자연언어처리 특강

11/15 (수)

전북대학교
JEONBUK NATIONAL UNIVERSITY

Overview

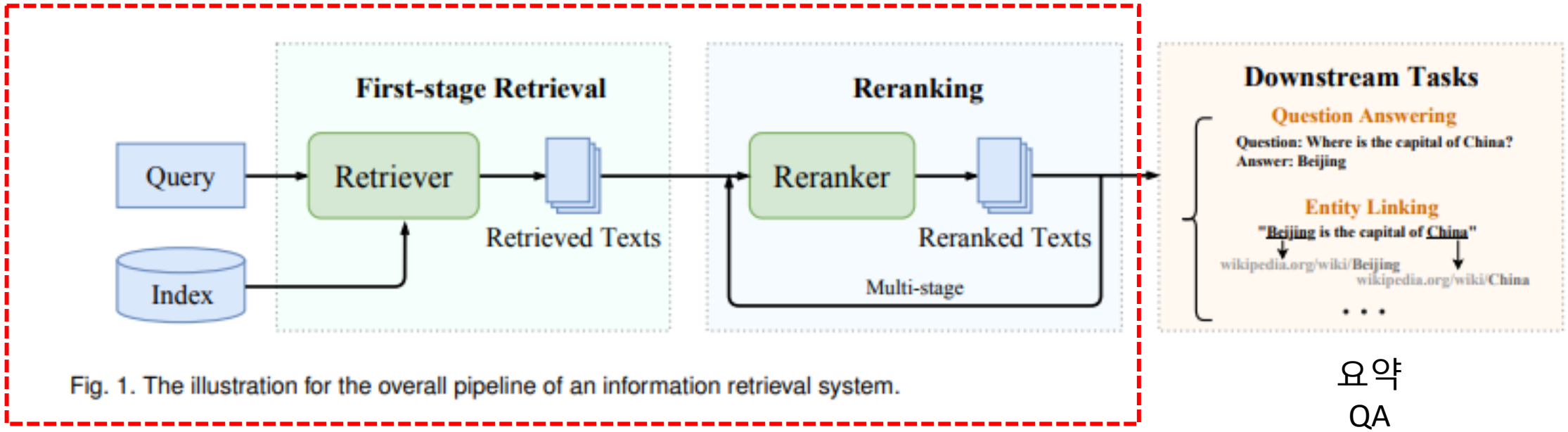


Fig. 1. The illustration for the overall pipeline of an information retrieval system.

요약
QA
…

관련 있는 문서들을 어떻게 찾을 것인가?
- Sparse Retrieval
- Dense Retrieval

Sparse Retrieval vs Dense Retrieval

- Sparse Retrieval model
  - 전통적인 검색 방법 중 하나로, lexical matching을 기반으로 작동한다.
  - TF-IDF, BM25
  - 장점
    - 학습이 필요 없다.
    - 웬만한 환경에서도 잘 작동한다.
    - 매우 빠르다.
  - 단점
    - 문맥을 파악하지 못한다. (단어의 순서나 관련성을 고려하지 않는다.)
- Dense Retrieval model
  - LM의 대두와 함께 사용되는 방법으로, Semantic matching을 기반으로 작동한다.
  - 문서와 질의를 고차원 벡터 공간으로 매핑하여 유사성을 평가한다.
  - Bi-Encoder, Cross-Encoder
  - 장점
    - **문맥을 고려**하므로 더 나은 검색 성능을 제공할 수 있다.
  - 단점
    - 계산 비용이 높고 시간이 많이 소요된다. (특히 큰 데이터셋에서는 효율성이 떨어진다.)
    - 검색 환경에 맞게 학습되지 않으면 좋은 성능을 기대하기 어렵다.

Vocabulary mismatch problem

NETFLIX | 고객 센터

← 고객 센터 홈으로 돌아가기

넷플릭스 해지 방법

넷플릭스 계정은 언제든지 해지할 수 있습니다.

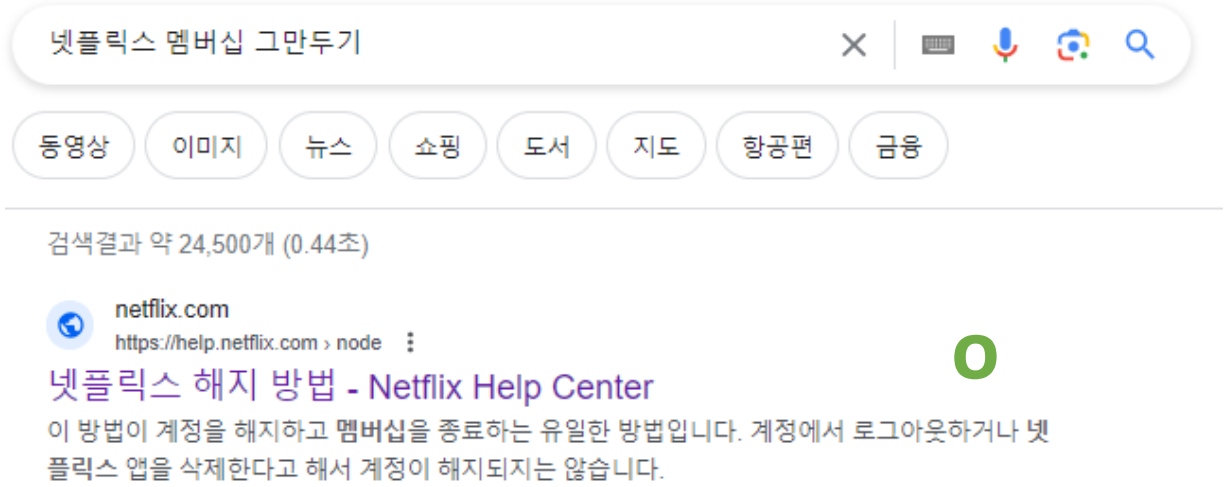
- 스트리밍 멤버십 해지
- **계정** 페이지에서의 DVD 멤버십 해지

> **참고:** 이 방법이 계정을 해지하고 멤버십을 종료하는 유일한 방법입니다. 계정이 해지되지는 않습니다.

