UNIVERSITAS INDONESIA

APLIKASI *BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS* UNTUK PEMERINGKATAN TEKS BAHASA INDONESIA

SKRIPSI

CARLES OCTAVIANUS
2006568613

FAKULTAS FAKULTAS MATEMATIKA DAN ILMU PENGATAHUAN ALAM
PROGRAM STUDI MATEMATIKA
DEPOK
JANUARI 2024

UNIVERSITAS INDONESIA

APLIKASI *BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS* UNTUK PEMERINGKATAN TEKS BAHASA INDONESIA

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar

Sarjana Sains

CARLES OCTAVIANUS

2006568613

FAKULTAS FAKULTAS MATEMATIKA DAN ILMU PENGATAHUAN ALAM

PROGRAM STUDI MATEMATIKA

DEPOK

JANUARI 2024

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	:	**Carles Octavianus**
NPM	:	**2006568613**
Tanda Tangan	:	
Tanggal	:	**3 Januari 2024**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Carles Octavianus
NPM	:	2006568613
Program Studi	:	Matematika
Judul Skripsi	:	Aplikasi *Bidirectional Encoder Representations from Transformers* untuk Pemeringkatan Teks Bahasa Indonesia

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana pada Program Studi Matematika, Fakultas Fakultas Matematika dan Ilmu Pengatahuan Alam, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing 1	:	Sarini Abdullah S.Si., M.Stats., Ph.D.	( )
Penguji 1	:	Penguji Pertama Anda	( )
Penguji 2	:	Penguji Kedua Anda	( )

Ditetapkan di : Depok
Tanggal :

KATA PENGANTAR

Template ini disediakan untuk orang-orang yang berencana menggunakan LaTeX untuk membuat dokumen tugas akhir.

@todo

Silakan ganti pesan ini dengan pendahuluan kata pengantar Anda.

Ucapan Terima Kasih:

1. Pembimbing.
2. Dosen.
3. Instansi.
4. Orang tua.
5. Sahabat.
6. Teman.

Penulis menyadari bahwa laporan Skripsi ini masih jauh dari sempurna. Oleh karena itu, apabila terdapat kesalahan atau kekurangan dalam laporan ini, Penulis memohon agar kritik dan saran bisa disampaikan langsung melalui *e-mail* `emailanda@mail.id`.

Terkait template ini, gambar lisensi di atas diambil dari `http://creativecommons.org/licenses/by-nc-sa/1.0/deed.en_CA`. Jika ingin mengentahui lebih lengkap mengenai *Creative Common License 1.0 Generic*, silahkan buka `http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode`. Seluruh dokumen yang dibuat dengan menggunakan template ini sepenuhnya menjadi hak milik pembuat dokumen dan bebas didistribusikan sesuai dengan keperluan masing-masing. Lisensi hanya berlaku jika ada orang yang membuat template baru dengan menggunakan template ini sebagai dasarnya.

Penyusun template ingin berterima kasih kepada Andreas Febrian, Lia Sadita, Fahrurrozi Rahman, Andre Tampubolon, dan Erik Dominikus atas kontribusinya dalam template yang menjadi pendahulu template ini. Penyusun template juga ingin mengucapkan terima kasih kepada Azhar Kurnia atas kontribusinya dalam template yang menjadi pendahulu template ini.

Semoga template ini dapat membantu orang-orang yang ingin mencoba menggu-

nakan LaTeX. Semoga template ini juga tidak berhenti disini dengan ada kontribusi dari para penggunanya. Jika Anda memiliki perubahan yang dirasa penting untuk disertakan dalam template, silakan lakukan *fork* repositori Git template ini di `https://gitlab.com/ichlaffterlalu/latex-skripsi-ui-2017`, lalu lakukan *merge request* perubahan Anda terhadap *branch* `master`. Kami berharap agar *template* ini dapat terus diperbarui mengikuti perubahan ketentuan dari pihak Rektorat Universitas Indonesia, dan hal itu tidak mungkin terjadi tanpa kontribusi dari teman-teman sekalian.

Depok, 3 Januari 2024

Carles Octavianus

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	:	Carles Octavianus
NPM	:	2006568613
Program Studi	:	Matematika
Jenis Karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Aplikasi *Bidirectional Encoder Representations from Transformers* untuk Pemeringkatan Teks Bahasa Indonesia

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 3 Januari 2024
Yang menyatakan

(Carles Octavianus)

ABSTRAK

Nama	:	Carles Octavianus
Program Studi	:	Matematika
Judul	:	Aplikasi *Bidirectional Encoder Representations from Transformers* untuk Pemeringkatan Teks Bahasa Indonesia
Pembimbing	:	Sarini Abdullah S.Si., M.Stats., Ph.D.

hello dunia.

