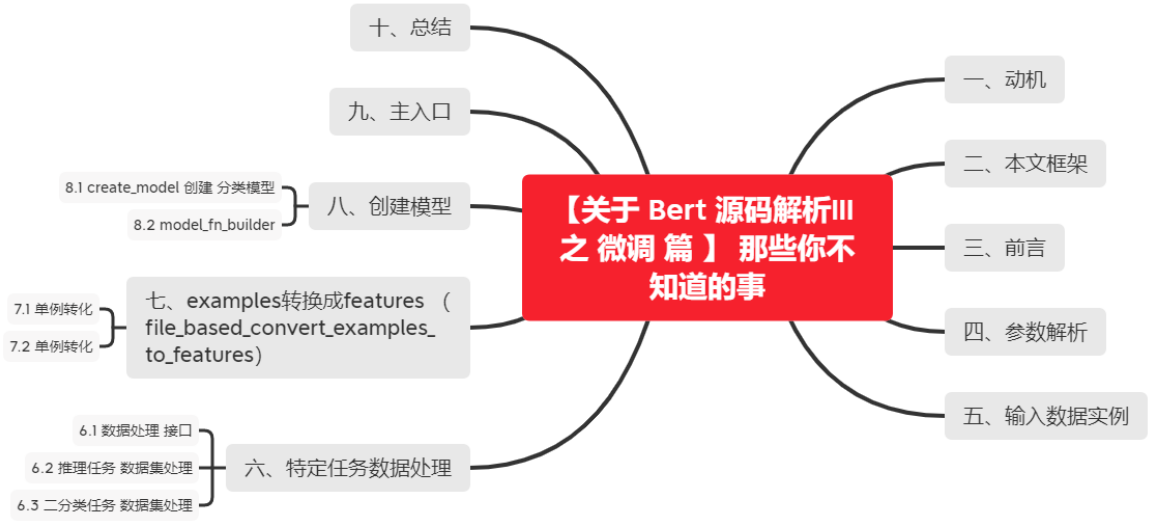
【关于 Bert 源码解析III 之 微调篇 】 那些你不知道的事



一、前言

本文 主要 解读 Bert 模型的 微调 模块代码：

- run_classifier.py：主要用于 文本分类 任务的微调

二、参数解析

```
flags = tf.flags
FLAGS = flags.FLAGS
'''
  必要参数
'''
# 数据地址
flags.DEFINE_string(
    "data_dir", None,
    "The input data dir. Should contain the .tsv files (or other data files) "
    "for the task.")
# Bert 配置文件地址
flags.DEFINE_string(
    "bert_config_file", None,
    "The config json file corresponding to the pre-trained BERT model. "
    "This specifies the model architecture.")
# 训练任务
flags.DEFINE_string("task_name", None, "The name of the task to train.")
# Bert 词库
flags.DEFINE_string("vocab_file", None,
                    "The vocabulary file that the BERT model was trained on.")
# 训练输出 地址
flags.DEFINE_string(
```

```
    "output_dir", None,
    "The output directory where the model checkpoints will be written.")
'''
  其他参数
'''
# 预训练 Bert 模型
flags.DEFINE_string(
    "init_checkpoint", None,
    "Initial checkpoint (usually from a pre-trained BERT model).")
# 是否小写
flags.DEFINE_bool(
    "do_lower_case", True,
    "Whether to lower case the input text. Should be True for uncased "
    "models and False for cased models.")
# 指定WordPiece tokenization 之后的sequence的最大长度，要求小于等于预训练模型的最大sequence长度。当输入的数据长度小于max_seq_length时用0补齐，如果长度大于max_seq_length则truncate处理；
flags.DEFINE_integer(
    "max_seq_length", 128,
    "The maximum total input sequence length after WordPiece tokenization. "
    "Sequences longer than this will be truncated, and sequences shorter "
    "than this will be padded.")
# 训练
flags.DEFINE_bool("do_train", False, "Whether to run training.")
# 验证
flags.DEFINE_bool("do_eval", False, "Whether to run eval on the dev set.")
# 预测
flags.DEFINE_bool(
    "do_predict", False,
    "Whether to run the model in inference mode on the test set.")
# 训练 Batch 大小
flags.DEFINE_integer("train_batch_size", 32, "Total batch size for training.")
# 评测 Batch 大小
flags.DEFINE_integer("eval_batch_size", 8, "Total batch size for eval.")
# 预测 Batch 大小
flags.DEFINE_integer("predict_batch_size", 8, "Total batch size for predict.")
# 学习率
flags.DEFINE_float("learning_rate", 5e-5, "The initial learning rate for Adam.")
# 训练 epochs
flags.DEFINE_float("num_train_epochs", 3.0,
                   "Total number of training epochs to perform.")
# 进行线性学习率预热的训练比例。
flags.DEFINE_float(
    "warmup_proportion", 0.1,
    "Proportion of training to perform linear learning rate warmup for. "
    "E.g., 0.1 = 10% of training.")
# 保存模型 步长
flags.DEFINE_integer("save_checkpoints_steps", 1000,
                     "How often to save the model checkpoint.")