NAVER

X

Google



O

Dense Retrieval: Bi-Encoder vs Cross-Encoder

- Bi-encoder (=dual encoder)
  - 두 개의 임베딩(encoder) 네트워크를 사용하여 문서와 질의 각각을 임베딩한 후, 유사도를 측정하여 관련성이 높은 문서들을 찾는다.
  - 장점
    - 문서들을 미리 임베딩하여 저장해놓을 수 있다. → 실제 검색 시, 질의에 대해서 임베딩한 후, 유사도 측정만을 진행하면 되기에 소모시간이 비교적 적다.
  - 단점
    - 질의-문서에 대해 같이 고려하지 않기 때문에, cross-encoder에 비해 정확성이 떨어진다.
- Cross-encoder
  - 하나의 임베딩 네트워크를 사용하여 문서와 질의 쌍을 같이 넣어준다.
  - 장점
    - 질의와 문서 간의 상호작용을 고려하기 때문에 비교적 정확하다.
  - 단점
    - 질의-모든 문서들에 대해 추론해야 하기 때문에, 시간이 많이 소모된다.

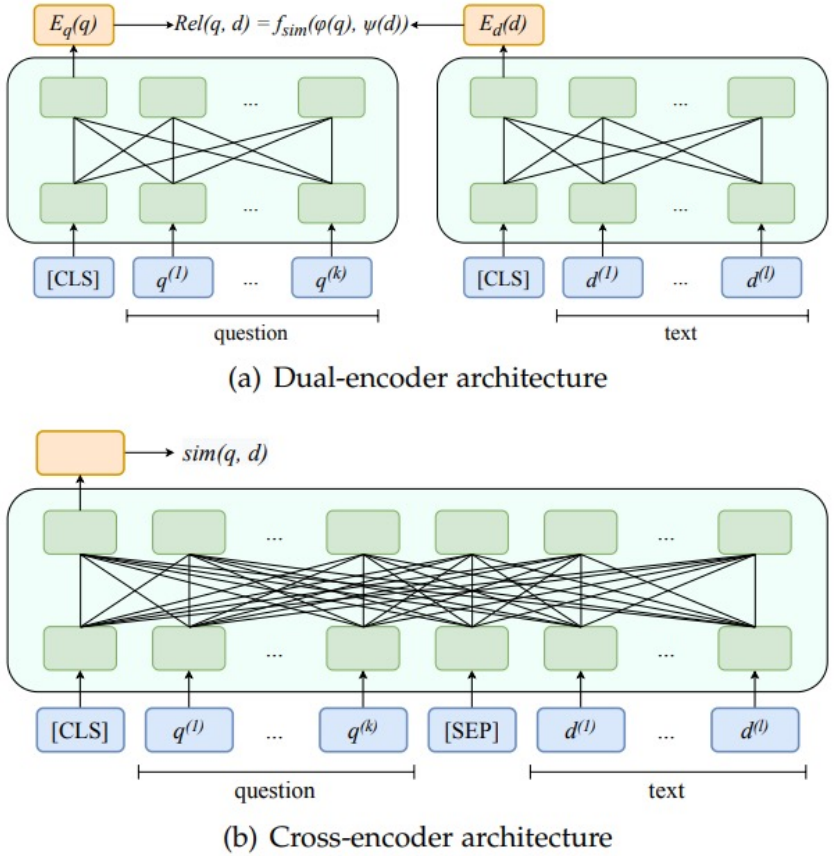
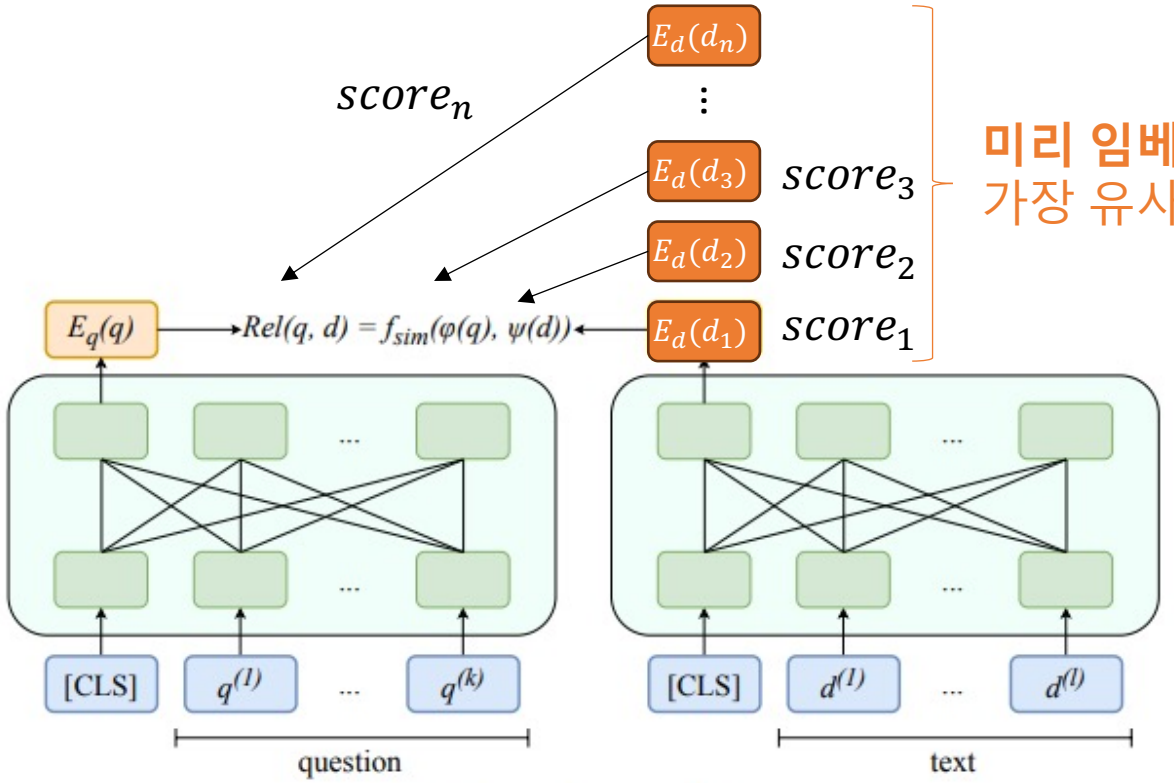


Fig. 2. Visual illustration of dual-encoder and cross-encoder architectures.