Kata kunci:
Keyword satu, kata kunci dua

ABSTRACT

Name	:	Carles Octavianus
Study Program	:	Mathematics
Title	:	Bidirectional Encoder Representations from Transformers Application for Text Ranking in Indonesian
Counselor	:	Sarini Abdullah S.Si., M.Stats., Ph.D.

Abstract content.

Key words:
Keyword one, keyword two

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

src/01-body/bab1 src/01-body/bab2 src/01-body/bab3 src/01-body/bab4 src/01-body/kesimpulan

LAMPIRAN

page2

LAMPIRAN 1: KODE SIMULASI

1. Repositori kode: `https://github.com/carlesoctav/beir-skripsi`
2. Repositori model dan data: `https://huggingface.co/carles-undergrad-thesis`
3. Repositori data (*raw*): `https://drive.google.com/drive/folders/1l_fbqJSn2AR8f-g1QnANl5czm2aLO0rO`

```
from time import time
from beir import util, LoggingHandler
from beir.retrieval import models
from beir.datasets.data_loader import GenericDataLoader
from beir.retrieval.evaluation import EvaluateRetrieval
from beir.retrieval.search.dense import DenseRetrievalExactSearch as DRES

import logging
import pathlib, os
import random
from pyprojroot import here


logging.basicConfig(format='%(asctime)s - %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    level=logging.INFO,
                    handlers=[LoggingHandler()])


corpus_path = str(here('datasets/miracl/corpus.jsonl'))
query_path = str(here('datasets/miracl/queries.jsonl'))
qrels_path = str(here('datasets/miracl/dev.tsv'))

corpus, queries, qrels = GenericDataLoader(
    corpus_file=corpus_path,
    query_file=query_path,
    qrels_file=qrels_path).load_custom()



model =
    DRES(models.SentenceBERT("carles-undergrad-thesis/indobert-mmarco-hardnegs-bm25"),
    batch_size=128)
retriever = EvaluateRetrieval(model, score_function="dot")
```

```
33
34
35 start_time = time()
36 results = retriever.retrieve(corpus, queries)
37 end_time = time()
38 print("Time taken to retrieve: {:.2f} seconds".format(end_time - start_time))
39
40
41 logging.info("Retriever evaluation for k in: {}".format(retriever.k_values))
42 ndcg, _map, recall, precision = retriever.evaluate(qrels, results, retriever.k_values)
43
44 mrr = retriever.evaluate_custom(qrels, results, retriever.k_values, metric="mrr")
45 recall_cap = retriever.evaluate_custom(qrels, results, retriever.k_values,
       metric="r_cap")
46 hole = retriever.evaluate_custom(qrels, results, retriever.k_values, metric="hole")
47
48 top_k = 10
49
50 query_id, ranking_scores = random.choice(list(results.items()))
51 scores_sorted = sorted(ranking_scores.items(), key=lambda item: item[1], reverse=True)
52 logging.info("Query : %s\n" % queries[query_id])
53
54 for rank in range(top_k):
55     doc_id = scores_sorted[rank][0]
56     logging.info("Rank %d: %s [%s] - %s\n" % (rank+1, doc_id,
       corpus[doc_id].get("title"), corpus[doc_id].get("text")))
```

Kode 1: Kode untuk mengevaluasi BERT_{DOT}

```
from beir import util, LoggingHandler
from beir.datasets.data_loader import GenericDataLoader
from beir.retrieval.evaluation import EvaluateRetrieval
from beir.retrieval.search.lexical import BM25Search as BM25
from beir.reranking.models import CrossEncoder
from beir.reranking import Rerank

import pathlib, os
import logging
import random
from pyprojroot import here


logging.basicConfig(format='%(asctime)s - %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    level=logging.INFO,
                    handlers=[LoggingHandler()])


corpus_path = str(here('datasets/mrtydi/indonesian/corpus.jsonl'))
query_path = str(here('datasets/mrtydi/indonesian/queries.jsonl'))
qrels_path = str(here('datasets/mrtydi/indonesian/qrels/dev.tsv'))

corpus, queries, qrels = GenericDataLoader(
    corpus_file=corpus_path,
    query_file=query_path,
    qrels_file=qrels_path).load_custom()

#### Provide parameters for Elasticsearch
hostname = "localhost" #localhost
index_name = "mrtydi-indo" # trec-covid
initialize = True # False
language = "indonesian"

model = BM25(index_name=index_name, hostname=hostname,
    initialize=initialize,language=language)
retriever = EvaluateRetrieval(model)

results = retriever.retrieve(corpus, queries)

cross_encoder_model =
    CrossEncoder('carles-undergrad-thesis/indobert-crossencoder-mmarco', max_length =
    512)

reranker = Rerank(cross_encoder_model, batch_size=256)

rerank_results = reranker.rerank(corpus, queries, results, top_k=100)

ndcg, _map, recall, precision = EvaluateRetrieval.evaluate(qrels, rerank_results,
    retriever.k_values)
```

```
47
48 mrr = retriever.evaluate_custom(qrels, rerank_results, retriever.k_values, metric="mrr")
49 recall_cap = retriever.evaluate_custom(qrels, rerank_results, retriever.k_values,
       metric="r_cap")
50 hole = retriever.evaluate_custom(qrels, rerank_results, retriever.k_values,
       metric="hole")
51
52 top_k = 10
53
54 query_id, ranking_scores = random.choice(list(rerank_results.items()))
55 scores_sorted = sorted(ranking_scores.items(), key=lambda item: item[1], reverse=True)
56 logging.info("Query : %s\n" % queries[query_id])
57
58 for rank in range(top_k):
59     doc_id = scores_sorted[rank][0]
60     logging.info("Rank %d: %s [%s] - %s\n" % (rank+1, doc_id,
       corpus[doc_id].get("title"), corpus[doc_id].get("text")))
```