# 每个 estimator call 调用中要执行多少步
flags.DEFINE_integer("iterations_per_loop", 1000,
                     "How many steps to make in each estimator call.")
# 是否 使用 TPU
flags.DEFINE_bool("use_tpu", False, "Whether to use TPU or GPU/CPU.")
# TPU 名称
```

```
tf.flags.DEFINE_string(
    "tpu_name", None,
    "The Cloud TPU to use for training. This should be either the name "
    "used when creating the Cloud TPU, or a grpc://ip.address.of.tpu:8470 "
    "url.")

tf.flags.DEFINE_string(
    "tpu_zone", None,
    "[Optional] GCE zone where the Cloud TPU is located in. If not "
    "specified, we will attempt to automatically detect the GCE project from "
    "metadata.")

tf.flags.DEFINE_string(
    "gcp_project", None,
    "[Optional] Project name for the Cloud TPU-enabled project. If not "
    "specified, we will attempt to automatically detect the GCE project from "
    "metadata.")

tf.flags.DEFINE_string("master", None, "[Optional] TensorFlow master URL.")

flags.DEFINE_integer(
    "num_tpu_cores", 8,
    "Only used if `use_tpu` is True. Total number of TPU cores to use.")
```

三、输入数据实例

```
class InputExample(object):
  """A single training/test example for simple sequence classification."""

  def __init__(self, guid, text_a, text_b=None, label=None):
    """Constructs a InputExample.

    Args:
      guid: 实例 唯一 id
      text_a: string. 第一个序列的未标记文本。 对于单序列任务，仅必须指定此序列。
      text_b: (Optional) string. 第二个序列的未标记文本。 仅必须为序列对任务指定。
      label: (Optional) string. 实例的标签。 应该为train和dev实例指定此名称，但不为测试实例指定，如果是test数据集则label统一为0。
    """
    self.guid = guid
    self.text_a = text_a
    self.text_b = text_b
    self.label = label
```

四、特定任务数据处理

4.1 数据处理 接口

- 作用：数据预处理 接口

```
class DataProcessor(object):
  """Base class for data converters for sequence classification data sets."""
```

```
  def get_train_examples(self, data_dir):
    """Gets a collection of `InputExample`s for the train set."""
    raise NotImplementedError()

  def get_dev_examples(self, data_dir):
    """Gets a collection of `InputExample`s for the dev set."""
    raise NotImplementedError()

  def get_test_examples(self, data_dir):
    """Gets a collection of `InputExample`s for prediction."""
    raise NotImplementedError()

  def get_labels(self):
    """Gets the list of labels for this data set."""
    raise NotImplementedError()

  @classmethod
  def _read_tsv(cls, input_file, quotechar=None):
    """Reads a tab separated value file."""
    with tf.gfile.Open(input_file, "r") as f:
      reader = csv.reader(f, delimiter="\t", quotechar=quotechar)
      lines = []
      for line in reader:
        lines.append(line)
      return lines
```