Dense Retrieval: Bi-Encoder



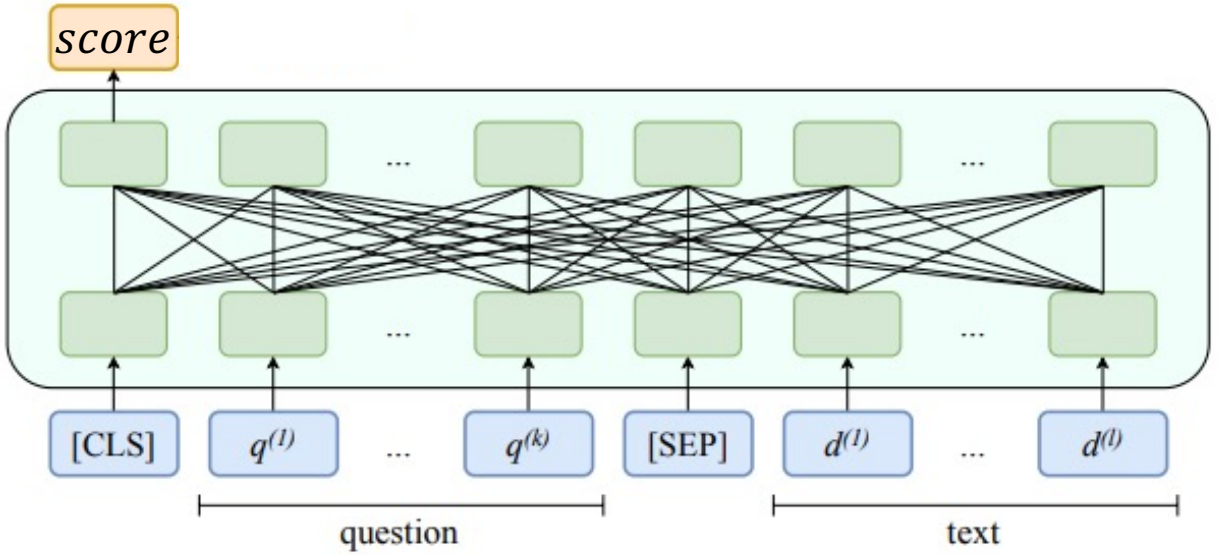
(a) Dual-encoder architecture

미리 임베딩된 문서들에 대해 cosine similarity를 구한다.
가장 유사성이(=점수) 높은 문서들을 반환한다.

Input: $query$

d_1 d_2 ... d_n

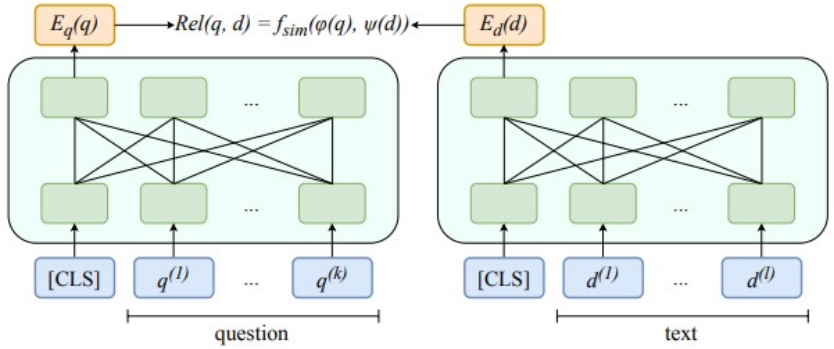
Dense Retrieval: Cross-Encoder



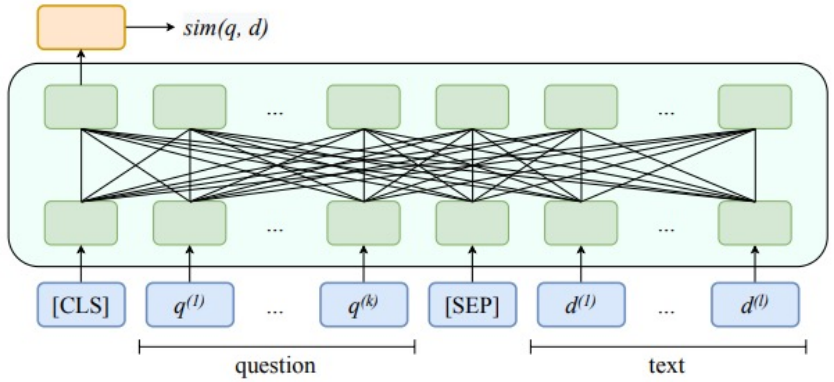
(b) Cross-encoder architecture

Input: $[query, d_1]$, $[query, d_2]$, ... $[query, d_n]$

구현

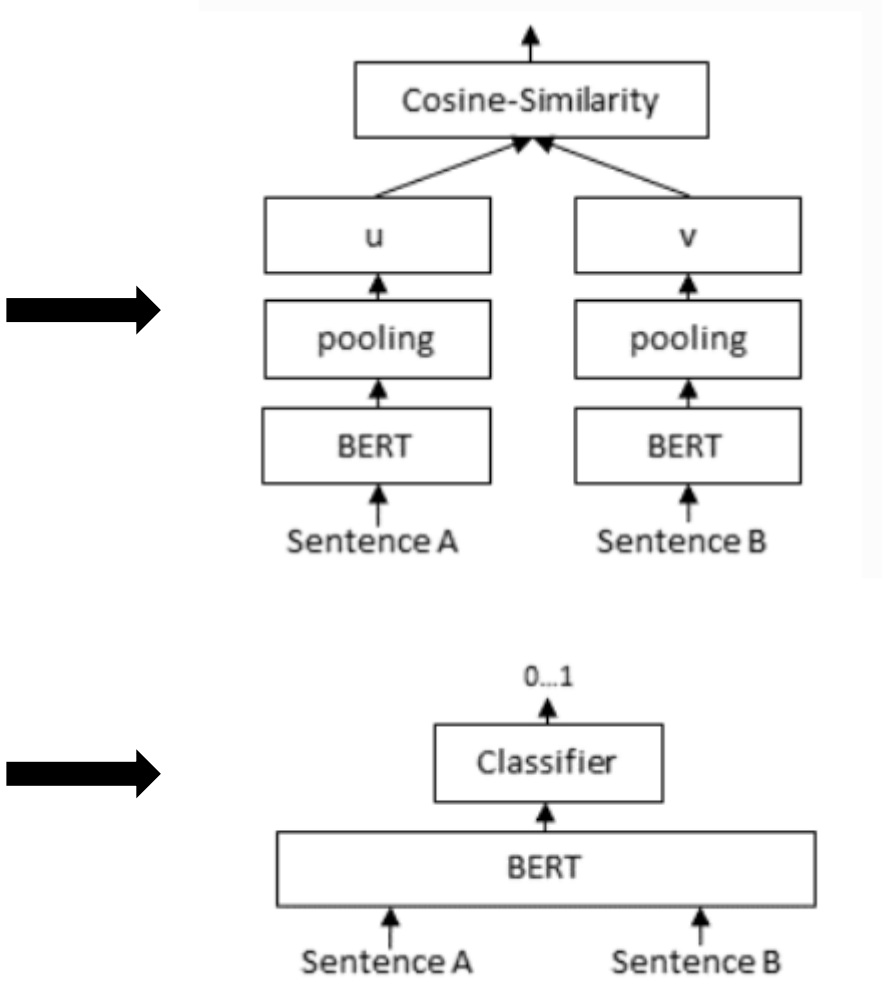


(a) Dual-encoder architecture



(b) Cross-encoder architecture

Fig. 2. Visual illustration of dual-encoder and cross-encoder architectures.



