학습 모드로 변환

추가로 학습하기 위해 .train() 사용

```
kogpt2model.train()
```

```
vocab_b_obj = gluonnlp.vocab.BERTVocab.from_sentencepiece(vocab_path,  
                                                         mask_token=None,  
                                                         sep_token=None,  
                                                         cls_token=None,  
                                                         unknown_token='<unk>',  
                                                         padding_token='<pad>',  
                                                         bos_token='<s>',  
                                                         eos_token='</s>')
```

BERT 단어장을 사용

```
tok_path = get_tokenizer()
```

```
model, vocab = kogpt2model, vocab_b_obj
```

```
tok = SentencepieceTokenizer(tok_path)
```

```
dataset = Read_Dataset(data_file_path, vocab, tok)
```

```
print("Read_Dataset ok")
```

```
data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True, pin_memory=True)
```

```
learning_rate = 3e-5
```

```
criterion = torch.nn.CrossEntropyLoss()
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

```
print('KoGPT-2 Transfer Learning Start')
```

```
avg_loss = (0.0, 0.0)
```

03 실행 결과

sizeDataSet

| A | B | |
|-------|-------|------------------------------------|
| score | genre | lyrics |
| 1 | 리뷰 | 저같은 통통이도 1사이즈시켰는데 넉넉하게 입을 수 있었어요 |
| 1 | 리뷰 | 청치마에 넣어 입어도 이쁘고 빼 입고 청바지에 입어도 이쁜데 |
| 1 | 리뷰 | 몸무게 70넘고요 평소 사이즈 100 아니면 XL 입는데 |
| 1 | 리뷰 | 이거는 그냥 오버핏이 아니라 남친꺼 뺏어 입은 것처럼 커요 근 |
| 1 | 리뷰 | 사이즈 고민하다가 리뷰보고 1샀는데 딱이네요 |

```
BOS = '<s>'
EOS = '</s>'
MASK = '<mask>'
NEWLINE = '<unused0>'
PAD = '<pad>'
```

```
def __init__(self, file_path, vocab, tokenizer):
    self.file_path = file_path
    self.data = []
    self.vocab = vocab
    self.tokenizer = tokenizer
    self.bos = BOS
    self.eos = EOS
    self.mask = MASK
    self.pad = PAD
    file = open(self.file_path, 'r', encoding='utf-8')

    df = pd.read_csv(self.file_path)
    print("data read: -----OK-----")
    datasets = []
    for _, row in df.iterrows():
        datasets.append([row["lyrics"], row["genre"], row["score"]])

    print("tokenizer ending")
    for line in datasets:
        if not line[0]:
            break
        if len(line[0]) < 3:
            continue
        tokenized_line = tokenizer(line[0][:-1])
        index_of_words = [vocab[self.bos], ] + vocab[tokenized_line] + [vocab[self.eos]]
        #print(line)
        self.data.append([index_of_words, line[1], line[2]])

    print(np.shape(self.data))
```

토큰나이저 적용

03

실행 결과

```
for epoch in range(epoch):
    for data in data_loader:
        optimizer.zero_grad()
        data = torch.stack(data[0]) # list of Tensor로 구성되어 있기 때문에 list를 stack을 통해 변환해준다. --shape문제 -> data[0]으로 수정
        data = data.transpose(1,0)

        data = data.to(ctx) # 해당 tensor를 GPU에 loading
        model = model.to(ctx)

        outputs = model(data, labels=data)
        loss, logits = outputs[:2]
        loss = loss.to(ctx)
        loss.backward()
        avg_loss = (avg_loss[0] * 0.99 + loss, avg_loss[1] * 0.99 + 1.0)
        optimizer.step()
        if count % 10 == 0:
            print('epoch no.{0} train no.{1} loss = {2:.5f} avg_loss = {3:.5f}' . format(epoch, count, loss, avg_loss[0] / avg_loss[1]))
            summary.add_scalar('loss/avg_loss', avg_loss[0] / avg_loss[1], count)
            summary.add_scalar('loss/loss', loss, count)
```

Sample 10개 볼때 마다 확인

03

실행 결과

Sample 1000개 학습 시 모델 저장

```
if (count > 0 and count % 1000 == 0) or (len(data) < batch_size):  
    # 모델 저장  
    try:  
        torch.save({  
            'epoch': epoch,  
            'train_no': count,  
            'model_state_dict': model.state_dict(),  
            'optimizer_state_dict': optimizer.state_dict(),  
            'loss': loss  
        }, save_path + 'KoGPT2_checkpoint_' + str(count) + '.tar')  
    except:  
        pass
```

내 드라이브 > KoGPT2-finetuning-master > checkpoint ▾ 👤

이름 ↓

- ☰ KoGPT2_checkpoint_11000.tar 👤
- ☰ KoGPT2_checkpoint_1280.tar 👤
- ☰ KoGPT2_checkpoint_1216.tar 👤
- ☰ KoGPT2_checkpoint_1152.tar 👤
- ☰ KoGPT2_checkpoint_1088.tar 👤
- ☰ KoGPT2_checkpoint_1024.tar 👤
- ☰ KoGPT2_checkpoint_1000.tar 👤

03

실행 결과

```
!python main.py --epoch=10 --data_file_path="/content/drive/MyDrive/KoGPT2-finetuning-master/delivery.csv" --load_path="" --batch_size=64

/usr/local/lib/python3.7/dist-packages/mxnet/optimizer/optimizer.py:167: UserWarning: WARNING: New optimizer gluonnlp.optimizer.lamb.LAMB is not registered. Please register it manually (e.g., Optimizer.opt_registry[name].__name__)
  Optimizer.opt_registry[name].__name__)
using cached model
0
using cached model
data read: -----OK-----
tokenizer ending
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequence (which Python has converted to a list of lists) is deprecated. If you determine this is a safe operation you may need to `astype(object, copy=False)` before creating an ndarray. Mind the reference.
  return array(a, dtype, copy=False, order=order)
(4070, 3)
Read_Dataset ok
KoGPT-2 Transfer Learning Start
epoch no.0 train no.0  loss = 14.84440 avg_loss = 14.84440
1
2
3
4
5
6
7
8
9
10
```

03 **실행 결과**

[illegible]

감사합니다!

20190431 박규현