4.2 推理任务 数据集处理

- 目标：读取两句话，并判定两者间的关系是否为“蕴含”（Entailment）、“矛盾”（Contradict）或“中性”（Neutral）
- 具体任务数据介绍：FAIR重磅发布大规模语料库XNLI：支持15种语言，解决跨语言理解难题

```
class XnliProcessor(DataProcessor):
  """Processor for the XNLI data set."""

  def __init__(self):
    self.language = "zh"

  def get_train_examples(self, data_dir):
    """See base class."""
    lines = self._read_tsv(
        os.path.join(data_dir, "multinli",
                     "multinli.train.%s.tsv" % self.language))
    examples = []
    for (i, line) in enumerate(lines):
      if i == 0:
        continue
      guid = "train-%d" % (i)
      text_a = tokenization.convert_to_unicode(line[0])
      text_b = tokenization.convert_to_unicode(line[1])
      label = tokenization.convert_to_unicode(line[2])
      if label == tokenization.convert_to_unicode("contradictory"):
        label = tokenization.convert_to_unicode("contradiction")
      examples.append(
```

```
          InputExample(guid=guid, text_a=text_a, text_b=text_b, label=label))
    return examples

  def get_dev_examples(self, data_dir):
    """See base class."""
    lines = self._read_tsv(os.path.join(data_dir, "xnli.dev.tsv"))
    examples = []
    for (i, line) in enumerate(lines):
      if i == 0:
        continue
      guid = "dev-%d" % (i)
      language = tokenization.convert_to_unicode(line[0])
      if language != tokenization.convert_to_unicode(self.language):
        continue
      text_a = tokenization.convert_to_unicode(line[6])
      text_b = tokenization.convert_to_unicode(line[7])
      label = tokenization.convert_to_unicode(line[1])
      examples.append(
          InputExample(guid=guid, text_a=text_a, text_b=text_b, label=label))
    return examples

  def get_labels(self):
    """See base class."""
    return ["contradiction", "entailment", "neutral"]
```

4.3 二分类任务 数据集处理

```
class ColaProcessor(DataProcessor):
  """Processor for the CoLA data set (GLUE version)."""

  def get_train_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "train.tsv")), "train")

  def get_dev_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "dev.tsv")), "dev")

  def get_test_examples(self, data_dir):
    """See base class."""
    return self._create_examples(
        self._read_tsv(os.path.join(data_dir, "test.tsv")), "test")

  def get_labels(self):
    """See base class."""
    return ["0", "1"]

  def _create_examples(self, lines, set_type):
    """Creates examples for the training and dev sets."""
    examples = []
    for (i, line) in enumerate(lines):
      # Only the test set has a header
```

```
      if set_type == "test" and i == 0:
        continue
      guid = "%s-%s" % (set_type, i)
      if set_type == "test":
        text_a = tokenization.convert_to_unicode(line[1])
        label = "0"
      else:
        text_a = tokenization.convert_to_unicode(line[3])
        label = tokenization.convert_to_unicode(line[1])
      examples.append(
          InputExample(guid=guid, text_a=text_a, text_b=None, label=label))
    return examples
```

五、examples转换成features (file_based_convert_examples_to_features)