사용 데이터 셋 및 모델

- 데이터 셋
  - Mr-tydi
    - https://huggingface.co/datasets/castorini/mr-tydi/viewer/korean
  - KorQuad
    - https://korquad.github.io/
    - https://huggingface.co/datasets/squad_kor_v1
- 모델
  - BERT
    - bert-base-multilingual-uncased
    - https://huggingface.co/bert-base-multilingual-uncased

    → 추후 한국어 데이터 셋으로 학습된 koBERT를 사용할 것을 권장
      - https://huggingface.co/skt/kobert-base-v1

Huggingface란?

- 자연어 처리(Natural Language Processing, NLP) 및 기계 학습(Machine Learning) 분야에서 사용되는 다양한 모델 및 도구들을 제공하는 오픈 소스 플랫폼이다. Transformer의 아키텍처나 관련 라이브러리들을 지원하며, 사전학습된 모델의 체크포인트나 데이터 셋 등을 쉽게 공유할 수 있는 환경을 제공한다.

Hugging Face

https://huggingface.co/

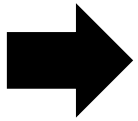
예시 – Dataset

- korQuAD homepage

KorQuAD 2.1의 전체 데이터는 47,957 개의 Wikipedia article 에 대해 102,960 개의 질의응답 쌍으로, Training set 83,486 개, Dev set 10,165 개의 질의응답쌍으로 구분하였습니다.
KorQuAD 2.0 데이터 중 HTML 태그의 속성이 완벽하게 제거되지 않은 오류를 수정하여 재배포한 데이터셋으로, KorQuAD 2.0의 태그 속성을 제외한 원본과 정답 텍스트가 바뀌는 경우는 없습니다.
KorQuAD 2.0의 데이터셋은 CC BY-ND 2.0 KR 라이센스를 따릅니다.
Codalab을 통한 모델 제출시 테스트 스코어 계산 및 리더보드를 통한 스코어 공개에 동의한 것으로 간주합니다. 참고로 제출한 모델 및 소스 코드 등에 대해서는 참가자가 직접 라이센스를 부여하고 이를 명시할 경우 그에 따릅니다.

TRAINING SET (6.4GB)

DEV SET (794MB)



Datasets: **squad_kor_v1** like 10

Tasks: Question Answering | Sub-tasks: extractive-qa | Languages: Korean | Multilinguality: monolingual | Size Categories: 10K<n<100K | Lan

Source Datasets: original | ArXiv: arxiv:1909.07005 | License: cc-by-nd-4.0

Dataset card | Files and versions | Community 1

Dataset Viewer Auto-converted to Parquet | API | Go to dataset viewer

Split: train (60.4k rows)

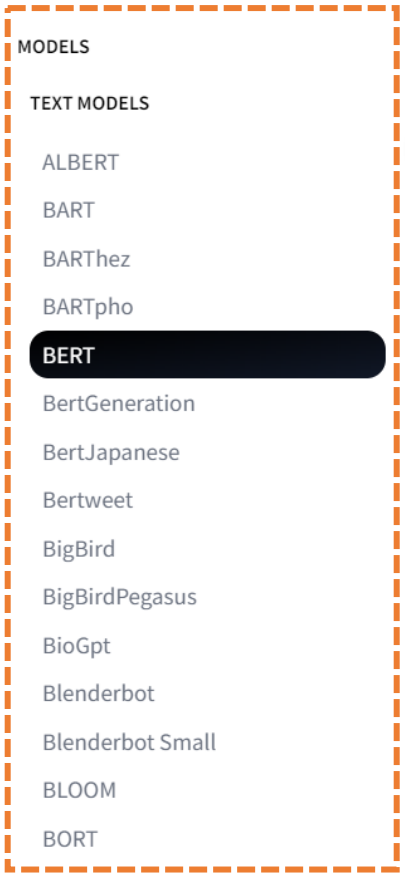
Search this dataset

id string	title string	context string	question string	answers sequence
6566495-0-0	파우스트_서곡	1839년 바그너는 괴테의 파우스트을 처음 읽고 그 내용에 마음이 끌려 이를 …	바그너는 괴테의 파우스트를 읽고 무엇을 쓰고자 했는가?	{ "text": ["교향곡"], "answer_start": [54] }
6566495-0-1	파우스트_서곡	1839년 바그너는 괴테의 파우스트을 처음 읽고 그 내용에 마음이 끌려 이를 …	바그너는 교향곡 작곡을 어디까지 쓴 뒤에 중단했는가?	{ "text": ["1악장"], "answer_start": [421] }
6566495-0-2	파우스트_서곡	1839년 바그너는 괴테의 파우스트을 처음 읽고 그 내용에 마음이 끌려 이를 …	바그너가 파우스트 서곡을 쓸 때 어떤 곡의 영향을 받았는가?	{ "text": ["베토벤의 교향곡 9번"], "answer_start": [194] }

코드 한 줄로 다운로드 가능! 바로 파이썬에서 사용 가능함

```
dataset = load_dataset('squad_kor_v1')
```

예시 – 모델 아키텍처



BERT

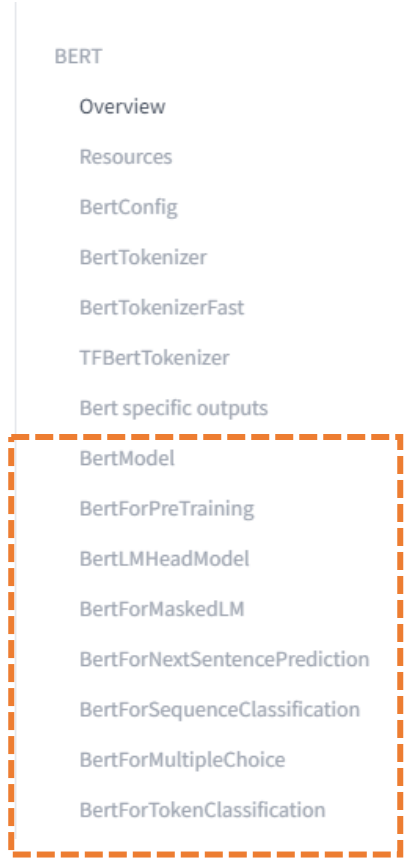
All model pages bert Hugging Face Spaces

Overview

The BERT model was proposed in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. It's a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia.