Kode 2: Kode untuk mengevaluasi BERT_{CAT}

```python
from beir import util, LoggingHandler
from beir.datasets.data_loader import GenericDataLoader
from beir.retrieval.evaluation import EvaluateRetrieval
from beir.retrieval.search.lexical import BM25Search as BM25

import pathlib, os, random
import logging
from pyprojroot import here

logging.basicConfig(format='%(asctime)s - %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    level=logging.INFO,
                    handlers=[LoggingHandler()])



corpus_path = str(here('datasets/miracl/corpus.jsonl'))
query_path = str(here('datasets/miracl/queries.jsonl'))
qrels_path = str(here('datasets/miracl/dev.tsv'))

corpus, queries, qrels = GenericDataLoader(
    corpus_file=corpus_path,
    query_file=query_path,
    qrels_file=qrels_path).load_custom()


hostname = "localhost"
index_name = "miracl-indo"

initialize = True

language = "indonesian"

number_of_shards = 1
model = BM25(index_name=index_name, hostname=hostname, language=language,
    initialize=initialize, number_of_shards=number_of_shards)
retriever = EvaluateRetrieval(model)

results = retriever.retrieve(corpus, queries)

logging.info("Retriever evaluation for k in: {}".format(retriever.k_values))
ndcg, _map, recall, precision = retriever.evaluate(qrels, results, retriever.k_values)

mrr = retriever.evaluate_custom(qrels, results, retriever.k_values, metric="mrr")
recall_cap = retriever.evaluate_custom(qrels, results, retriever.k_values,
    metric="r_cap")
hole = retriever.evaluate_custom(qrels, results, retriever.k_values, metric="hole")


query_id, scores_dict = random.choice(list(results.items()))
```

```
logging.info("Query : %s\n" % queries[query_id])

scores = sorted(scores_dict.items(), key=lambda item: item[1], reverse=True)
for rank in range(10):
    doc_id = scores[rank][0]
    logging.info("Doc %d: %s [%s] - %s\n" % (rank+1, doc_id,
    corpus[doc_id].get("title"), corpus[doc_id].get("text")))
```

Kode 3: Kode untuk mengevaluasi BM25

```python
import torch
import argparse
from datasets import load_dataset
from transformers import (
    Trainer,
    AutoModelForSequenceClassification,
    AutoTokenizer,
    AutoConfig,
    TrainingArguments,
)


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--output_dir", required=True)
    parser.add_argument(
        "--model", default="indolem/indobert-base-uncased", type=str, required=False
    )

    parser.add_argument("--learning_rate", default=2e-5, type=float, required=False)
    parser.add_argument("--batch_size", default=16, type=int, required=False)
    parser.add_argument("--max_samples", default=250_000, type=int, required=False)
    args = parser.parse_args()

    tokenizer = AutoTokenizer.from_pretrained(args.model)
    config = AutoConfig.from_pretrained(args.model)
    config.num_labels = 1
    config.problem_type = "multi_label_classification"
    model = AutoModelForSequenceClassification.from_pretrained(
        args.model, config=config
    )

    dataset = load_dataset("carles-undergrad-thesis/indo-mmarco-500k")["train"]
    if args.max_samples:
        dataset = dataset.select(range(args.max_samples))

    def split_examples(batch):
        queries = []
        passages = []
        labels = []
        for label in ["positive", "negative"]:
            for (query, passage) in zip(batch["query"], batch[label]):
                queries.append(query)
                passages.append(passage)
                labels.append(int(label == "positive"))
        return {"query": queries, "passage": passages, "label": labels}

    dataset = dataset.map(
        split_examples, batched=True, remove_columns=["positive", "negative"]
    )
```

```
51
52      def tokenize(batch):
53          tokenized = tokenizer(
54              batch["query"],
55              batch["passage"],
56              padding=True,
57              truncation="only_second",
58              max_length=512,
59          )
60          tokenized["labels"] = [[float(label)] for label in batch["label"]]
61          return tokenized
62
63      dataset = dataset.map(
64          tokenize, batched=True, remove_columns=["query", "passage", "label"]
65      )
66      dataset.set_format("torch")
67
68      print(len(dataset))
69
70      training_args = TrainingArguments(
71          output_dir=args.output_dir,
72          # fp16=True,
73          # fp16_backend="amp",
74          per_device_train_batch_size=args.batch_size,
75          logging_steps=10_000,
76          warmup_steps=0.1*len(dataset)*args.batch_size,
77          save_total_limit=1,
78          num_train_epochs=1,
79
80      )
81      trainer = Trainer(
82          model,
83          training_args,
84          train_dataset=dataset,
85          tokenizer=tokenizer,
86      )
87
88      train_result = trainer.train()
89      trainer.save_model()
```

