基于 OpenVINO™ 开发套件“无缝”部署 PaddleNLP 模型

作者：杨亦诚

任务背景

1. 情感分析 (Sentiment Analysis)

情感分析旨在对带有情感色彩的主观性文本进行分析、处理、归纳和推理，其广泛应用于消费决策、舆情分析、个性化推荐等领域，具有很高的商业价值。例如：食行生鲜自动生成菜品评论标签辅助用户购买，并指导运营采购部门调整选品和促销策略；房天下向购房者和开发商直观展示楼盘的用户口碑情况，并对好评楼盘置顶推荐；国美搭建服务智能化评分系统，客服运营成本减少 40%，负面反馈处理率 100%。

2. 自然语言处理（NLP）技术

自然语言处理（英语：Natural Language Process，简称 NLP）是计算机科学、信息工程以及人工智能的子领域，专注于人机语言交互，探讨如何处理和运用自然语言。最近几年，随着深度学习以及相关技术的发展，NLP 领域的研究取得一个又一个突破，研究者设计各种模型和方法，来解决 NLP 的各类问题，其中比较常见包括 LSTM, BERT, GRU, Transformer, GPT 等算法模型。

方案简介

本方案采用 PaddleNLP 工具套件进行模型训练，并基于 OpenVINO™ 开发套件实现在 Intel 平台上的高效能部署。本文将主要分享如何在 OpenVINO™ 开发套件中“无缝”部署 PaddlePaddle BERT 模型，并对输出结果做验证。

1. PaddleNLP

PaddleNLP 是一款简单易用且功能强大的自然语言处理开发库。聚合业界优质预训练模型并提供开箱即用的开发体验，覆盖 NLP 多场景的模型库搭配产业实践范例可满足开发者灵活定制的需求。

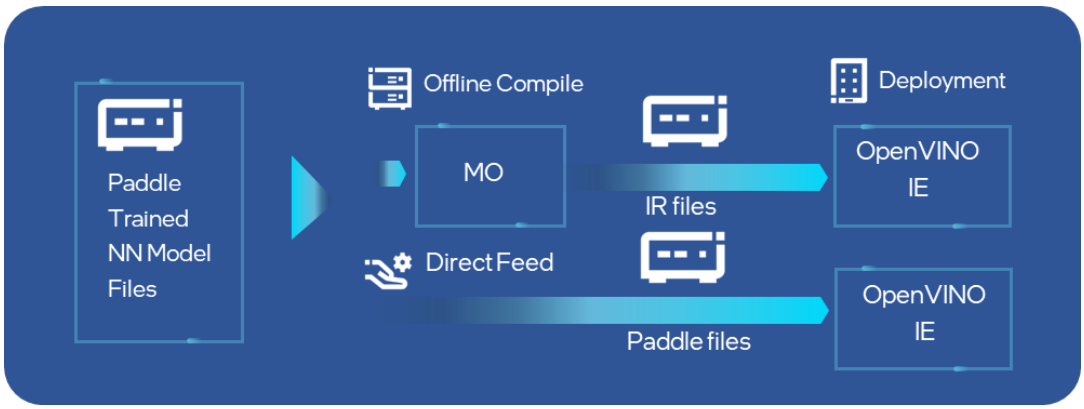
2. OpenVINO ™ 开发套件

OpenVINO™ 开发套件是 Intel 平台原生的深度学习推理框架，自 2018 年推出以来，Intel 已经帮助数十万开发者大幅提升了 AI 推理性能，并将其应用从边缘计算扩展到企业和客户端。英特尔于 2022 年巴塞罗那世界移动通信大会前夕，推出了英特尔®发行版 OpenVINO™ 开发套件的全新版本。其中的新功能主要根据开发者过去三年半的反馈而开发，包括更多的深度学习模型选择、更多的

设备可移植性选择以及更高的推理性能和更少的代码更改。为了更好地对 Paddle 模型进行支持，新版 OpenVINO™ 开发套件分别做了一下升级：

- 直接支持 Paddle 格式模型

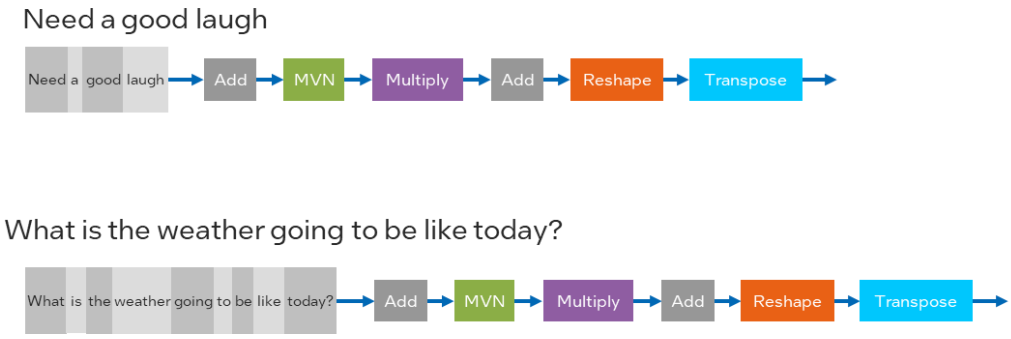
目前 OpenVINO TM 开发套件 2022.1 发行版中已完成对 PaddlePaddle 模型的直接支持，OpenVINO™ 开发套件的 Model Optimizer 工具已经可以直接完成对 Paddle 模型的离线转化，同时 runtime api 接口也可以直接读取加载 Paddle 模型到指定的硬件设备，省去了离线转换的过程，大大提升了 Paddle 开发者在 Intel 平台上部署的效率。经过性能和准确性验证，在 OpenVINO™ 开发套件 2022.1 发行版中，会有 13 个模型涵盖 5 大应用场景的 Paddle 模型将被直接支持，其中不乏像 PPYolo 和 PPOCR 这样非常受开发者欢迎的网络。