The abstract from the paper is the following:

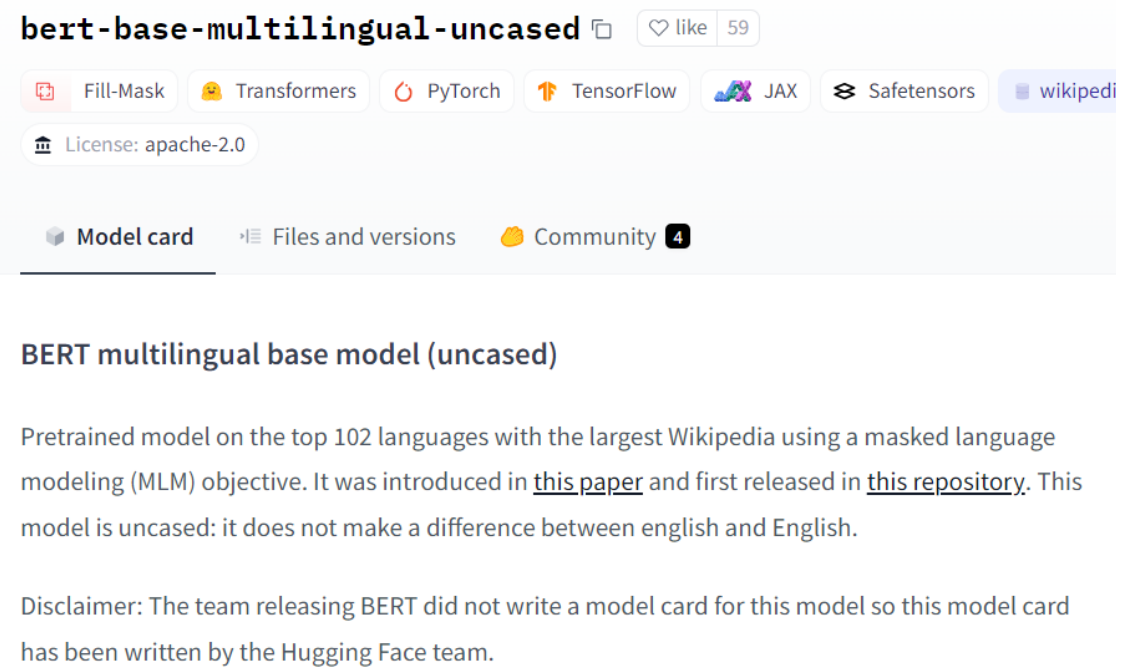
We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

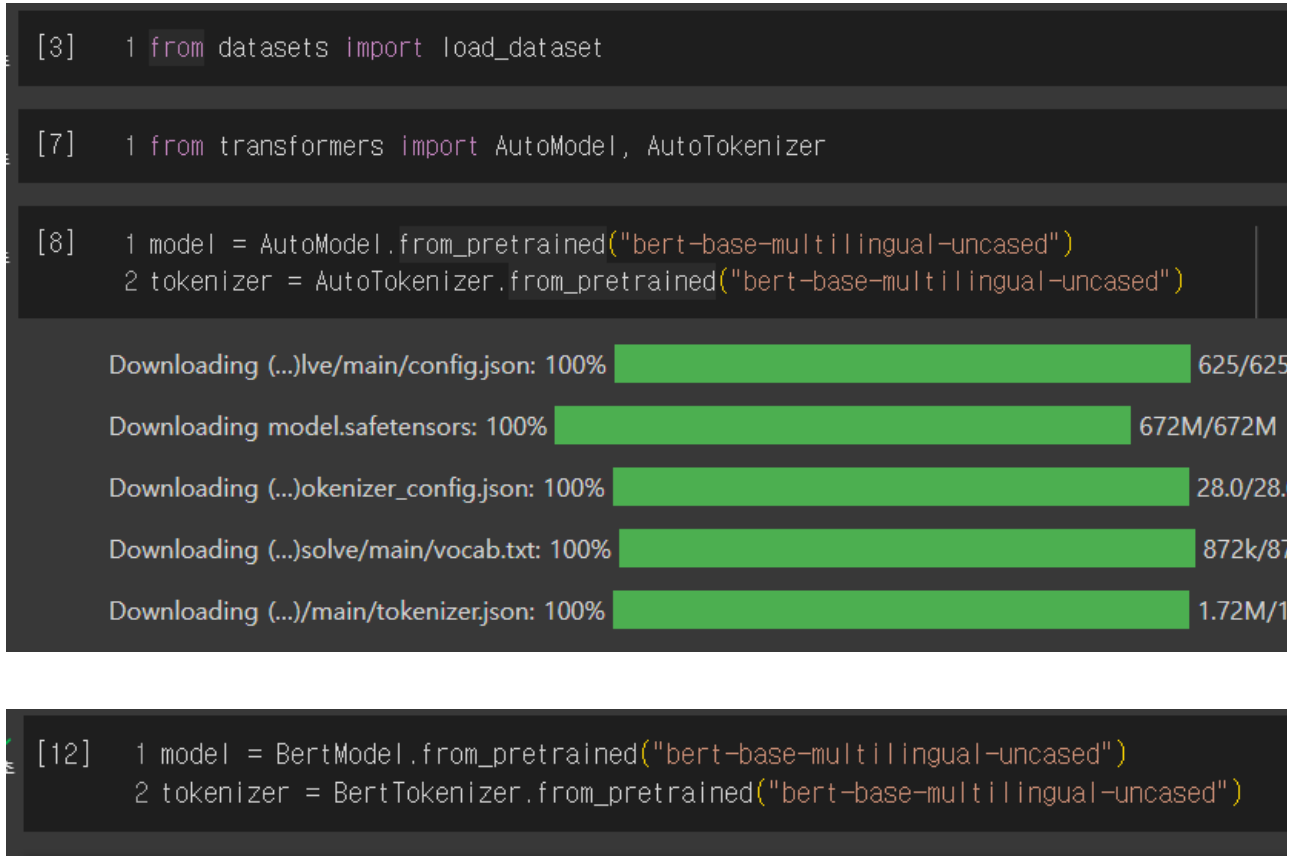


https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForSequenceClassification

