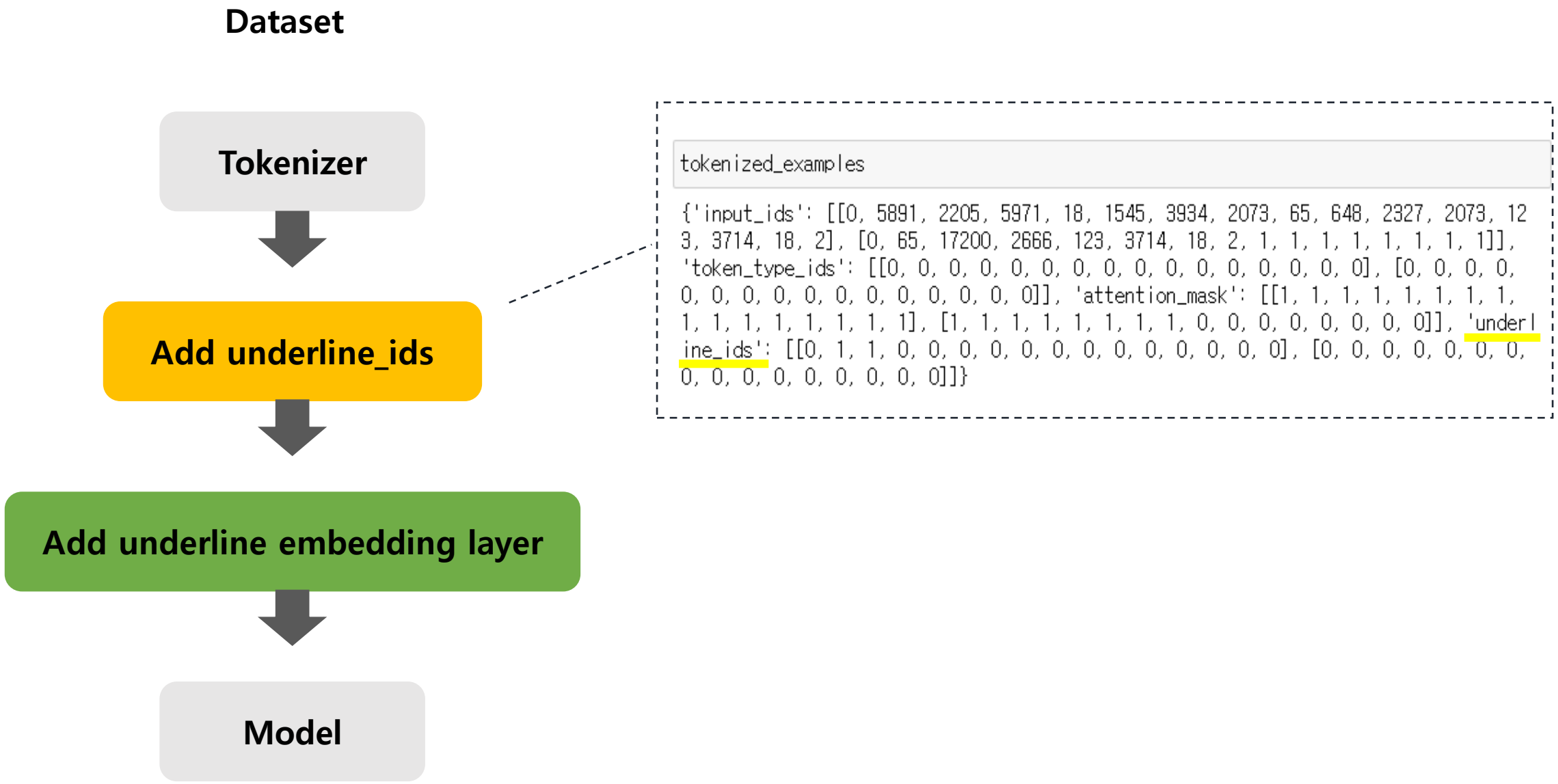
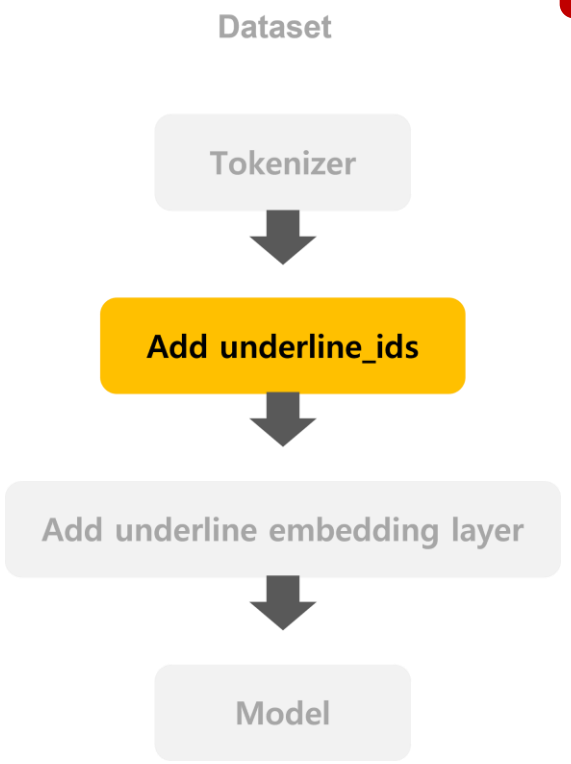
Underline Embedding Layer

