简单高效的Bert中文文本分类模型开发和部署

准备环境工作

- **操作系统**：Linux
- **TensorFlow Version**：1.13.1，动态图模式
- **GPU**：12G GPU，文档后面有显存不足的解决方案
- **TensorFlow Serving**：simple-tensorflow-serving
- **依赖库**：requirements.txt

目录结构说明

```
.
├── bert
|   ├── README.md
|   ├── __init__.py
|   ├── create_pretraining_data.py
|   ├── extract_features.py
|   ├── modeling.py
|   ├── modeling_test.py
|   ├── multilingual.md
|   ├── optimization.py
|   ├── optimization_test.py
|   ├── requirements.txt
|   ├── run_classifier.py
|   ├── run_pretraining.py
|   ├── run_squad.py
|   ├── sample_text.txt
|   ├── tokenization.py
|   └── tokenization_test.py
├── bert_classifier.py
├── client.py
├── client.sh
├── data
|   ├── sentiment_analysis_data.txt
|   ├── test.data
|   ├── train.data
|   └── val.data
├── export.py
├── export.sh
├── file_base_client.py
├── predict.sh
├── requirements.txt
└── train.sh

2 directories, 29 files
```

- bert是官方源码
- data是3分类的文本情感分析数据
- train.sh、classifier.py 训练文件
- export.sh、export.py导出TF serving的模型
- client.sh、client.py、file_base_client.py 处理输入数据并向部署的TF serving的模型发出请求，打印输出结果

训练代码

1. 写一个自己的文本处理器。需要**注意**：
2. 改写label
3. 把create_examples改成了共有方法，因为我们后面要调用。
4. file_base的时候注意跳过第一行，文件数据的第一行是title

```
class MyProcessor(DataProcessor):

    def get_test_examples(self, data_dir):
        return self.create_examples(
            self._read_tsv(os.path.join(data_dir, "test.tsv")), "test")

    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_pred_examples(self, data_dir):
        return self.create_examples(
            self._read_tsv(os.path.join(data_dir, "pred.tsv")), "pred")

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

    def create_examples(self, lines, set_type, file_base=True):
        """Creates examples for the training and dev sets. each line is
label+\t+text_a+\t+text_b """
        examples = []
        for (i, line) in tqdm(enumerate(lines)):

            if file_base:
                if i == 0:
                    continue

            guid = "%s-%s" % (set_type, i)
            text = tokenization.convert_to_unicode(line[1])
            if set_type == "test" or set_type == "pred":
                label = "0"
            else:
                label = tokenization.convert_to_unicode(line[0])
            examples.append(
                InputExample(guid=guid, text_a=text, label=label))
        return examples
```

5. 其他的训练代码，照抄官方的就行
6. 可以直接运行train.sh，**注意修改对应的路径**
7. 生成的ckpt文件在output路径下

导出模型

主要代码如下，生成的pb文件在api文件夹下

```
def serving_input_receiver_fn():
    input_ids = tf.placeholder(dtype=tf.int64, shape=[None,
FLAGS.max_seq_length], name='input_ids')
    input_mask = tf.placeholder(dtype=tf.int64, shape=[None,
FLAGS.max_seq_length], name='input_mask')
    segment_ids = tf.placeholder(dtype=tf.int64, shape=[None,
FLAGS.max_seq_length], name='segment_ids')
    label_ids = tf.placeholder(dtype=tf.int64, shape=[None, ],
name='unique_ids')

    receive_tensors = {'input_ids': input_ids, 'input_mask': input_mask,
'segment_ids': segment_ids,
                       'label_ids': label_ids}
    features = {'input_ids': input_ids, 'input_mask': input_mask,
'segment_ids': segment_ids, "label_ids": label_ids}
    return tf.estimator.export.ServingInputReceiver(features, receive_tensors)

estimator.export_savedmodel(FLAGS.serving_model_save_path,
serving_input_receiver_fn)
```