图：OpenVINO™ 开发套件的 MO 和 IE 可以直接支持 Paddle 模型输入

- 全面引入动态输入支持

为了适配更广泛的模型种类，OpenVINO™ 2022.1 版本的 CPU Plugin 已经支持了动态 input shape，让开发者以更便捷的方式部署类似 NLP 或者 OCR 这样的网络，OpenVINO™ 开发套件用户可以在不需要对模型做 reshape 的前提下，任意送入不同 shape 的图片或者向量作为输入数据，OpenVINO™ 开发套件会自动在 runtime 过程中对模型结构与内存空间进行动态调整，进一步优化 dynamic shape 的推理性能.



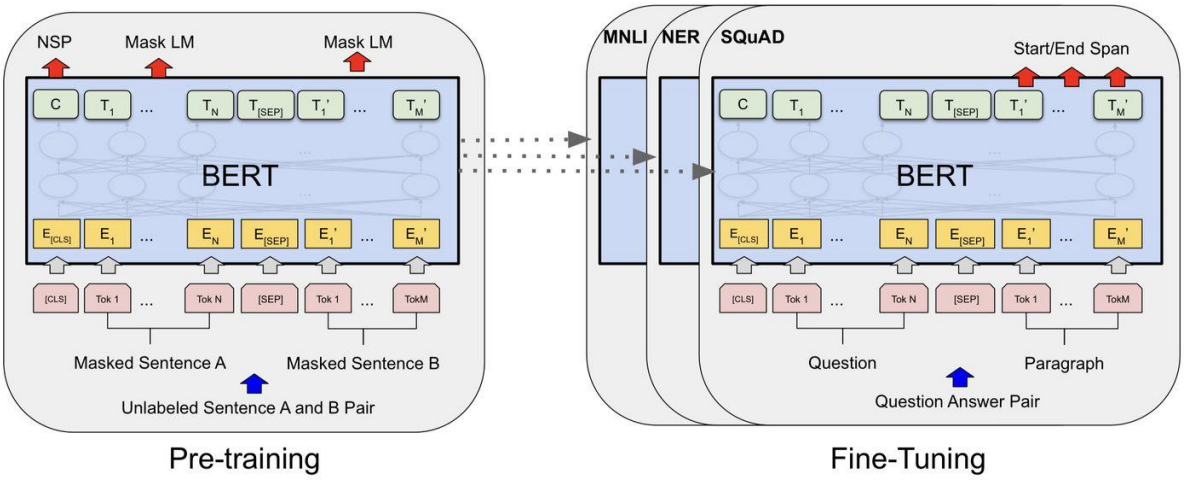
图：在 NLP 中的 Dynamic Input Shape

详细介绍可以参考：
https://docs.openvino.ai/latest/openvino_docs_OV_UG_DynamicShapes.html

BERT 原理简介

1. BERT 结构介绍

BERT （Bidirectional Encoder Representations from Transformers）以 Transformer 编码器为网络基本组件，使用掩码语言模型（Masked Language Model）和邻接句子预测（Next Sentence Prediction）两个任务在大规模无标注文本语料上进行预训练（pre-train），得到融合了双向内容的通用语义表示模型。以预训练产生的通用语义表示模型为基础，结合任务适配的简单输出层，微调（fine-tune）后即可应用到下游的 NLP 任务，效果通常也较直接在下游的任务上训练的模型更优。此前 BERT 即在 GLUE 评测任务上取得了 SOTA 的结果。



图：BERT 的 2 阶段训练任务

不难发现，其模型结构是 Transformer 的 Encoder 层，只需要将特定任务的输入，输出插入到 Bert 中，利用 Transformer 强大的注意力机制就可以模拟很多下游任务。(句子对关系判断，单文本主题分类，问答任务(QA)，单句贴标签(命名实体识别))，BERT 的训练过程可以分成预训练和微调两部分组成。

2. 预训练任务（Pre-training）

BERT 是一个多任务模型，它的任务是由两个自监督任务组成，即 MLM 和 NSP。

- Task #1： Masked Language Model

所谓 MLM 是指在训练的时候随即从输入预料上 mask 掉一些单词，然后通过的上下文预测该单词，该任务非常像我们在中学时期经常做的完形填空。正如传统的语言模型算法和 RNN 匹配那样，MLM 的这个性质和 Transformer 的结构是非常匹配的。