5.1 单例转化

- 作用：将单个InputExample转换为单个InputFeatures。
- 流程：
  - step 1：判断 example 是否是 PaddingInputExample
  - step 2：构建 label map
  - step 3：text_a 序列化
  - step 4：text_b 序列化
  - step 5：训练 长度修改
  - step 6：输入数据 转化未 Bert 所要求类型数据
  - step 7:输入数据 转化为 id 系列
  - step 8：Mask 数据
  - step 9：利用 0 填充
  - step 10：标签 处理
  - step 11：构建 InputExample

```
def convert_single_example(
  ex_index,
  example,
  label_list,
  max_seq_length,
  tokenizer):
  """将单个 InputExample 转换为单个InputFeatures。"""
  # step 1：判断 example 是否是 PaddingInputExample
  if isinstance(example, PaddingInputExample):
    return InputFeatures(
        input_ids=[0] * max_seq_length,
        input_mask=[0] * max_seq_length,
        segment_ids=[0] * max_seq_length,
        label_id=0,
        is_real_example=False)
  # step 2：构建 label map
  label_map = {}
  for (i, label) in enumerate(label_list):
    label_map[label] = i
```

```
  # step 3: text_a 序列化
  tokens_a = tokenizer.tokenize(example.text_a)
  # step 4: text_b 序列化
  tokens_b = None
  if example.text_b:
    tokens_b = tokenizer.tokenize(example.text_b)
  # step 5: 训练 长度修改
  if tokens_b:
    # 在适当位置修改`tokens_a`和`tokens_b`，以使总长度小于指定长度。
    # Account for [CLS], [SEP], [SEP] with "- 3"
    _truncate_seq_pair(tokens_a, tokens_b, max_seq_length - 3)
  else:
    # Account for [CLS] and [SEP] with "- 2"
    if len(tokens_a) > max_seq_length - 2:
      tokens_a = tokens_a[0:(max_seq_length - 2)]

  # step 6: 输入数据 转化未 Bert 所要求类型数据
  # The convention in BERT is:
  # (a) For sequence pairs:
  #  tokens:   [CLS] is this jack ##son ##ville ? [SEP] no it is not . [SEP]
  #  type_ids: 0     0  0    0    0     0       0 0     1  1  1  1   1 1
  # (b) For single sequences:
  #  tokens:   [CLS] the dog is hairy . [SEP]
  #  type_ids: 0     0   0   0  0     0 0
  #
  # Where "type_ids" are used to indicate whether this is the first sequence or
the second sequence. The embedding vectors for `type=0` and `type=1` were learned
during pre-training and are added to the wordpiece embedding vector (and position
vector). This is not *strictly* necessary since the [SEP] token unambiguously
separates the sequences, but it makes it easier for the model to learn the
concept of sequences.
  #
  # For classification tasks, the first vector (corresponding to [CLS]) is used
as the "sentence vector". Note that this only makes sense because the entire
model is fine-tuned.
  tokens = []
  segment_ids = []
  tokens.append("[CLS]")
  segment_ids.append(0)
  for token in tokens_a:
    tokens.append(token)
    segment_ids.append(0)
  tokens.append("[SEP]")
  segment_ids.append(0)

  if tokens_b:
    for token in tokens_b:
      tokens.append(token)
      segment_ids.append(1)
    tokens.append("[SEP]")
    segment_ids.append(1)

  # step 7:输入数据 转化为 id 系列
  input_ids = tokenizer.convert_tokens_to_ids(tokens)
```

```
  # step 8: Mask 数据
  # The mask has 1 for real tokens and 0 for padding tokens. Only real
  # tokens are attended to.
  input_mask = [1] * len(input_ids)

  # step 9: 利用 0 填充
  # Zero-pad up to the sequence length.
  while len(input_ids) < max_seq_length:
    input_ids.append(0)
    input_mask.append(0)
    segment_ids.append(0)

  assert len(input_ids) == max_seq_length
  assert len(input_mask) == max_seq_length
  assert len(segment_ids) == max_seq_length
  # step 10: 标签 处理
  label_id = label_map[example.label]
  if ex_index < 5:
    tf.logging.info("*** Example ***")
    tf.logging.info("guid: %s" % (example.guid))
    tf.logging.info("tokens: %s" % " ".join(
        [tokenization.printable_text(x) for x in tokens]))
    tf.logging.info("input_ids: %s" % " ".join([str(x) for x in input_ids]))
    tf.logging.info("input_mask: %s" % " ".join([str(x) for x in input_mask]))
    tf.logging.info("segment_ids: %s" % " ".join([str(x) for x in segment_ids]))
    tf.logging.info("label: %s (id = %d)" % (example.label, label_id))
  # step 11:构建 InputFeatures 实例
  feature = InputFeatures(
      input_ids=input_ids,
      input_mask=input_mask,
      segment_ids=segment_ids,
      label_id=label_id,
      is_real_example=True)
  return feature
```