Kode 4: Kode untuk melatih IndoBERT$_{\text{CAT}}$

```python
from sentence_transformers import losses, models, SentenceTransformer
from beir import util, LoggingHandler
from beir.datasets.data_loader import GenericDataLoader
from beir.retrieval.train import TrainRetriever
import pathlib, os
import logging
from pyprojroot import here
import torch

logging.basicConfig(format='%(asctime)s - %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    level=logging.INFO,
                    handlers=[LoggingHandler()])

dataset = "mmarco"
corpus_path = str(here('datasets/mmarco/indonesian/corpus.jsonl'))
query_path = str(here('datasets/mmarco/indonesian/queries.jsonl'))
qrels_path = str(here('datasets/mmarco/indonesian/qrels/dev.tsv'))

corpus, queries, qrels = GenericDataLoader(
    corpus_file=corpus_path,
    query_file=query_path,
    qrels_file=qrels_path).load_custom()

model_name = "indolem/indobert-base-uncased"
word_embedding_model = models.Transformer(model_name, max_seq_length=256)
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension(),
    pooling_mode = "cls")
model = SentenceTransformer(modules=[word_embedding_model, pooling_model])


retriever = TrainRetriever(model=model, batch_size=32)

train_samples = retriever.load_train(corpus, queries, qrels)
train_dataloader = retriever.prepare_train(train_samples, shuffle=True)

train_loss = losses.MultipleNegativesRankingLoss(model=retriever.model, scale=1.0,
    similarity_fct=util.dot_score)

ir_evaluator = retriever.load_dummy_evaluator()

model_save_path = os.path.join(pathlib.Path(__file__).parent.absolute(), "output",
    "{}-v1-{}".format(model_name, dataset))
os.makedirs(model_save_path, exist_ok=True)


num_epochs = 5
evaluation_steps = 1_000_000
warmup_steps = int(len(train_samples) * num_epochs / retriever.batch_size * 0.1)

```

```
retriever.fit(train_objectives=[(train_dataloader, train_loss)],
                evaluator=ir_evaluator,
                epochs=num_epochs,
                output_path=model_save_path,
                warmup_steps=warmup_steps,
                evaluation_steps=evaluation_steps,
                use_amp=True)


# model.save_to_hub("st-indobert-mmarco-v1", "carles-undergrad-thesis")
```

Kode 5: Kode untuk melatih IndoBERT$_{\text{DOT}}$

```python
import tensorflow as tf
from transformers import TFXLMRobertaModel, AutoTokenizer, TFAutoModel
from datasets import load_dataset
from datetime import datetime
import logging
from pyprojroot.here import here
import os

class mean_pooling_layer(tf.keras.layers.Layer):
    def __init__(self):
        super(mean_pooling_layer, self).__init__()

    def call(self, inputs):
        token_embeddings = inputs[0]
        attention_mask = inputs[1]
        input_mask_expanded = tf.cast(
            tf.broadcast_to(tf.expand_dims(attention_mask, -1),
    tf.shape(token_embeddings)),
            tf.float32
        )

        embeddings = tf.math.reduce_sum(token_embeddings * input_mask_expanded, axis=1)
    / tf.clip_by_value(tf.math.reduce_sum(input_mask_expanded, axis=1), 1e-9,
    tf.float32.max)
        return embeddings


    def get_config(self):
        config = super(mean_pooling_layer, self).get_config()
        return config


def create_model():
    base_student_model =
    TFAutoModel.from_pretrained("distilbert-base-multilingual-cased",from_pt=True)
    input_ids_en = tf.keras.layers.Input(shape=(256,),name='input_ids_en',
    dtype=tf.int32)
    attention_mask_en = tf.keras.layers.Input(shape=(256,), name='attention_mask_en',
    dtype=tf.int32)
    input_ids_id = tf.keras.layers.Input(shape=(256,),name='input_ids_id',
    dtype=tf.int32)
    attention_mask_id = tf.keras.layers.Input(shape=(256,), name='attention_mask_id',
    dtype=tf.int32)
    mean_pooling = mean_pooling_layer()

    output_en = base_student_model.distilbert(input_ids_en,
    attention_mask=attention_mask_en).last_hidden_state[:,0,:]
    output_id = base_student_model.distilbert(input_ids_id,
    attention_mask=attention_mask_id).last_hidden_state[:,0,:]