핵심 문장을 파악하라 !

Team KiYOUNG2, Chaeeun Kim

“ 핵심 문장을 강조해서 읽도록 punctuation 및 underline embedding layer추가 ”





batch

```python
tokenized_examples = tokenize_fn(examples)

if data_args.underline == True:
    tokenized_examples = get_underline_embedding(tokenized_examples)
```

```python
def get_underline_embedding(tokenized_examples):

    underline_ids = np.zeros_like(tokenized_examples['input_ids'])

    punct_start_token = '^'
    punct_end_token = '※'

    punct_start_id = tokenizer.convert_tokens_to_ids(punct_start_token)
    punct_end_id = tokenizer.convert_tokens_to_ids(punct_end_token)

    for i, row in enumerate(tokenized_examples['input_ids']):
        if punct_start_id in row and punct_end_id in row:
            punct_start = row.index(punct_start_id)
            punct_end = row.index(punct_start_id)
            underline_ids[i][punct_start+1:punct_end] = 1
        else:
            continue

    underline_ids = underline_ids.tolist()
    tokenized_examples.update({"underline_ids": underline_ids})

    return tokenized_examples
```

예시

※ context → KSS (sentence split) → sentences

- question : MRC대회 기간은?
- answer : 4주
- context : 어느덧 11월이다. ^4주간의 긴 MRC대회가 끝났다.※ 다음은 최적화 대회다.

punctuation

train dataset

- 정답이 포함된 문장 양 끝에 punctuation추가
- 정답이 포함된 문장은 1로 embedding

test dataset

- 질문과 유사도가 높은 문장에 punctuation추가
- 질문과 유사도가 높은 문장은 1로 embedding

```
sentence_encoder.eval()

with torch.no_grad():
    p_outputs = sentence_encoder(**p_inputs)
    q_outputs = sentence_encoder(**q_inputs)

dot_prod_scores = torch.matmul(
    q_outputs, torch.transpose(p_outputs, 0, 1))
rank = torch.argsort(dot_prod_scores, dim=1,
                     descending=True).squeeze()
topk_sentences = rank[:self.args.top_k_punctuation].tolist()

new_contexts = []
for i, sentence in enumerate(contexts):
    if i in topk_sentences:
        sentence = '^' + sentence + '※'
        new_contexts.append(sentence)
    else:
        new_contexts.append(sentence)

return " ".join(new_contexts)
```