TensorFlow Serving部署

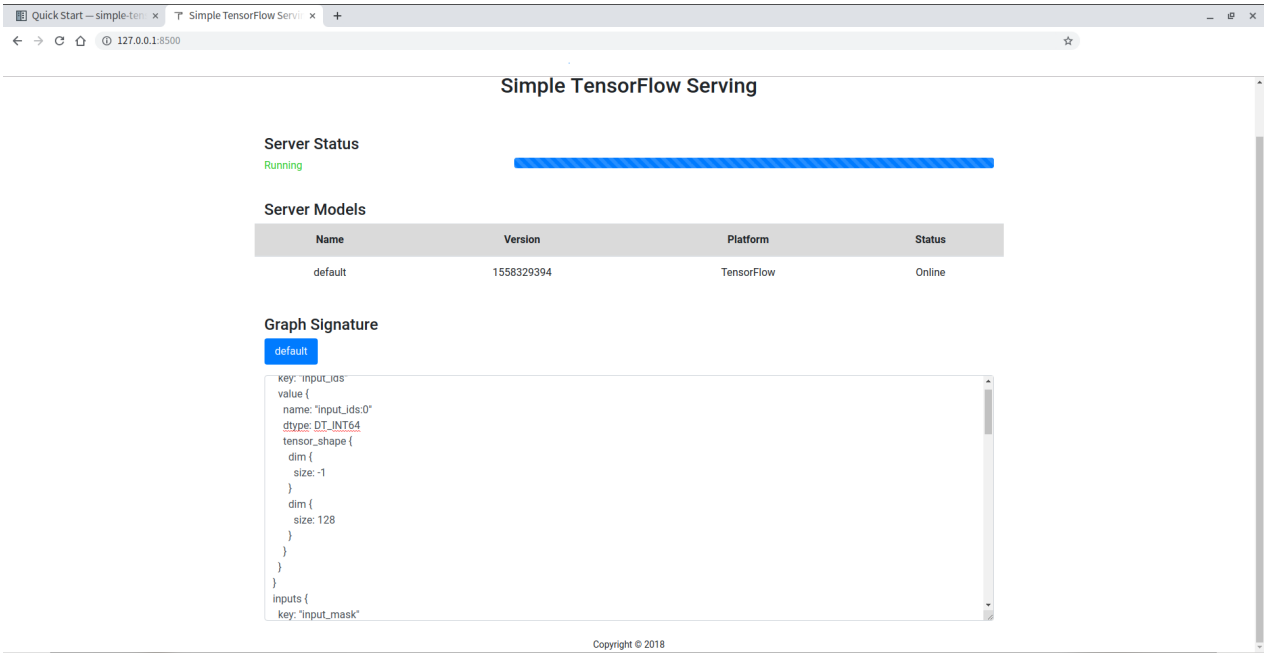
一键部署：

```
simple_tensorflow_serving --model_base_path="./api"
```

正常启动终端界面：

```
Instructions for updating:
Use standard file APIs to check for files with this prefix.
INFO:tensorflow:Restoring parameters from ./api/1558329394/variables/variables
2019-06-20 05:19:22 INFO     Restoring parameters from ./api/1558329394/variables/variables
 * Serving Flask app "simple_tensorflow_serving.server" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
2019-05-20 05:19:23 INFO      * Running on http://0.0.0.0:8500/ (Press CTRL+C to quit)
2019-05-20 05:19:44 INFO     127.0.0.1 - - [20/May/2019 05:19:44] "GET / HTTP/1.1" 200 -
2019-05-20 05:20:39 INFO     127.0.0.1 - - [20/May/2019 05:20:39] "POST / HTTP/1.1" 400 -
2019-05-20 05:22:16 INFO     127.0.0.1 - - [20/May/2019 05:22:16] "GET /v1/models/default/gen_client?language=python HTTP/1.1" 200 -
2019-05-20 05:22:41.714323: I tensorflow/stream_executor/dso_loader.cc:152] successfully opened CUDA library libcublas.so.10.0 locally
2019-05-20 05:22:41 INFO     127.0.0.1 - - [20/May/2019 05:22:41] "POST / HTTP/1.1" 200 -
2019-05-20 05:24:27 INFO     127.0.0.1 - - [20/May/2019 05:24:27] "POST / HTTP/1.1" 200 -
2019-05-20 05:25:59 INFO     127.0.0.1 - - [20/May/2019 05:25:59] "POST / HTTP/1.1" 200 -
2019-05-20 05:27:32 INFO     127.0.0.1 - - [20/May/2019 05:27:32] "POST / HTTP/1.1" 200 -
2019-05-20 05:30:57 INFO     127.0.0.1 - - [20/May/2019 05:30:57] "POST / HTTP/1.1" 200 -
2019-05-20 05:31:40 INFO     127.0.0.1 - - [20/May/2019 05:31:40] "POST / HTTP/1.1" 200 -
2019-05-20 05:41:23 INFO     127.0.0.1 - - [20/May/2019 05:41:23] "POST / HTTP/1.1" 200 -
2019-05-20 05:42:02 INFO     127.0.0.1 - - [20/May/2019 05:42:02] "POST / HTTP/1.1" 200 -
```

浏览器访问界面：

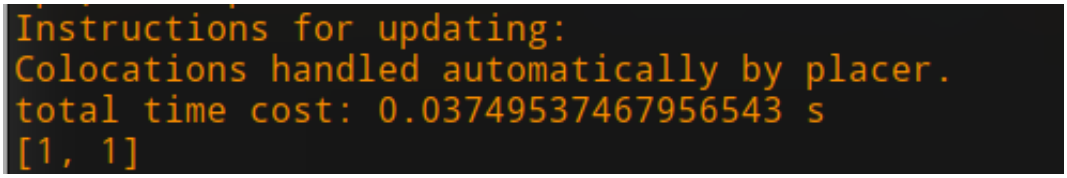


这部分认真阅读simple-tensorflow-serving的文档

本地请求代码

分为两种，一种是读取文件的，就是要预测的文本是tsv文件的，叫做file_base_client.py，另一个直接输入文本的是client.py。首先更改input_fn_builder，返回dataset，然后从dataset中取数据，转换为list格式，传入模型，返回结果。

正常情况下的运行结果：



问题解答

- 训练的显存不足怎么办

  答：按照官方的建议，调小\max_seq_length和train_batch_size