```

```
    student_model = tf.keras.Model(inputs=[input_ids_en, attention_mask_en,
    input_ids_id, attention_mask_id], outputs=[output_en, output_id])
    print(student_model.summary())
    return student_model

class sentence_translation_metric(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epoch,logs):
        embeddings_en, embeddings_id = self.model.predict(val_dataset, verbose=1)
        # get the embeddings
        # compute the cosine similarity between the two
        #normalize the embeddings
        similarity_matrix = tf.matmul(embeddings_en, embeddings_id, transpose_b=True)
        print(f"==>> similarity_matrix: {similarity_matrix}")
        # get the mean similarity
        correct_en_id = 0
        for i in range(similarity_matrix.shape[0]):
            if tf.math.argmax(similarity_matrix[i]) == i:
                correct_en_id += 1

        similarity_matrix_T = tf.transpose(similarity_matrix)
        correct_id_en = 0
        for i in range(similarity_matrix_T.shape[0]):
            if tf.math.argmax(similarity_matrix_T[i]) == i:
                correct_id_en += 1

        acc_en_id = correct_en_id / similarity_matrix.shape[0]
        acc_id_en = correct_id_en / similarity_matrix_T.shape[0]
        avg_acc = (acc_en_id + acc_id_en) / 2
        print(f"translation accuracy from english to indonesian = {acc_en_id}")
        print(f"translation accuracy from indonesian to english = {acc_id_en}")
        print(f"average translation accuracy = {avg_acc}")

        logs["val_acc_en_id"] = acc_en_id
        logs["val_acc_id_en"] = acc_id_en
        logs["val_avg_acc"] = avg_acc


class ConstantScheduler(tf.keras.optimizers.schedules.LearningRateSchedule):
  def __init__(self, max_lr, warmup_steps=5000):
    super().__init__()
    self.max_lr = tf.cast(max_lr, tf.float32)
    self.warmup_steps = warmup_steps

  def __call__(self, step):
    step = tf.cast(step, tf.float32)
    condition  = tf.cond(step < self.warmup_steps, lambda: step / self.warmup_steps,
    lambda: 1.0)
    return self.max_lr * condition


```

```
if __name__ == "__main__":
    num_data = 0
    dataset = load_dataset("carles-undergrad-thesis/en-id-parallel-sentences-embedding")

    dataset_1 = dataset["train"]

    # for split in dataset:
    #     dataset_1 = concatenate_datasets([dataset_1, dataset[split]])


    batch_size = 512
    dataset = dataset_1.train_test_split(test_size=0.01, shuffle=True)
    train_dataset = dataset["train"]
    val_dataset = dataset["test"]

    print(f"==>> val_dataset.shape: {val_dataset.shape}")

    train_dataset = train_dataset.to_tf_dataset(
        columns=["input_ids_en", "attention_mask_en", "input_ids_id",
    "attention_mask_id"],
        label_cols="target_embedding",
        batch_size=batch_size,
    ).unbatch()

    val_dataset = val_dataset.to_tf_dataset(
        columns=["input_ids_en", "attention_mask_en", "input_ids_id",
    "attention_mask_id"],
        label_cols="target_embedding",
        batch_size=batch_size,
    ).unbatch()

    #check feature
    print(train_dataset.element_spec)
    print(val_dataset.element_spec)

    train_dataset = train_dataset.batch(batch_size, drop_remainder=True).cache()
    val_dataset = val_dataset.batch(batch_size, drop_remainder=True).cache()


    warm_up_steps = 5_000_000 / batch_size *0.1
    learning_rate = ConstantScheduler(2e-5, warmup_steps= warm_up_steps)

    optimizer = tf.keras.optimizers.Adam(learning_rate, beta_1=0.9, beta_2=0.98,
                                         epsilon=1e-9)


    loss = tf.keras.losses.MeanSquaredError()

    date_time = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")