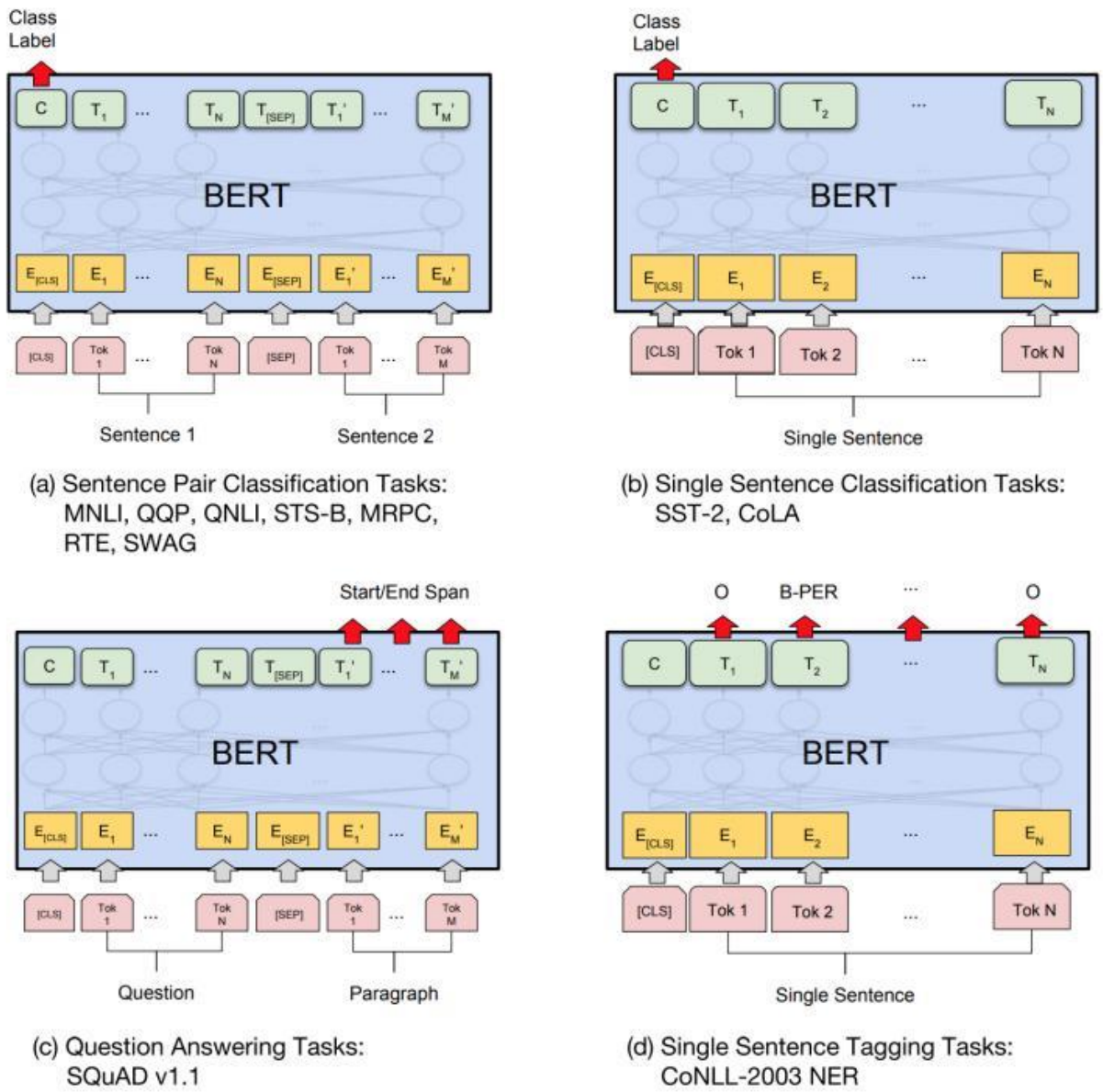
- Task #2: Next Sentence Prediction

Next Sentence Prediction（NSP）的任务是判断句子 B 是否是句子 A 的下文。如果是的话输出'IsNext'，否则输出'NotNext'。训练数据的生成方式是从平行语料中随机抽取的连续两句话，其中 50%保留抽取的两句话，它们符合 IsNext 关系，另外 50%的第二句话是随机从预料中提取的，它们的关系是 NotNext 的。

3. 微调任务 （Fine-tuning）

在海量单预料上训练完 BERT 之后，便可以将其应用到 NLP 的各个任务中了。以下展示了 BERT 在 11 个不同任务中的模型，它们只需要在 BERT 的基础上再添加一个输出层便可以完成对特定任务的微调。这些任务类似于我们做过的文科试卷，其中有选择题，简答题等等。微调的任务包括：

- 基于句子对的分类任务
- 基于单个句子的分类任务
- 问答任务
- 命名实体识别



图：BERT 的 4 大下游微调任务

训练与部署流程

本示例包含 PaddleNLP 训练和 OpenVINO™ 开发套件部署两部分组成。

1. 环境安装

打开命令行终端，分别输入以下命令，完成本地环境安装和配置。

1.1. 安装 PaddlePaddle (AI studio 环境中可以略过)

- 如果是 CPU 训练环境需要执行以下命令进行安装：

[22]
```
!pip install paddlepaddle==2.3.0
```

运行时长: 38秒499毫秒　结束时间: 2022-06-10 17:04:06

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting paddlepaddle==2.3.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/9d/a4/0da466d67fbfd52d7dba987a41ae0d71bd80a7f6492d1dab9c585d6eb5bf/paddlepaddle-2.3.0-cp37-cp37m-manylinux1_x86_64.whl (112.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.3/112.3 MB 2.8 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: numpy>=1.13 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle==2.3.0) (1.19.5)
Requirement already satisfied: requests>=2.20.0 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle==2.3.0) (2.24.0)
Requirement already satisfied: astor in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle==2.3.0) (0.8.1)
Requirement already satisfied: opt-einsum==3.3.0 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle==2.3.0) (3.3.0)
Requirement already satisfied: protobuf>=3.1.0 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle==2.3.0) (3.14.0)
```

- 如果是 GPU 训练环境需要执行以下命令进行安装：