5.2 单例转化

```
def file_based_convert_examples_to_features(
    examples, label_list, max_seq_length, tokenizer, output_file):
  """Convert a set of `InputExample`s to a TFRecord file."""

  writer = tf.python_io.TFRecordWriter(output_file)

  for (ex_index, example) in enumerate(examples):
    if ex_index % 10000 == 0:
      tf.logging.info("Writing example %d of %d" % (ex_index, len(examples)))

    feature = convert_single_example(ex_index, example, label_list,
                                     max_seq_length, tokenizer)

    def create_int_feature(values):
      f = tf.train.Feature(int64_list=tf.train.Int64List(value=list(values)))
      return f
```

```
    features = collections.OrderedDict()
    features["input_ids"] = create_int_feature(feature.input_ids)
    features["input_mask"] = create_int_feature(feature.input_mask)
    features["segment_ids"] = create_int_feature(feature.segment_ids)
    features["label_ids"] = create_int_feature([feature.label_id])
    features["is_real_example"] = create_int_feature(
        [int(feature.is_real_example)])

    tf_example = tf.train.Example(features=tf.train.Features(feature=features))
    writer.write(tf_example.SerializeToString())
  writer.close()
```

六、创建模型

6.1 create_model 创建 分类模型

```
def create_model(bert_config, is_training, input_ids, input_mask, segment_ids,
                 labels, num_labels, use_one_hot_embeddings):
  """创建 分类模型"""
  model = modeling.BertModel(
      config=bert_config,
      is_training=is_training,
      input_ids=input_ids,
      input_mask=input_mask,
      token_type_ids=segment_ids,
      use_one_hot_embeddings=use_one_hot_embeddings)

  # In the demo, we are doing a simple classification task on the entire
segment.
  #
  # If you want to use the token-level output, use model.get_sequence_output()
instead.
  output_layer = model.get_pooled_output()

  hidden_size = output_layer.shape[-1].value

  output_weights = tf.get_variable(
      "output_weights", [num_labels, hidden_size],
      initializer=tf.truncated_normal_initializer(stddev=0.02))

  output_bias = tf.get_variable(
      "output_bias", [num_labels], initializer=tf.zeros_initializer())
  # 计算损失函数
  with tf.variable_scope("loss"):
    if is_training:
      # I.e., 0.1 dropout
      output_layer = tf.nn.dropout(output_layer, keep_prob=0.9)

    logits = tf.matmul(output_layer, output_weights, transpose_b=True)
    logits = tf.nn.bias_add(logits, output_bias)
    probabilities = tf.nn.softmax(logits, axis=-1)
    log_probs = tf.nn.log_softmax(logits, axis=-1)
```

```
    one_hot_labels = tf.one_hot(labels, depth=num_labels, dtype=tf.float32)

    per_example_loss = -tf.reduce_sum(one_hot_labels * log_probs, axis=-1)
    loss = tf.reduce_mean(per_example_loss)

    return (loss, per_example_loss, logits, probabilities)
```