```

```
output_path = here(f"disk/model/{date_time}/model.h5")

if not os.path.exists(here("disk/model")):
    os.makedirs(here("disk/model"))

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
                filepath = output_path,
                save_weights_only = True,
                monitor = "val_avg_acc",
                mode = 'auto',
                verbose = 1,
                save_best_only = True,
                initial_value_threshold = 0.5,
                )


# tensor_board  = tf.keras.callbacks.TensorBoard(
#                   log_dir = "gs://dicoding-capstone/output/logs/"+date_time
# )

if not os.path.exists(here("disk/performance_logs")):
    os.makedirs(here("disk/performance_logs"))


csv_logger = tf.keras.callbacks.CSVLogger(
                filename = here(f"disk/performance_logs/log-{date_time}.csv"),
                separator = ",",
                append = False
)


callbacks = [sentence_translation_metric(), model_checkpoint, csv_logger]


cluster_resolver = tf.distribute.cluster_resolver.TPUClusterResolver("local")
tf.config.experimental_connect_to_cluster(cluster_resolver)
tf.tpu.experimental.initialize_tpu_system(cluster_resolver)
strategy = tf.distribute.TPUStrategy(cluster_resolver)

with strategy.scope():
    student_model = create_model()
    student_model.compile(optimizer=optimizer, loss=loss)

student_model.fit(train_dataset, epochs=5, validation_data=val_dataset,
callbacks=callbacks)

last_epoch_save = here(f"disk/model/last_epoch/{date_time}.h5")

if not os.path.exists(here("disk/model/last_epoch")):
    os.makedirs(here("disk/model/last_epoch"))
```

```
186
187     student_model.save_weights(last_epoch_save)
```