예시 – pretrained model

- 미리 학습시켜 놓은 모델을 코드 한 줄로 다운로드 가능



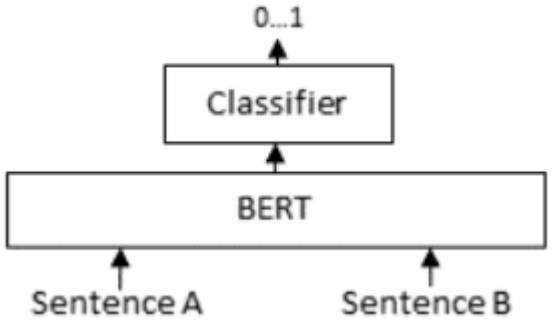


실습

- BM25, Bi-encoder, Cross-Encoder 사용해보기
  - https://colab.research.google.com/drive/1Gx8sjSWz3W64ZbhPRTX_3uyHvisqtigS#scrollTo=D_hDi8KzNgMM
- SenteceBERT 라이브러리를 이용한 Bi-encoder, Cross-Encoder 학습
  - https://colab.research.google.com/drive/1KetpPOHrmSgif6UXmIJovl2phcKAZdTg#scrollTo=rqUfYfFROkq7

실습 코드 1 – 아키텍처

- Cross-encoder




```
class BertForSequenceClassification(BertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.config = config

        self.bert = BertModel(config)
        classifier_dropout = (
            config.classifier_dropout if config.classifier_dropout is not N
        )
        self.dropout = nn.Dropout(classifier_dropout)
        self.classifier = nn.Linear(config.hidden_size, config.num_labels)
```

```
def forward(
    outputs = self.bert(
        input_ids,
        attention_mask=attention_mask,
        token_type_ids=token_type_ids,
        position_ids=position_ids,
        head_mask=head_mask,
        inputs_embeds=inputs_embeds,
        output_attentions=output_attentions,
        output_hidden_states=output_hidden_states,
        return_dict=return_dict,
    )

    pooled_output = outputs[1]

    pooled_output = self.dropout(pooled_output)
    logits = self.classifier(pooled_output)
    reshaped_logits = logits.view(-1, num_choices)
```

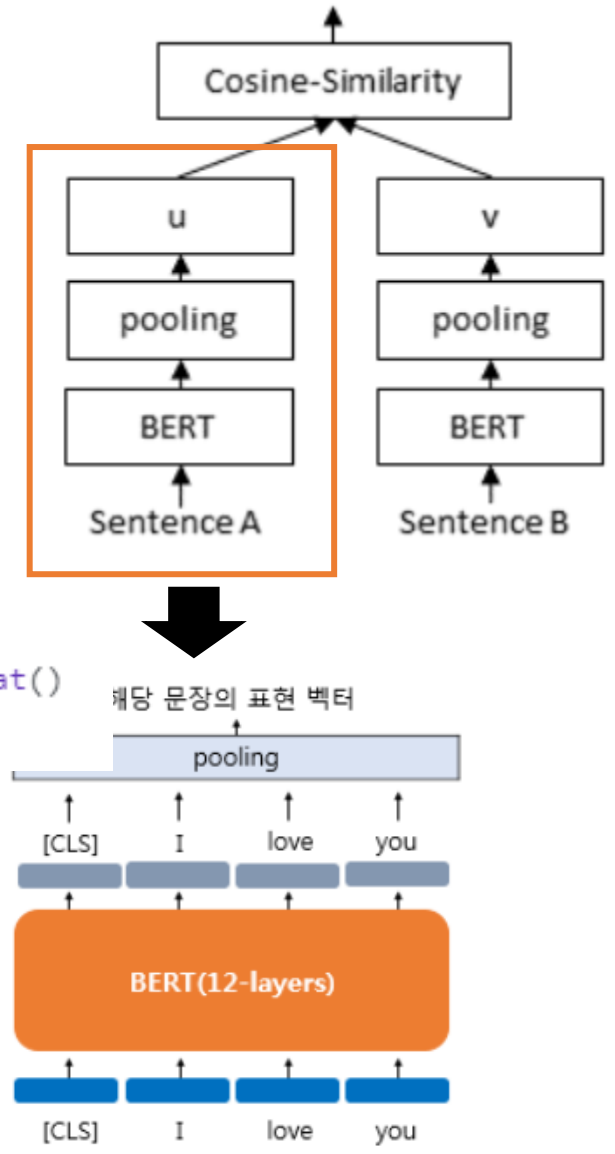
실습 코드 1 – 아키텍처

- Bi-encoder

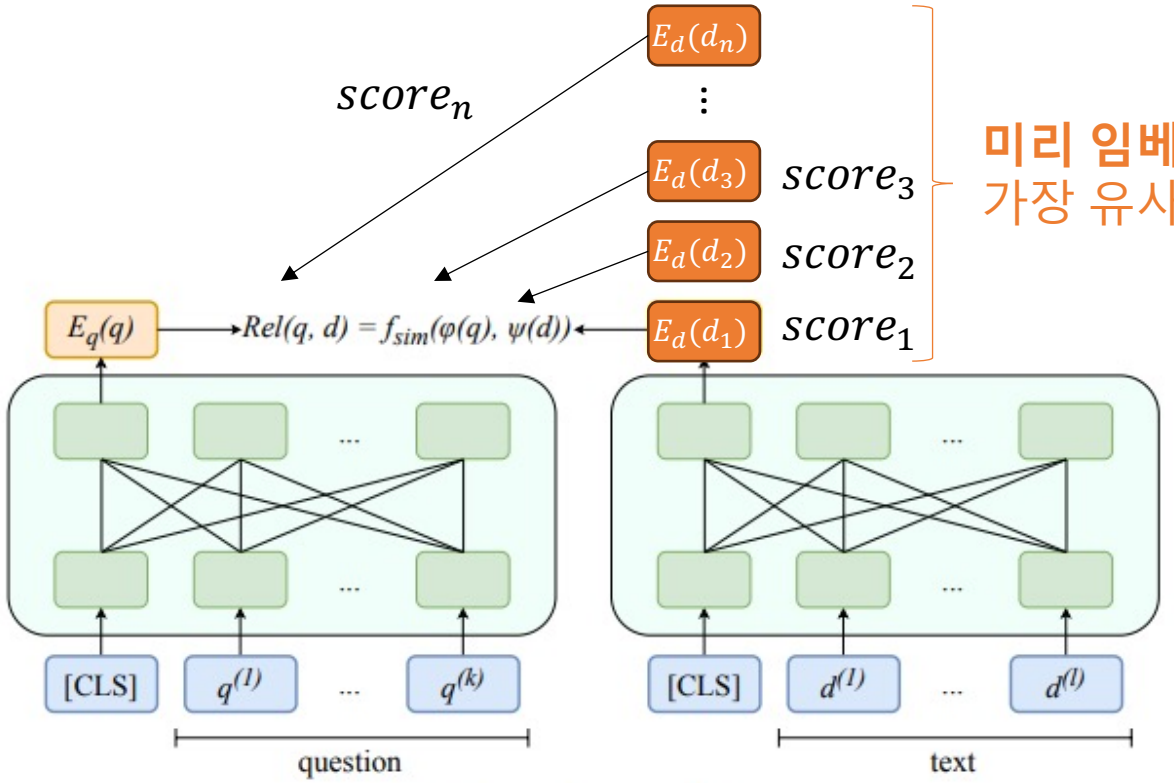
```
transformer_model = Transformer(model_name_or_path)
pooling_model = Pooling(transformer_model.get_word_embedding_dimension(), 'mean')
```

```
self.auto_model = AutoModel.from_pretrained(model_name_or_path, config=config, cache_dir=cache_dir,
```

```
if self.pooling_mode_mean_tokens or self.pooling_mode_mean_sqrt_len_tokens:
    input_mask_expanded = attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()
    sum_embeddings = torch.sum(token_embeddings * input_mask_expanded, 1)
    output_vectors.append(sum_embeddings / sum_mask)
```