6.2 model_fn_builder

- 作用:

```
def model_fn_builder(bert_config, num_labels, init_checkpoint, learning_rate,
                     num_train_steps, num_warmup_steps, use_tpu,
                     use_one_hot_embeddings):
  """Returns `model_fn` closure for TPUEstimator."""

  def model_fn(features, labels, mode, params):  # pylint: disable=unused-
argument
    """The `model_fn` for TPUEstimator."""

    tf.logging.info("*** Features ***")
    for name in sorted(features.keys()):
      tf.logging.info("  name = %s, shape = %s" % (name, features[name].shape))

    input_ids = features["input_ids"]
    input_mask = features["input_mask"]
    segment_ids = features["segment_ids"]
    label_ids = features["label_ids"]
    is_real_example = None
    if "is_real_example" in features:
      is_real_example = tf.cast(features["is_real_example"], dtype=tf.float32)
    else:
      is_real_example = tf.ones(tf.shape(label_ids), dtype=tf.float32)

    is_training = (mode == tf.estimator.ModeKeys.TRAIN)

    # 总的损失定义为两者之和
    (total_loss, per_example_loss, logits, probabilities) = create_model(
        bert_config, is_training, input_ids, input_mask, segment_ids, label_ids,
        num_labels, use_one_hot_embeddings)
    # 获取所有变量
    tvars = tf.trainable_variables()
    initialized_variable_names = {}
    scaffold_fn = None
    # 如果有之前保存的模型，则进行恢复
    if init_checkpoint:
      (assignment_map, initialized_variable_names
      ) = modeling.get_assignment_map_from_checkpoint(tvars, init_checkpoint)
      if use_tpu:

        def tpu_scaffold():
          tf.train.init_from_checkpoint(init_checkpoint, assignment_map)
          return tf.train.Scaffold()

        scaffold_fn = tpu_scaffold
```

```
      else:
        tf.train.init_from_checkpoint(init_checkpoint, assignment_map)

    tf.logging.info("**** Trainable Variables ****")
    for var in tvars:
      init_string = ""
      if var.name in initialized_variable_names:
        init_string = ", *INIT_FROM_CKPT*"
      tf.logging.info("  name = %s, shape = %s%s", var.name, var.shape,
                      init_string)
    # 训练过程，获得spec
    output_spec = None
    if mode == tf.estimator.ModeKeys.TRAIN:

      train_op = optimization.create_optimizer(
          total_loss, learning_rate, num_train_steps, num_warmup_steps, use_tpu)

      output_spec = tf.contrib.tpu.TPUEstimatorSpec(
          mode=mode,
          loss=total_loss,
          train_op=train_op,
          scaffold_fn=scaffold_fn)
    # 验证过程spec
    elif mode == tf.estimator.ModeKeys.EVAL:

      def metric_fn(per_example_loss, label_ids, logits, is_real_example):
        predictions = tf.argmax(logits, axis=-1, output_type=tf.int32)
        accuracy = tf.metrics.accuracy(
            labels=label_ids, predictions=predictions, weights=is_real_example)
        loss = tf.metrics.mean(values=per_example_loss, weights=is_real_example)
        return {
            "eval_accuracy": accuracy,
            "eval_loss": loss,
        }

      eval_metrics = (metric_fn,
                      [per_example_loss, label_ids, logits, is_real_example])
      output_spec = tf.contrib.tpu.TPUEstimatorSpec(
          mode=mode,
          loss=total_loss,
          eval_metrics=eval_metrics,
          scaffold_fn=scaffold_fn)
    # 预测过程spec
    else:
      output_spec = tf.contrib.tpu.TPUEstimatorSpec(
          mode=mode,
          predictions={"probabilities": probabilities},
          scaffold_fn=scaffold_fn)
    return output_spec

  return model_fn
```