```
!pip install paddlepaddle-gpu==2.3.0
```

运行时长: 2分钟49秒736毫秒　结束时间: 2022-06-10 17:01:06

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting paddlepaddle-gpu==2.3.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/e5/6c/b9cf5eafe91afe38397c1ee78ba9f3a4435faa5a89dc216d69fcf8ae9627/paddlepaddle_gpu-2.3.0-cp37-cp37m-manylinux1_x86_64.whl (576.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 576.1/576.1 MB 1.3 MB/s eta 0:00:0000:0100:04
Requirement already satisfied: requests>=2.20.0 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle-gpu==2.3.0) (2.24.0)
Requirement already satisfied: astor in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle-gpu==2.3.0) (0.8.1)
Requirement already satisfied: numpy>=1.13 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle-gpu==2.3.0) (1.19.5)
Requirement already satisfied: protobuf>=3.1.0 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle-gpu==2.3.0) (3.14.0)
Requirement already satisfied: paddle-bfloat==0.1.2 in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from paddlepaddle-gpu==2.3.0) (0.1.2)
```

1.2. 安装 PaddleNLP 与相关依赖

- 下载 PaddleNLP

```
!git clone https://gitee.com/paddlepaddle/PaddleNLP.git
```

```
Cloning into 'PaddleNLP'...
remote: Enumerating objects: 22494, done.
remote: Counting objects: 100% (9716/9716), done.
remote: Compressing objects: 100% (4445/4445), done.
remote: Total 22494 (delta 6139), reused 8227 (delta 4936), pack-reused 12778
Receiving objects: 100% (22494/22494), 70.41 MiB | 8.66 MiB/s, done.
Resolving deltas: 100% (14489/14489), done.
Checking connectivity... done.
```

- 安装 PaddleNLP 相关依赖

```
!pip install -r PaddleNLP/requirements.txt
!pip install PaddleNLP/.
```

运行时长: 14秒432毫秒　结束时间: 2022-06-10 17:01:45

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: jieba in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from -r
PaddleNLP/requirements.txt (line 1)) (0.42.1)
Requirement already satisfied: colorlog in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from
-r PaddleNLP/requirements.txt (line 2)) (4.1.0)
Requirement already satisfied: colorama in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from
-r PaddleNLP/requirements.txt (line 3)) (0.4.4)
Requirement already satisfied: seqeval in /opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages (from
-r PaddleNLP/requirements.txt (line 4)) (1.2.2)
```

1.3. 安装 OpenVINO™ 开发套件

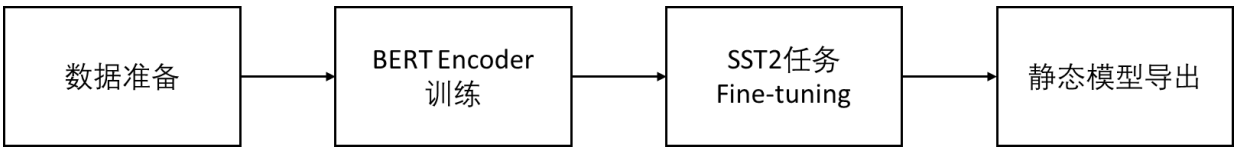
```
!pip install openvino-dev
```

运行时长: 2分钟1秒808毫秒　结束时间: 2022-06-10 17:07:04

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting openvino-dev
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/f2/99/9e55ddb1abce5ff7def768470810044a6de48a5da99b9195498a
d75dfe8a/openvino_dev-2022.1.0-7019-py3-none-any.whl (5.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.8/5.8 MB 3.1 MB/s eta 0:00:0000:0100:01
Collecting scikit-image>=0.17.2
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/d2/d9/d16d4cbb4840e0fb3bd329b49184d240b82b649e1bd579489394
fbc85c81/scikit_image-0.19.2-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (13.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.5/13.5 MB 3.5 MB/s eta 0:00:0000:0100:01
Collecting scikit-learn~=0.24.1
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/a8/eb/a48f25c967526b66d5f1fa7a984594f0bf0a5afafa94a8c4dbc3
17744620/scikit_learn-0.24.2-cp37-cp37m-manylinux2010_x86_64.whl (22.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.3/22.3 MB 3.0 MB/s eta 0:00:0000:0100:01
Collecting requests>=2.25.1
```

2. 训练部分

训练部分是 BERT 在 Paddle 2.0 上的开源实现，可以分为数据准备，BERT Encoder 预训练，SST2 情感分类任务微调以及推理模型导出这四个步骤，可以参 Paddle 官方的案例说明（https://github.com/PaddlePaddle/PaddleNLP/tree/develop/model_zoo/bert），以下过程做了简要汇总。



图：Paddle BERT 模型训练流程

除此之外，我们也可以借助 Paddle AI studio 直接运行训练脚本(**无脑点击运行就可以了:)**)，链接如下：

https://aistudio.baidu.com/aistudio/projectdetail/4193790?contributionType=1

2.1. 步骤一：数据准备（可略过）

PaddleNLP 中 BERT 任务下自带的 create_pretraining_data.py 是创建预训练程序所需数据的脚本。其以文本文件（使用换行符换行和空白符分隔，data 目录下提供了部分示例数据）为输入，经由 BERT tokenizer 进行 tokenize 后再做生成 sentence pair 正负样本、掩码 token 等处理，最后输出 hdf5 格式的数据文件。使用方式如下，在命令行输入：