실습 코드 1 : Bi-Encoder



(a) Dual-encoder architecture

미리 임베딩된 문서들에 대해 cosine similarity를 구한다.
가장 유사성이(=점수) 높은 문서들을 반환한다.

Input: $query$

d_1 d_2 ... d_n

실습 코드 1 – Sentence Transformers

1. Model load

```
# Bi-encoder
bi_encoder = SentenceTransformer('intfloat/multilingual-e5-large')
```

2. 문서들을 벡터 공간으로 매핑 (임베딩)
   - Input: [CLS]문서[SEP]
   - Output: embedding

```
1 # passage들을 vector space로 mapping한다.
2 corpus_embeddings = bi_encoder.encode(passages, convert_to_tensor=True, show_progress_bar=True)
```

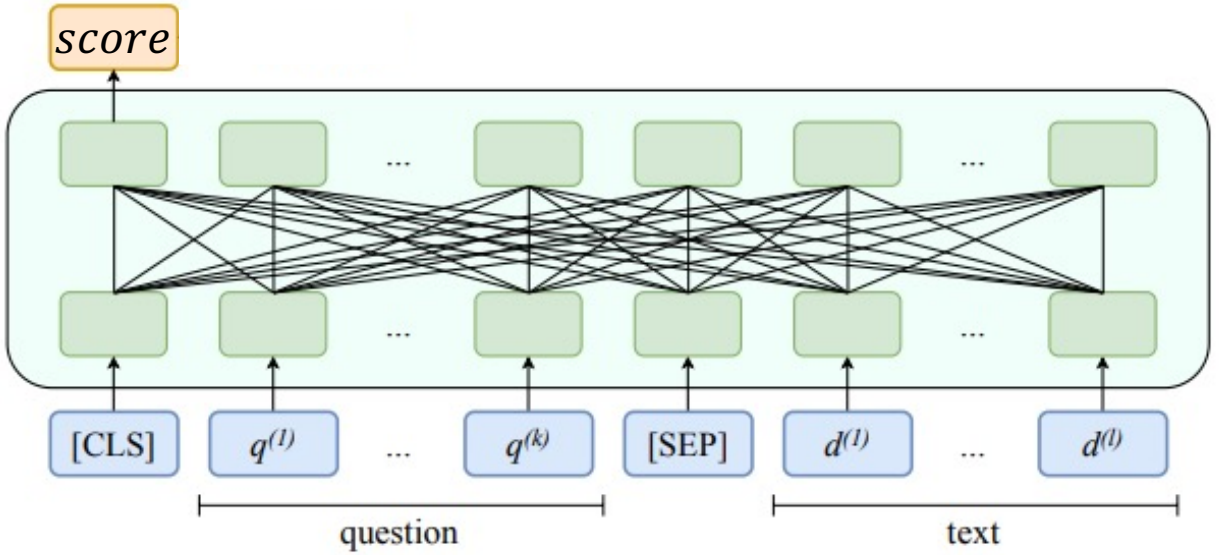
3. 질의 임베딩
   - Input: [CLS]질의[SEP]
   - Output: embedding

```
question_embedding = bi_encoder.encode(query, convert_to_tensor=True)
```

4. 문서 임베딩들과 질의 임베딩에 대해 유사도(=점수) 측정 후 랭킹
   - Cosine similarity 측정
   - Faiss 사용 시 빠르게 비교 가능

```
hits = util.semantic_search(question_embedding, corpus_embeddings, top_k=top_k)
```

실습 코드 1 : Cross-Encoder



(b) Cross-encoder architecture

Input: $[query, d_1]$, $[query, d_2]$, ... $[query, d_n]$

실습 코드 1 – Sentence Transformers

1. Model load

```
# Cross-encoder
cross_encoder = CrossEncoder('bongsoo/albert-small-kor-cross-encoder-v1')
```

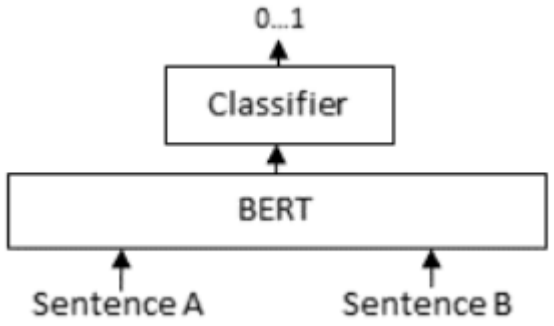
2. 질의와 문서 쌍에 대해 유사도(=점수) 측정
   - Input: [CLS]질의[SEP]문서[SEP]
   - Output: Score

```
cross_inp = [[query, passages[hit['corpus_id']]] for hit in hits]
cross_scores = cross_encoder.predict(cross_inp)
```

3. 유사도가 높은 순으로 랭킹

실습 코드 2 - Cross Encoder 구현

```python
class CrossEncoder():
    def __init__(self, model_name:str, num_labels:int = None, max_length:int = None, device:str = None, tokenizer_args:Dict = {},
                 automodel_args:Dict = {}, default_activation_function = None):

        self.model = AutoModelForSequenceClassification.from_pretrained(model_name, config=self.config, **automodel_args)
        self.tokenizer = AutoTokenizer.from_pretrained(model_name, **tokenizer_args)
        self.max_length = max_length

        if device is None:
            device = "cuda" if torch.cuda.is_available() else "cpu"
            logger.info("Use pytorch device: {}".format(device))

        self._target_device = torch.device(device)
```




```python
class BertForSequenceClassification(BertPreTrainedModel):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.config = config

        self.bert = BertModel(config)
        classifier_dropout = (
            config.classifier_dropout if config.classifier_dropout is not N
        )
        self.dropout = nn.Dropout(classifier_dropout)
        self.classifier = nn.Linear(config.hidden_size, config.num_labels)
```

```python
def forward(
    outputs = self.bert(
        input_ids,
        attention_mask=attention_mask,
        token_type_ids=token_type_ids,
        position_ids=position_ids,
        head_mask=head_mask,
        inputs_embeds=inputs_embeds,
        output_attentions=output_attentions,
        output_hidden_states=output_hidden_states,
        return_dict=return_dict,
    )

    pooled_output = outputs[1]

    pooled_output = self.dropout(pooled_output)
    logits = self.classifier(pooled_output)
    reshaped_logits = logits.view(-1, num_choices)
```