七、主入口

```
def main(_):
  tf.logging.set_verbosity(tf.logging.INFO)
  # 任务处理器 映射表
  processors = {
      "cola": ColaProcessor,
      "mnli": MnliProcessor,
      "mrpc": MrpcProcessor,
      "xnli": XnliProcessor,
  }

  tokenization.validate_case_matches_checkpoint(FLAGS.do_lower_case,
                                                FLAGS.init_checkpoint)

  if not FLAGS.do_train and not FLAGS.do_eval and not FLAGS.do_predict:
    raise ValueError(
        "At least one of `do_train`, `do_eval` or `do_predict' must be True.")
  # 加载 Bert 配置
  bert_config = modeling.BertConfig.from_json_file(FLAGS.bert_config_file)

  if FLAGS.max_seq_length > bert_config.max_position_embeddings:
    raise ValueError(
        "Cannot use sequence length %d because the BERT model "
        "was only trained up to sequence length %d" %
        (FLAGS.max_seq_length, bert_config.max_position_embeddings))

  tf.gfile.MakeDirs(FLAGS.output_dir)

  task_name = FLAGS.task_name.lower()

  if task_name not in processors:
    raise ValueError("Task not found: %s" % (task_name))
  # 定义任务处理器
  processor = processors[task_name]()
  # 获取标签项
  label_list = processor.get_labels()
  # 数据预处理
  tokenizer = tokenization.FullTokenizer(
      vocab_file=FLAGS.vocab_file, do_lower_case=FLAGS.do_lower_case)

  tpu_cluster_resolver = None
  if FLAGS.use_tpu and FLAGS.tpu_name:
    tpu_cluster_resolver = tf.contrib.cluster_resolver.TPUClusterResolver(
        FLAGS.tpu_name, zone=FLAGS.tpu_zone, project=FLAGS.gcp_project)

  is_per_host = tf.contrib.tpu.InputPipelineConfig.PER_HOST_V2
  run_config = tf.contrib.tpu.RunConfig(
      cluster=tpu_cluster_resolver,
      master=FLAGS.master,
      model_dir=FLAGS.output_dir,
      save_checkpoints_steps=FLAGS.save_checkpoints_steps,
      tpu_config=tf.contrib.tpu.TPUConfig(
          iterations_per_loop=FLAGS.iterations_per_loop,
          num_shards=FLAGS.num_tpu_cores,
          per_host_input_for_training=is_per_host))
```

```
train_examples = None
num_train_steps = None
num_warmup_steps = None
# 模型训练 数据加载
if FLAGS.do_train:
  # 加载训练数据
  train_examples = processor.get_train_examples(FLAGS.data_dir)
  num_train_steps = int(
      len(train_examples) / FLAGS.train_batch_size * FLAGS.num_train_epochs)
  num_warmup_steps = int(num_train_steps * FLAGS.warmup_proportion)
# 自定义模型用于estimator训练
model_fn = model_fn_builder(
    bert_config=bert_config,
    num_labels=len(label_list),
    init_checkpoint=FLAGS.init_checkpoint,
    learning_rate=FLAGS.learning_rate,
    num_train_steps=num_train_steps,
    num_warmup_steps=num_warmup_steps,
    use_tpu=FLAGS.use_tpu,
    use_one_hot_embeddings=FLAGS.use_tpu)

# 如果没有TPU，会自动转为CPU/GPU的Estimator
estimator = tf.contrib.tpu.TPUEstimator(
    use_tpu=FLAGS.use_tpu,
    model_fn=model_fn,
    config=run_config,
    train_batch_size=FLAGS.train_batch_size,
    eval_batch_size=FLAGS.eval_batch_size,
    predict_batch_size=FLAGS.predict_batch_size)
# 模型 训练
if FLAGS.do_train:
  train_file = os.path.join(FLAGS.output_dir, "train.tf_record")
  file_based_convert_examples_to_features(
      train_examples, label_list, FLAGS.max_seq_length, tokenizer, train_file)
  tf.logging.info("***** Running training *****")
  tf.logging.info("  Num examples = %d", len(train_examples))
  tf.logging.info("  Batch size = %d", FLAGS.train_batch_size)
  tf.logging.info("  Num steps = %d", num_train_steps)
  train_input_fn = file_based_input_fn_builder(
      input_file=train_file,
      seq_length=FLAGS.max_seq_length,
      is_training=True,
      drop_remainder=True)
  estimator.train(input_fn=train_input_fn, max_steps=num_train_steps)
# 模型 验证 数据加载
if FLAGS.do_eval:
  eval_examples = processor.get_dev_examples(FLAGS.data_dir)
  num_actual_eval_examples = len(eval_examples)
  if FLAGS.use_tpu:
    # TPU requires a fixed batch size for all batches, therefore the number
    # of examples must be a multiple of the batch size, or else examples
    # will get dropped. So we pad with fake examples which are ignored