[5]

```
! python PaddleNLP/model_zoo/bert/create_pretraining_data.py \
  --input_file=PaddleNLP/model_zoo/bert/data/sample_text.txt \
  --output_file=PaddleNLP/model_zoo/bert/data/training_data.hdf5 \
  --bert_model=bert-base-uncased \
  --max_seq_length=128 \
  --max_predictions_per_seq=20 \
  --masked_lm_prob=0.15 \
  --random_seed=12345 \
  --dupe_factor=5

```

运行时长: 2秒387毫秒　结束时间: 2022-06-10 14:26:51

```
/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/setuptools/depends.py:2: DeprecationWarning: the i
mp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  import imp
Namespace(bert_model='bert-base-uncased', do_lower_case=True, dupe_factor=5, input_file='PaddleNLP/model_zoo/bert/dat
a/sample_text.txt', masked_lm_prob=0.15, max_predictions_per_seq=20, max_seq_length=128, output_file='PaddleNLP/model
_zoo/bert/data/training_data.hdf5', random_seed=12345, short_seq_prob=0.1, vocab_file=None)
[2022-06-10 14:26:51,232] [    INFO] - Already cached /home/aistudio/.paddlenlp/models/bert-base-uncased/bert-base-un
cased-vocab.txt
creating instance from PaddleNLP/model_zoo/bert/data/sample_text.txt
100%|[redacted]| 88/88 [00:00<00:00, 19636.05it/s]
saving data
```

2.2. 步骤二：GPU 训练(可略过)

使用 paddle.distributed.launch 配置项运行 run_pretrain.py 训练脚本，可以在多卡 GPU 环境下启动 BERT 预训练任务。命令行指令如下：

```
!unset CUDA_VISIBLE_DEVICES
!python -m paddle.distributed.launch --gpus "0" PaddleNLP/model_zoo/bert/run_pretrain.py \
    --model_type bert \
    --model_name_or_path bert-base-uncased \
    --max_predictions_per_seq 20 \
    --batch_size 32   \
    --learning_rate 1e-4 \
    --weight_decay 1e-2 \
    --adam_epsilon 1e-6 \
    --warmup_steps 10000 \
    --num_train_epochs 3 \
    --input_dir PaddleNLP/model_zoo/bert/data/ \
    --output_dir pretrained_models/ \
    --logging_steps 1 \
    --save_steps 20000 \
    --max_steps 1000000 \
    --device gpu \
    --use_amp False

```

- **model_type** 指示了模型类型，使用 BERT 模型时设置为 bert 即可。
- **model_name_or_path** 指示了某种特定配置的模型，对应有其预训练模型和预训练时使用的 tokenizer。若模型相关内容保存在本地，这里也可以提供相应目录地址。
- **input_dir** 表示输入数据的目录，该目录下所有文件名中包含 training 的文件将被作为训练数据。

- **output_dir** 表示模型的保存目录。

```
W0610 15:29:01.302079  6961 gpu_context.cc:306] device: 0, cuDNN Version: 7.6.
2022-06-10 15:29:06,205-INFO: global step: 1, epoch: 0, batch: 0, loss: 11.224227, avg_reader_cost: 0.11948 sec, av
g_batch_cost: 0.35530 sec, avg_samples: 32.00000, ips: 90.06535 sequences/sec
2022-06-10 15:29:06,426-INFO: global step: 2, epoch: 0, batch: 1, loss: 11.270069, avg_reader_cost: 0.00015 sec, av
g_batch_cost: 0.13860 sec, avg_samples: 32.00000, ips: 230.88839 sequences/sec
2022-06-10 15:29:06,599-INFO: global step: 3, epoch: 0, batch: 2, loss: 11.198269, avg_reader_cost: 0.00022 sec, av
g_batch_cost: 0.11379 sec, avg_samples: 32.00000, ips: 281.22383 sequences/sec
2022-06-10 15:29:06,834-INFO: global step: 4, epoch: 1, batch: 0, loss: 11.201858, avg_reader_cost: 0.00728 sec, av
g_batch_cost: 0.14715 sec, avg_samples: 32.00000, ips: 217.46723 sequences/sec
2022-06-10 15:29:07,055-INFO: global step: 5, epoch: 1, batch: 1, loss: 11.261638, avg_reader_cost: 0.00022 sec, av
g_batch_cost: 0.13678 sec, avg_samples: 32.00000, ips: 233.95637 sequences/sec
2022-06-10 15:29:07,229-INFO: global step: 6, epoch: 1, batch: 2, loss: 11.255038, avg_reader_cost: 0.00023 sec, av
g_batch_cost: 0.11445 sec, avg_samples: 32.00000, ips: 279.59464 sequences/sec
2022-06-10 15:29:07,465-INFO: global step: 7, epoch: 2, batch: 0, loss: 11.288174, avg_reader_cost: 0.00711 sec, av
g_batch_cost: 0.14773 sec, avg_samples: 32.00000, ips: 216.61297 sequences/sec
2022-06-10 15:29:07,687-INFO: global step: 8, epoch: 2, batch: 1, loss: 11.264122, avg_reader_cost: 0.00023 sec, av
g_batch_cost: 0.13797 sec, avg_samples: 32.00000, ips: 231.92812 sequences/sec
2022-06-10 15:29:07,860-INFO: global step: 9, epoch: 2, batch: 2, loss: 11.186976, avg_reader_cost: 0.00023 sec, av
g_batch_cost: 0.11360 sec, avg_samples: 32.00000, ips: 281.70134 sequences/sec
INFO 2022-06-10 15:29:10,605 launch.py:402] Local processes completed.
INFO 2022-06-10 15:29:10,605 launch.py:402] Local processes completed.
```