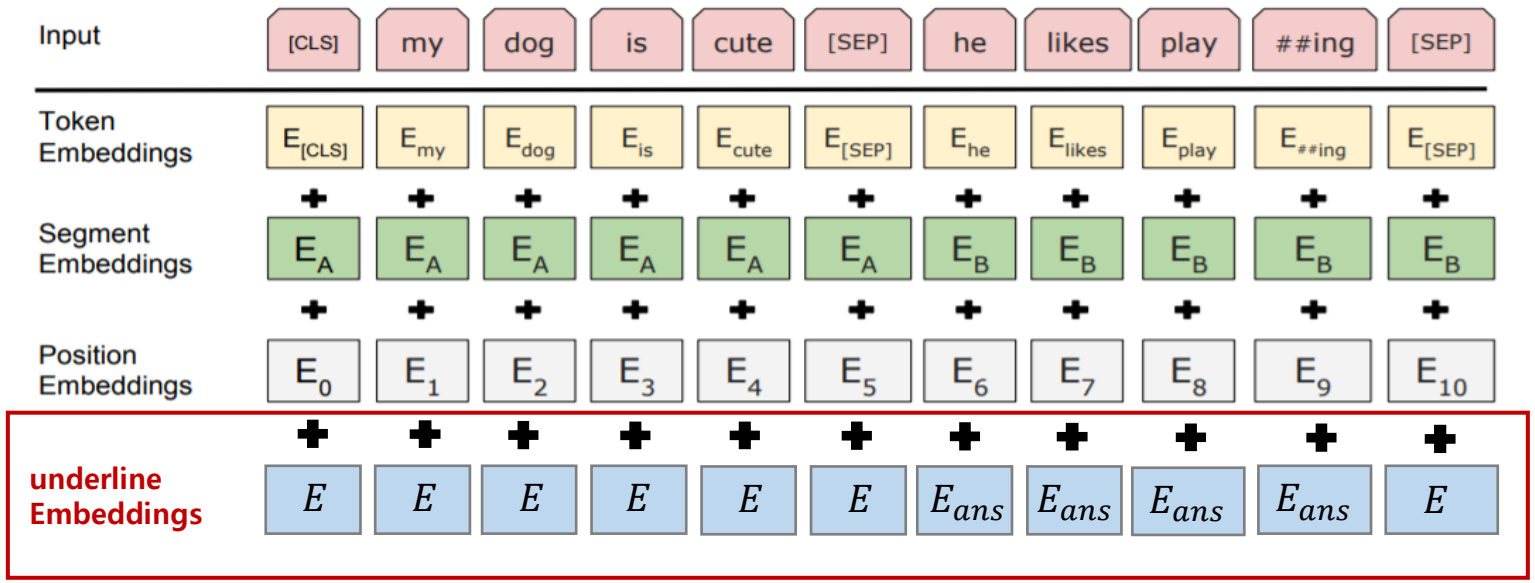
Dataset

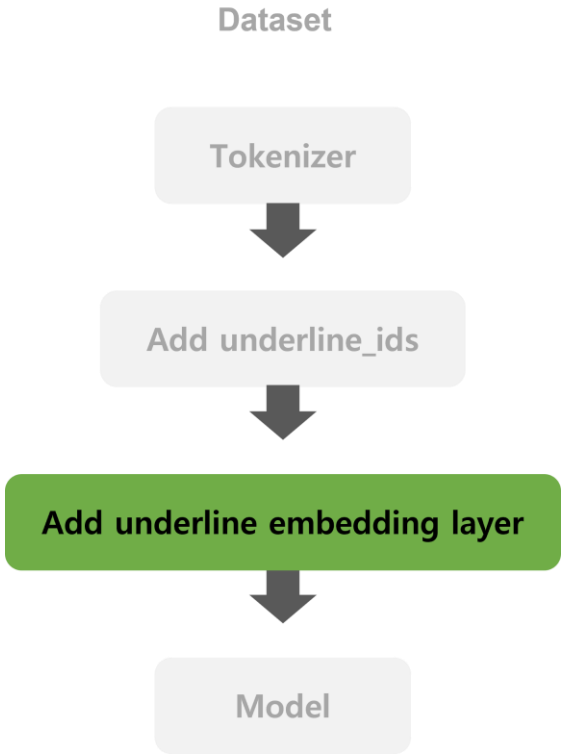
Tokenizer

Add underline_ids

Add underline embedding layer

Model





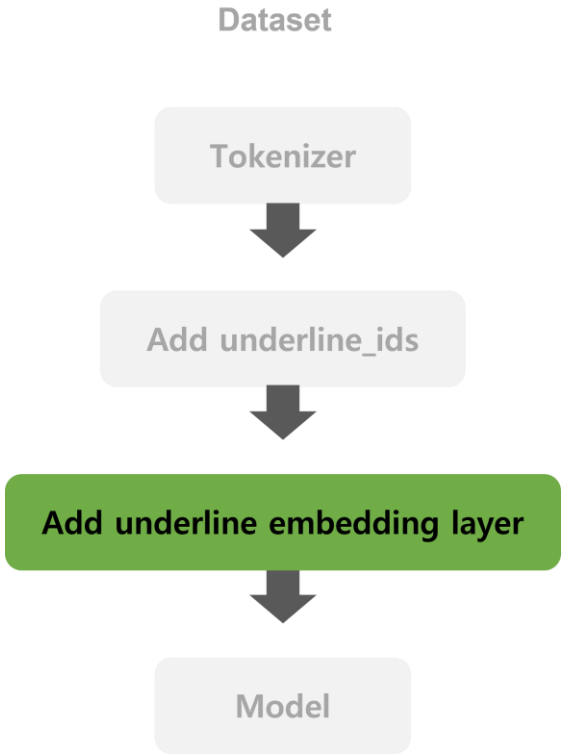

```python
class RobertaForQAWithUnderline(RobertaPreTrainedModel):
    reader_type: str = "extractive"

    _keys_to_ignore_on_load_unexpected = [r"pooler"]
    _keys_to_ignore_on_load_missing = [r"position_ids"]

    def __init__(self, config):
        super().__init__(config)
        assert config.reader_type == self.reader_type

        config.num_labels = 2
        self.num_labels = config.num_labels

        self.roberta = RobertaModelWithUnderline(
            config, add_pooling_layer=False)
```




```python
class RobertaEmbeddingsWithUnderline(RobertaEmbeddings):

    def __init__(self, config):
        super().__init__(config)
        self.underline_embeddings = nn.Embedding(2, config.hidden_size)
        self.config = config
```

```python
class RobertaModelWithUnderline(RobertaPreTrainedModel):

    _keys_to_ignore_on_load_missing = [r"position_ids"]

    # Copied from transformers.models.bert.modeling_bert.BertModel
    def __init__(self, config, add_pooling_layer=True):
        super().__init__(config)
        self.config = config

        self.embeddings = RobertaEmbeddingsWithUnderline(config)
```

- **train dataset**

```
Run summary:
          eval/exact_match 72.91667
                   eval/f1 81.34292
              eval/runtime 9.7051
   eval/samples_per_second 48.119
     eval/steps_per_second 3.091
               train/epoch 1.85
          train/global_step 4300
        train/learning_rate 0.0
                 train/loss 0.0564
```

```
{'eval_exact_match': 74.58333333333333, 'eval_f1': 82.11078042328043,
```

1. punctuation 및 underline의 강조효과가 강력한 듯..
2. train에서는 정답이 포함된 문장을 강조하여 answer을 잘 예측하였으나,
3. inference에서는 유사도가 높은 문장을 강조하게 됨.
4. 유사도가 높다고 하여 정답이 포함되었다고 할 수 없음.
5. train과 inference모두 유사도를 기반으로 강조하도록 동일하게 세팅할 수 있을 듯
6. topk_punctuation은 최대한으로 주는 것이 안전할 것