Kode 6: Kode untuk melatih IndoBERT$_{\text{DOTKD}}$

LAMPIRAN 2: *OUTPUT* DARI SIMULASI

Terlampir contoh *output* pengevaluasian dari suatu model, Informasi lebih lanjut dapat dilihat di `https://github.com/carlesoctav/beir-skripsi`.

```
100%|██████████████████████████████████████████████████████████████████████████|
8841823/8841823 [00:43<00:00, 2045[107/1908]
2023-10-20 10:45:58 - Loaded 8841823 DEV Documents.
2023-10-20 10:45:58 - Doc Example: {'text': 'Kehadiran komunikasi di
tengah pikiran ilmiah sama pentingnya dengan kebe
rhasilan Proyek Manhattan seperti kecerdasan ilmiah. Kehadiran komunikasi
di tengah pikiran ilmiah sama pentingnya den
gan keberhasilan Proyek Manhattan seperti kecerdasan ilmiah. Satu -
satunya awan yang menggantung di atas prestasi men
gesankan para peneliti dan insinyur atom adalah apa sebenarnya tujuan
kesuksesan mereka; ratusan ribu nyawa yang tidak
 bersalah dilenyapkan.', 'title': ''}
2023-10-20 10:45:58 - Loading Queries...
2023-10-20 10:46:01 - Loaded 6980 DEV Queries.
2023-10-20 10:46:01 - Query Example: Berapa tahun William Bradford
melayani sebagai gubernur koloni Plymouth?
2023-10-20 10:46:01 - Activating Elasticsearch....
2023-10-20 10:46:01 - Elastic Search Credentials: {'hostname':
'localhost', 'index_name': 'mmarco-indo', 'keys': {'tit
le': 'title', 'body': 'txt'}, 'timeout': 100, 'retry_on_timeout': True,
'maxsize': 24, 'number_of_shards': 'default',
'language': 'indonesian'}
que:   0%|
| 0/55 [00:00<?, ?it/s]
/usr/local/lib/python3.8/dist-
packages/elasticsearch/connection/base.py:190:
ElasticsearchDeprecationWarning: Elastics
earch built-in security features are not enabled. Without authentication,
your cluster could be accessible to anyone.
See
https://www.elastic.co/guide/en/elasticsearch/reference/7.17/security-
minimal-setup.html to enable security.
  warnings.warn(message, category=ElasticsearchDeprecationWarning)
que:
100%|████████████████████████████████████████████████████████████████████████
████████| 55/55 [01:01<00:00,  1.12s/it]
2023-10-20 10:47:05 - Use pytorch device: cuda
2023-10-20 10:47:45 - Starting To Rerank Top-1000....
Batches:   2%|▉
| 475/26727 [02:21<2:31:56,  2.88it/s]
Batches:   2%|▉
| 476/26727 [02:21<2:20:09,  3.12it/s]
Batches:
100%|██████████████████████████████████████████████████████████████████████████|
26727/26727 [2:11:06<00:00,  3.40it/s]
2023-10-20 13:01:31 - For evaluation, we ignore identical query and
document ids (default), please explicitly set ``ig

2023-10-20 10:47:05 - Use pytorch device: cuda
[87/1908]
2023-10-20 10:47:45 - Starting To Rerank Top-1000....
Batches:   2%|▉
| 475/26727 [02:21<2:31:56,  2.88it/s]
```

```
Batches:   2%|▉
| 476/26727 [02:21<2:20:09,  3.12it/s]
Batches:
100%|████████████████████████████████████████████████████████████████████████████|
26727/26727 [2:11:06<00:00,  3.40it/s]
2023-10-20 13:01:31 - For evaluation, we ignore identical query and
document ids (default), please explicitly set ``ig
nore_identical_ids=False`` to ignore this.
2023-10-20 13:01:36 -

2023-10-20 13:01:36 - NDCG@1: 0.1052
2023-10-20 13:01:36 - NDCG@3: 0.1667
2023-10-20 13:01:36 - NDCG@5: 0.1940
2023-10-20 13:01:36 - NDCG@10: 0.2243
2023-10-20 13:01:36 - NDCG@100: 0.2707
2023-10-20 13:01:36 - NDCG@1000: 0.2786
2023-10-20 13:01:36 -

2023-10-20 13:01:36 - MAP@1: 0.1019
2023-10-20 13:01:36 - MAP@3: 0.1500
2023-10-20 13:01:36 - MAP@5: 0.1652
2023-10-20 13:01:36 - MAP@10: 0.1778
2023-10-20 13:01:36 - MAP@100: 0.1874
2023-10-20 13:01:36 - MAP@1000: 0.1878
2023-10-20 13:01:36 -

2023-10-20 13:01:36 - Recall@1: 0.1019

2023-10-20 13:01:36 - Recall@1: 0.1019
[62/1908]
2023-10-20 13:01:36 - Recall@3: 0.2105
2023-10-20 13:01:36 - Recall@5: 0.2765
2023-10-20 13:01:36 - Recall@10: 0.3688
2023-10-20 13:01:36 - Recall@100: 0.5829
2023-10-20 13:01:36 - Recall@1000: 0.6421
2023-10-20 13:01:36 -

2023-10-20 13:01:36 - P@1: 0.1052
2023-10-20 13:01:36 - P@3: 0.0731
2023-10-20 13:01:36 - P@5: 0.0576
2023-10-20 13:01:36 - P@10: 0.0386
2023-10-20 13:01:36 - P@100: 0.0062
2023-10-20 13:01:36 - P@1000: 0.0007
2023-10-20 13:01:36 -

2023-10-20 13:01:39 - MRR@1: 0.1047
2023-10-20 13:01:39 - MRR@3: 0.1538
2023-10-20 13:01:39 - MRR@5: 0.1689
2023-10-20 13:01:39 - MRR@10: 0.1814
2023-10-20 13:01:39 - MRR@100: 0.1908
2023-10-20 13:01:39 - MRR@1000: 0.1912
2023-10-20 13:01:40 -

2023-10-20 13:01:43 - R_cap@1: 0.1047
```

2023-10-20 13:01:43 - R_cap@3: 0.2101

2023-10-20 13:01:43 - R_cap@5: 0.2760
[36/1908]
2023-10-20 13:01:43 - R_cap@10: 0.3680
2023-10-20 13:01:43 - R_cap@100: 0.5817
2023-10-20 13:01:43 - R_cap@1000: 0.6408
2023-10-20 13:01:43 -

2023-10-20 13:01:47 - Hole@1: 0.8920
2023-10-20 13:01:47 - Hole@3: 0.9235
2023-10-20 13:01:47 - Hole@5: 0.9386
2023-10-20 13:01:47 - Hole@10: 0.9564
2023-10-20 13:01:47 - Hole@100: 0.9816
2023-10-20 13:01:47 - Hole@1000: 0.9786
2023-10-20 13:01:47 - Query : Apa akar dari semua kejahatan

2023-10-20 13:01:47 - Rank 1: 7213594 [] - Secara pribadi, saya percaya akar dari semua kejahatan adalah keegoisan dan
 keegoisan diungkapkan dalam segala cara. Sebaliknya adalah cinta yang mengorbankan diri sendiri. Seseorang juga dapat
 berbicara tentang akar segala kejahatan dalam hal dusta. ♪ And to my mind the root of evil is to deny Godvine reveal
♪

2023-10-20 13:01:47 - Rank 2: 7213591 [] - Akar dari semua setan berasal dari manusia. Manusia itu sendiri adalah akar
 dari semua kejahatan. Sekarang saya tidak mencoba untuk mengatakan bahwa setiap pria, wanita, dan anak secara inheren
 jahat tetapi tidak dapat dibantah bahwa bahkan tidak akan ada konsep kejahatan jika bukan untuk manusia.