2.3. 步骤三：模型 Fine-tunning

如果自己没有准备训练数据集的话，也可以跳过前面的步骤，直接使用 huggingface 提供的预训练模型进行 Fine-tuning，以 GLUE 中的 SST-2 任务为例，该脚本会自动下载 SST-2 任务中所需要的英文数据集，启动 Fine-tuning 的方式如下：

```
!python PaddleNLP/model_zoo/bert/run_glue.py \
    --model_type bert \
    --model_name_or_path bert-base-uncased \
    --task_name SST2 \
    --max_seq_length 128 \
    --batch_size 32   \
    --learning_rate 2e-5 \
    --num_train_epochs 3 \
    --logging_steps 1 \
    --save_steps 500 \
    --output_dir ./tmp/ \
    --device gpu \
--use_amp False


```

- **model_name_or_path** 指示了某种特定配置的模型，对应有其预训练模型和预训练时使用的 tokenizer。若模型相关内容保存在本地，这里也可以提供相应目录地址。注：bert-base-uncased 等对应使用的预训练模型转自 huggingface/transformers

可以看到启动 Fine-tuning 任务以后，脚本会自动下载 bert-base-uncased 预训练模型，以及用于 Fine-tuning 的 bert-base-uncased-vocab.txt 数据集。

```
Downloading and preparing dataset glue/sst2 to /home/aistudio/.cache/huggingface/datasets/glue/sst2/1.0.0/b3964b7f1
489d233b36987f534947a98b792c280b034ce8509d9ce2f74027449...
Downloading data: 100%|                    | 7.44M/7.44M [00:00<00:00, 18.0MB/s]
Dataset glue downloaded and prepared to /home/aistudio/.cache/huggingface/datasets/glue/sst2/1.0.0/b3964b7f1489d233
b36987f534947a98b792c280b034ce8509d9ce2f74027449. Subsequent calls will reuse this data.
[2022-06-09 14:18:16,982] [    INFO] - Downloading https://bj.bcebos.com/paddle-hapi/models/bert/bert-base-uncased-
vocab.txt and saved to /home/aistudio/.paddlenlp/models/bert-base-uncased
[2022-06-09 14:18:16,982] [    INFO] - Downloading bert-base-uncased-vocab.txt from https://bj.bcebos.com/paddle-ha
pi/models/bert/bert-base-uncased-vocab.txt
100%|                                        | 226k/226k [00:00<00:00, 2.63MB/s]
100%|                                        | 68/68 [00:19<00:00,  3.42ba/s]
Reusing dataset glue (/home/aistudio/.cache/huggingface/datasets/glue/sst2/1.0.0/b3964b7f1489d233b36987f534947a98b7
92c280b034ce8509d9ce2f74027449)
100%|                                        | 1/1 [00:00<00:00,  2.34ba/s]
[2022-06-09 14:18:37,933] [    INFO] - Downloading https://bj.bcebos.com/paddlenlp/models/transformers/bert-base-un
cased.pdparams and saved to /home/aistudio/.paddlenlp/models/bert-base-uncased
[2022-06-09 14:18:37,934] [    INFO] - Downloading bert-base-uncased.pdparams from https://bj.bcebos.com/paddlenlp/
models/transformers/bert-base-uncased.pdparams
100%|                                        | 775M/775M [00:19<00:00, 42.4MB/s]
```

当训练任务到达预先设定的 step 轮数以后，便会停止训练，并且将.pdparam 格式的模型权重保存在 tmp 目录下。

```
global step 6303/6315, epoch: 2, batch: 2092, rank_id: 0, loss: 0.025963, lr: 0.0000000422, speed: 13.3599 step/s
global step 6304/6315, epoch: 2, batch: 2093, rank_id: 0, loss: 0.020290, lr: 0.0000000387, speed: 13.6808 step/s
global step 6305/6315, epoch: 2, batch: 2094, rank_id: 0, loss: 0.013629, lr: 0.0000000352, speed: 15.1960 step/s
global step 6306/6315, epoch: 2, batch: 2095, rank_id: 0, loss: 0.061183, lr: 0.0000000317, speed: 11.1823 step/s
global step 6307/6315, epoch: 2, batch: 2096, rank_id: 0, loss: 0.025866, lr: 0.0000000281, speed: 13.1975 step/s
global step 6308/6315, epoch: 2, batch: 2097, rank_id: 0, loss: 0.112942, lr: 0.0000000246, speed: 13.8721 step/s
global step 6309/6315, epoch: 2, batch: 2098, rank_id: 0, loss: 0.013249, lr: 0.0000000211, speed: 10.1835 step/s
global step 6310/6315, epoch: 2, batch: 2099, rank_id: 0, loss: 0.045254, lr: 0.0000000176, speed: 12.7596 step/s
global step 6311/6315, epoch: 2, batch: 2100, rank_id: 0, loss: 0.023305, lr: 0.0000000141, speed: 13.6231 step/s
global step 6312/6315, epoch: 2, batch: 2101, rank_id: 0, loss: 0.096776, lr: 0.0000000106, speed: 9.4435 step/s
global step 6313/6315, epoch: 2, batch: 2102, rank_id: 0, loss: 0.071986, lr: 0.0000000070, speed: 16.8990 step/s
global step 6314/6315, epoch: 2, batch: 2103, rank_id: 0, loss: 0.090073, lr: 0.0000000035, speed: 11.5432 step/s
global step 6315/6315, epoch: 2, batch: 2104, rank_id: 0, loss: 0.011474, lr: 0.0000000000, speed: 16.4419 step/s
eval loss: 0.053647, acc: 0.926605504587156, eval done total : 1.4521238803863525 s
[2022-06-09 14:33:09,083] [    INFO] - tokenizer config file saved in ./tmp/sst2_ft_model_6315.pdparams/tokenizer_c
onfig.json
[2022-06-09 14:33:09,083] [    INFO] - Special tokens file saved in ./tmp/sst2_ft_model_6315.pdparams/special_token
s_map.json
```