실습 코드 2 - Cross Encoder 구현

```
class CrossEncoder():
    def __init__(self, model_name:str, num_labels:int = None, max_length:int = None, device:str = None, tokenizer_args:Dict = {},
                 automodel_args:Dict = {}, default_activation_function = None):

    def fit(self,
            train_dataloader: DataLoader,
            evaluator: SentenceEvaluator = None,
            epochs: int = 1,
            loss_fct = None,
            activation_fct = nn.Identity(),
            scheduler: str = 'WarmupLinear',
            warmup_steps: int = 10000,
            optimizer_class: Type[Optimizer] = torch.optim.AdamW,
            optimizer_params: Dict[str, object] = {'lr': 2e-5},
            weight_decay: float = 0.01,
            evaluation_steps: int = 0,
            output_path: str = None,
            save_best_model: bool = True,
            max_grad_norm: float = 1,
            use_amp: bool = False,
            callback: Callable[[float, int, int], None] = None,
            show_progress_bar: bool = True
            ):
```

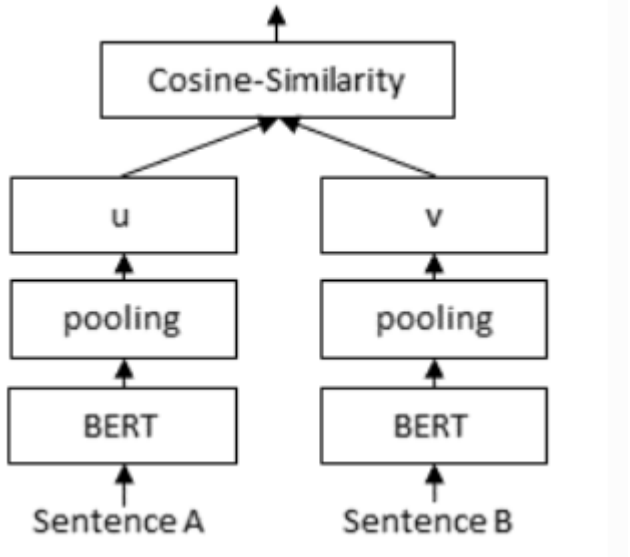
실습 코드 2 - Cross Encoder 구현

```
def fit(self,
    if isinstance(scheduler, str):
        scheduler = SentenceTransformer._get_scheduler(optimizer, scheduler=scheduler, warmup_steps=warmup_steps,

    if loss_fct is None:
        loss_fct = nn.BCEWithLogitsLoss() if self.config.num_labels == 1 else nn.CrossEntropyLoss()

    for epoch in trange(epochs, desc="Epoch", disable=not show_progress_bar):
        training_steps = 0
        self.model.zero_grad()
        self.model.train()

        for features, labels in tqdm(train_dataloader, desc="Iteration", smoothing=0.05, disable=not show_progress_bar):
            model_predictions = self.model(**features, return_dict=True)
            logits = activation_fct(model_predictions.logits)
            if self.config.num_labels == 1:
                logits = logits.view(-1)
            loss_value = loss_fct(logits, labels)
            loss_value.backward()
            torch.nn.utils.clip_grad_norm_(self.model.parameters(), max_grad_norm)
            optimizer.step()
            optimizer.zero_grad()

            if evaluator is not None and evaluation_steps > 0 and training_steps % evaluation_steps == 0:
                self._eval_during_training(evaluator, output_path, save_best_model, epoch, training_steps, callback)

                self.model.zero_grad()
                self.model.train()
```

실습 코드 2 - Bi-encoder 구현

```
def _load_auto_model(self, model_name_or_path):
    """
    Creates a simple Transformer + Mean Pooling model and returns the modules
    """
    logger.warning("No sentence-transformers model found with name {}. Creating a new one with MEAN pooling.".format(model_name_or_path))
    transformer_model = Transformer(model_name_or_path)
    pooling_model = Pooling(transformer_model.get_word_embedding_dimension(), 'mean')
    return [transformer_model, pooling_model]
```

학습 loop는 어떤 loss를 썼는지만 달라지고 전반적인 흐름은 같기에 생략



실습 코드 2 - Bi-encoder 구현

```python
class MultipleNegativesRankingLoss(nn.Module):

    def __init__(self, model: SentenceTransformer, scale: float = 20.0, similarity_fct = util.cos_sim):
        """
        :param model: SentenceTransformer model
        :param scale: Output of similarity function is multiplied by scale value
        :param similarity_fct: similarity function between sentence embeddings. By default, cos_sim. Can also b
        """
        super(MultipleNegativesRankingLoss, self).__init__()
        self.model = model
        self.scale = scale
        self.similarity_fct = similarity_fct
        self.cross_entropy_loss = nn.CrossEntropyLoss()


    def forward(self, sentence_features: Iterable[Dict[str, Tensor]], labels: Tensor):
        reps = [self.model(sentence_feature)['sentence_embedding'] for sentence_feature in sentence_features]
        embeddings_a = reps[0]
        embeddings_b = torch.cat(reps[1:])

        scores = self.similarity_fct(embeddings_a, embeddings_b) * self.scale
        labels = torch.tensor(range(len(scores)), dtype=torch.long, device=scores.device)  # Example a[i] shoul
        return self.cross_entropy_loss(scores, labels)
```

Ref

- Cross Encoder from scratch
  - https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/ms_marco/train_cross-encoder_scratch.py
  - https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers/cross_encoder/CrossEncoder.py
  - https://www.sbert.net/docs/package_reference/cross_encoder.html?highlight=model%20fit

- Bi encoder from scratch
  - https://github.com/UKPLab/sentence-transformers/blob/master/examples/training/ms_marco/train_bi-encoder_mnrl.py
  - https://www.sbert.net/examples/training/ms_marco/README.html?highlight=multiplenegative#bi-encoder