    # later on. These do NOT count towards the metric (all tf.metrics
    # support a per-instance weight, and these get a weight of 0.0).
    while len(eval_examples) % FLAGS.eval_batch_size != 0:
```

```
        eval_examples.append(PaddingInputExample())

    eval_file = os.path.join(FLAGS.output_dir, "eval.tf_record")
    file_based_convert_examples_to_features(
        eval_examples, label_list, FLAGS.max_seq_length, tokenizer, eval_file)

    tf.logging.info("***** Running evaluation *****")
    tf.logging.info("  Num examples = %d (%d actual, %d padding)",
                    len(eval_examples), num_actual_eval_examples,
                    len(eval_examples) - num_actual_eval_examples)
    tf.logging.info("  Batch size = %d", FLAGS.eval_batch_size)

    # This tells the estimator to run through the entire set.
    eval_steps = None
    # However, if running eval on the TPU, you will need to specify the
    # number of steps.
    if FLAGS.use_tpu:
      assert len(eval_examples) % FLAGS.eval_batch_size == 0
      eval_steps = int(len(eval_examples) // FLAGS.eval_batch_size)

    eval_drop_remainder = True if FLAGS.use_tpu else False
    eval_input_fn = file_based_input_fn_builder(
        input_file=eval_file,
        seq_length=FLAGS.max_seq_length,
        is_training=False,
        drop_remainder=eval_drop_remainder)

    result = estimator.evaluate(input_fn=eval_input_fn, steps=eval_steps)

    output_eval_file = os.path.join(FLAGS.output_dir, "eval_results.txt")
    with tf.gfile.GFile(output_eval_file, "w") as writer:
      tf.logging.info("***** Eval results *****")
      for key in sorted(result.keys()):
        tf.logging.info("  %s = %s", key, str(result[key]))
        writer.write("%s = %s\n" % (key, str(result[key])))
  # 模型预测
  if FLAGS.do_predict:
    predict_examples = processor.get_test_examples(FLAGS.data_dir)
    num_actual_predict_examples = len(predict_examples)
    if FLAGS.use_tpu:
      # TPU requires a fixed batch size for all batches, therefore the number
      # of examples must be a multiple of the batch size, or else examples
      # will get dropped. So we pad with fake examples which are ignored
      # later on.
      while len(predict_examples) % FLAGS.predict_batch_size != 0:
        predict_examples.append(PaddingInputExample())

    predict_file = os.path.join(FLAGS.output_dir, "predict.tf_record")
    file_based_convert_examples_to_features(predict_examples, label_list,
                                            FLAGS.max_seq_length, tokenizer,
                                            predict_file)

    tf.logging.info("***** Running prediction*****")
    tf.logging.info("  Num examples = %d (%d actual, %d padding)",
                    len(predict_examples), num_actual_predict_examples,
```

```
                    len(predict_examples) - num_actual_predict_examples)
    tf.logging.info("  Batch size = %d", FLAGS.predict_batch_size)

    predict_drop_remainder = True if FLAGS.use_tpu else False
    predict_input_fn = file_based_input_fn_builder(
        input_file=predict_file,
        seq_length=FLAGS.max_seq_length,
        is_training=False,
        drop_remainder=predict_drop_remainder)

    result = estimator.predict(input_fn=predict_input_fn)

    output_predict_file = os.path.join(FLAGS.output_dir, "test_results.tsv")
    with tf.gfile.GFile(output_predict_file, "w") as writer:
      num_written_lines = 0
      tf.logging.info("***** Predict results *****")
      for (i, prediction) in enumerate(result):
        probabilities = prediction["probabilities"]
        if i >= num_actual_predict_examples:
          break
        output_line = "\t".join(
            str(class_probability)
            for class_probability in probabilities) + "\n"
        writer.write(output_line)
        num_written_lines += 1
    assert num_written_lines == num_actual_predict_examples
```

八、总结

本章 主要介绍了 利用 Bert fineturn，代码比较简单。

参考文档

1. Bert系列（四）——源码解读之Fine-tune
2. BERT源码分析PART III