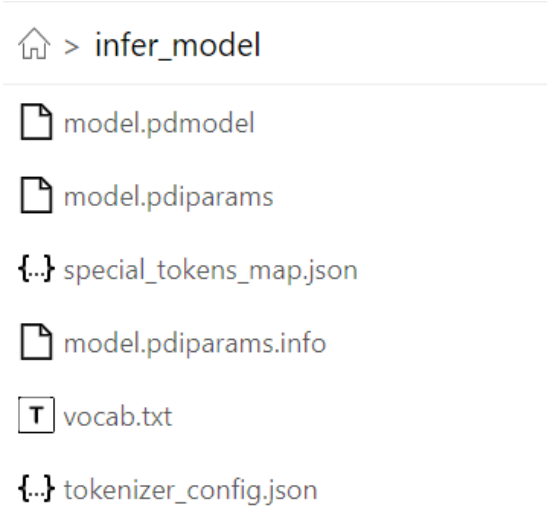
2.4. 步骤四：模型导出

在 Fine-tuning 完成后，我们可以使用如下方式导出希望用来预测的 Paddle 静态模型，并保存在 infer_model 路径下：

```
!python -u PaddleNLP/model_zoo/bert/export_model.py \
      --model_type bert \
      --model_path bert-base-uncased \
      --output_path ./infer_model/model
```

```
[2022-06-09 14:42:28,107] [    INFO] - Already cached /home/aistudio/.paddlenlp/models/bert-base-uncased/bert-base-un
cased.pdparams
W0609 14:42:28.108512  2546 gpu_context.cc:278] Please NOTE: device: 0, GPU Compute Capability: 7.0, Driver API Versi
on: 11.2, Runtime API Version: 10.1
W0609 14:42:28.113461  2546 gpu_context.cc:306] device: 0, cuDNN Version: 7.6.
```

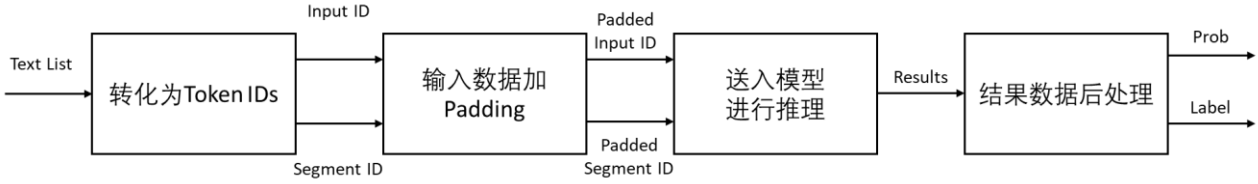
导出后的模型文件包含 **model.pdmodel, model.pdiparams.info, model.pdiparams**，推理时需要保证这三个文件在同一个目录下：



图：导出后的 Paddle BERT 静态模型文件

3. 部署部分

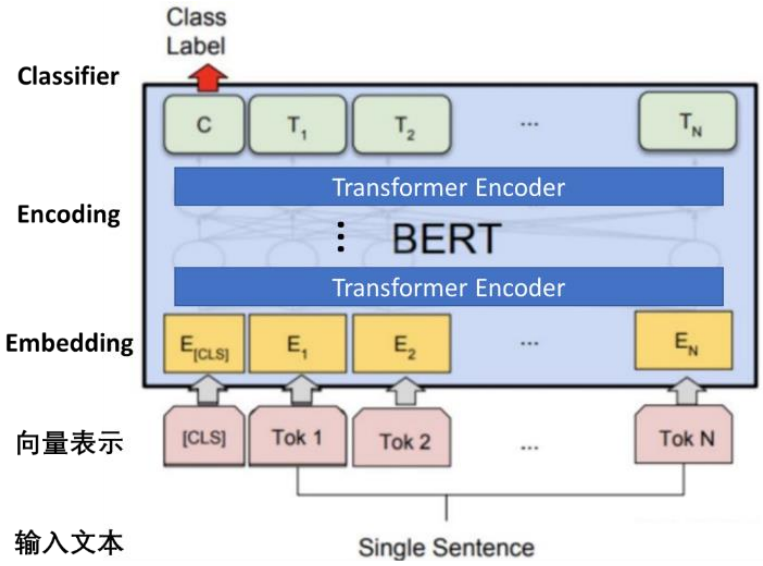
该示例将基于 OpenVINO™ 开发套件进行 Paddle 的静态模型部署，需要开发者提前准备好用于做部署的 Intel 平台硬件，可以是个人电脑，也可以是云服务器虚机。整体流程可以分为以下几个步骤：