2023-10-20 13:01:47 - Rank 3: 5451019 [] - """Ya, karena cinta akan uang adalah akar dari semua kejahatan,"" artinya,
bukan berarti setiap kejahatan harus berasal dari ""cinta uang,"" tetapi bahwa tidak ada yang dapat dibayangkan kejaha
tan yang dapat terjadi pada anak laki-laki dan perempuan dari laki-laki yang tidak mungkin musim semi dari keserakahan

2023-10-20 13:01:47 - Rank 3: 5451019 [] - """Ya, karena cinta akan uang adalah akar dari semua kejahatan,"" [13/1908]
bukan berarti setiap kejahatan harus berasal dari ""cinta uang,"" tetapi bahwa tidak ada yang dapat dibayangkan kejaha
tan yang dapat terjadi pada anak laki-laki dan perempuan dari laki-laki yang tidak mungkin musim semi dari keserakahan
 cinta emas dan kekayaan."

2023-10-20 13:01:47 - Rank 4: 8380731 [] - Contoh Kalimat & Contoh. 1 Simone Weil: Kejahatan adalah akar misteri, rasa
 sakit adalah akar dari pengetahuan. Jadi menurutmu uang adalah akar dari semua kejahatan. Apakah Anda pernah bertanya
 apa akar dari semua uang. 3 Med Yones: Dalam tradisi agama, cinta akan uang adalah akar segala kejahatan. Dalam ekono
mi, kelangkaan adalah akar dari semua kejahatan.

2023-10-20 13:01:47 - Rank 5: 3611440 [] - Simone Weil: Kejahatan adalah akar misteri, rasa sakit adalah akar dari pen
getahuan. Jadi menurutmu uang adalah akar dari semua kejahatan. Apakah Anda pernah bertanya apa akar dari semua uang.
Med Yones: Dalam tradisi agama, cinta akan uang adalah akar segala kejahatan.

2023-10-20 13:01:47 - Rank 6: 1599590 [] - "Ringkasan Cepat. Kata akar Latin yang berarti ""buruk"" atau ""jahat."" Ak
ar ini adalah kata yang berasal dari banyak kosakata bahasa Inggris, termasuk mal, mal treat, dan mal ice. Anda dapat
mengingat bahwa mal berarti ""buruk"" melalui fungsi mal, atau ""buruk"" bekerja bagian, dan bahwa itu berarti ""jahat
"" melalui es mal, atau sengaja ""jahat"" dilakukan untuk yang lain."

2023-10-20 13:01:47 - Rank 7: 4715621 [] - akar akar akar: akar utama tanaman, yang langsung mengalir ke bumi hingga k
edalaman yang cukup dalam tanpa membelah. Akar kejahatan, akar akar dari yang berkembang kejahatan masyarakat modern,
adalah ide laba. Salah satunya adalah rasa hormat terhadap otoritas, yang hilang adalah akar dari Bolshevism.

2023-10-20 13:01:47 - Rank 8: 7867039 [] - Apa akar segala macam kejahatan? Jawaban 1 Ada kepercayaan bahwa uang adala
h akar dari semua jahat Jawaban 2 1 Timotius 6:10 - Karena cinta uang adalah akar dari segala macam kejahatan, untuk y
ang beberapa ha... telah menyimpang dari iman dalam keserakahan mereka, dan menusuk diri melalui dengan banyak penderi

2023-10-20 13:01:47 - Rank 9: 5451020 [] - 10. Cinta akan uang bukan uang itu sendiri, tapi cinta itu sendiri, keingin
an untuk menjadi kaya (1Ti 6:9) adalah akar (Ellicott dan Middleton: bukan sebagai English Version, 'akar') dari semua
 kejahatan. (So the Greek plural). Orang yang paling kaya mungkin tidak kaya dalam arti yang buruk; yang termiskin mun
gkin ingin menjadi begitu (Ps 62:10).

2023-10-20 13:01:47 - Rank 10: 583029 [] - "Ringkasan Cepat. Kata akar Latin yang berarti ""buruk"" atau ""jahat."" Ak
ar ini adalah asal usul dari banyak kosakata bahasa Inggris, termasuk kata - kata yang salah bentuk, perlakuan yang sa
lah, dan niat jahat. Anda dapat mengingat bahwa mal berarti ""buruk"" melalui kerusakan, atau ""buruk"" bagian kerja,
dan bahwa itu berarti ""jahat"" melalui kedengkian, atau sengaja ""jahat"" dilakukan untuk yang lain."