图： BERT 模型部署流程

对于情感分析任务，BERT 网络的识别流程可以分成以下几个步骤：

- 输入语句文本，并转为相应的 Token ID
- 为每一行 Token ID 添加 Padding，使其保持长度一致
- Token ID 作为输入数据送入 BERT 模型进行推理 (模型内流程逻辑参考下图)，通过 Embedding Layer 将一个词映射成为固定维度的稠密向量，降维后的向量会再通过 Encoder 提取 Self-attentions 后的向量间的关系特征，最后经过 Classifier 对情感分类任务做出判断。
- 获取模型结果数据，通过后处理函数，计算分类标签与每一类标签的置信度



图： BERT for SST2 模型内部逻辑

BERT for SST2 的输入的编码向量（长度不固定）是 2 个嵌入特征的单位和，这 2 个词嵌入特征是：

- input_ids：输入文本被转化为 token 后的单个字的 id；
- segment_ids：就是句子级别（上下句）的标签，用于区分两个句子，例如 B 是否是 A 的下文（对话场景，问答场景等）。由于在情感分析任务中没有下句，所以这里 segment_ids 为全部为 0 的向量。

3.1. 步骤一：文本 Token 表示

定义数据转换模块，将原始的输入语句转化为 input_ids 与 segment_ids，作为输入数据。这边我们将会使用 PaddleNLP 自带的 tokenizer()方法进行转换。

```
# define the convert_examples_to_features method to get parameters from the input
def convert_example(text, tokenizer, max_seq_length=128):
    """
    Define the convert_examples_to_features method to get parameters from the input

    :param: text: a list of input sentance
            tokenizer: the pretrained tokenizers object
            max_seq_length: the maximum length of a sentance can be loaded into inference engine, a over-sized sentance will be clustered
    :retuns:
            input_ids: token embedding
            segment_ids: segment embedding
    """
    encoded_inputs = tokenizer(text=text, max_seq_len=max_seq_length)
    input_ids = encoded_inputs["input_ids"]
    segment_ids = encoded_inputs["token_type_ids"]

    return input_ids, segment_ids
```

3.2. 步骤二：Padding

需要保证 input_ids 与 segment_ids 数组在 axis0 方向的长度一致，由于这边 input_ids 与 segment_ids 均为一维数组，所以也可以不进行该操作。

```
Padding = Tuple(
    Pad(axis=0, pad_val=tokenizer.pad_token_id, dtype="int64"),  # input
    Pad(axis=0, pad_val=tokenizer.pad_token_id, dtype="int64")   # segment
)
# Seperates data into some batches.
batches = [
    examples[idx:idx + batch_size]
    for idx in range(0, len(examples), batch_size)
]
outputs = []
results = []
for batch in batches:
    input_ids, segment_ids = Padding(batch)
```

3.3. 步骤三：模型推理

部署代码里最**核心的部分**就是要定义基于 OpenVINO™ 开发套件的预测器，这里使用 CPU 作为模型的部署平台，可以看到通过 read_model 这个函数接口我们可以直接读取原始的.pdmodel 格式模型，省去了之前繁杂的离线转化过程。此外我们需要通过 compile_model 这个函数讲读取后的模型在指定的硬件平台进行加载和编译。最后创建 infer_request 推理请求进行推理任务部署。

```
ie = Core()
# Directly loading a Paddle format model
model = ie.read_model(model=model_path)
compiled_model = ie.compile_model(model=model, device_name="CPU")
request = compiled_model.create_infer_request()
output_layer = next(iter(model.outputs))
```

由于输入语句的长度往往不一致，这也导致编码后的向量长度也不一致，这里 OpenVINO™ 开发套件 CPU Plugin 的支持上已经全面**引入了 Dynamic Shape 功能**，无需再手动调整输入数据的长度，OpenVINO™ 开发套件会在 runtime 过程中自动匹配并动态申请一定的内存空间进行推理，优化性能表现。

```
request.infer(inputs={model.inputs[1].any_name: input_ids, model.inputs[0].any_name: segment_ids})
result = request.get_output_tensor(output_layer.index).data
```

由于新版 OpenVINO™ 开发套件已经**全面支持 Intel 12 代酷睿处理器**，为了取得更佳的推理性能，我们建议使用最新的硬件平台进行测试。

3.4. 步骤四：结果后处理

此处得到的结果数据为两种不同评价的可能性，我们需要将其通过 softmax 函数还原成百分比形式，并且找到可能性最大的那个评价序号所对应的标签（Positive，Negative）。

```
# Postprocessing to get the sentiment label and probility of each sentiment of each sentance
probs = softmax(result, axis=1)
idx = np.argmax(probs, axis=1)
idx = idx.tolist()
labels = [label_map[i] for i in idx]
outputs.extend(probs)
results.extend(labels)
```

最后我们找一组测试语句作为输入数据，将其封装成 List 以后，送入到识别器中进行识别，可以发现结果都是符合我们的先验预期的。

```
data = [
    'OpenVINO accelerates applications with high-performance, AI and deep learning inference deployed from edge to cloud',
    'The main disadvantage of padding is a bad performance due to spending time for processing dummy elements in the padding area',
    'Model Optimizer adjusts deep learning models for optimal execution on end-point target devices',
    'Reduce resource demands and efficiently deploy on a range of Intel® platforms from edge to cloud',
    'It may have some accuracy drop'
]
label_map = {0: 'negative', 1: 'positive'}
model_path = Path("model/model.pdmodel")
max_seq_length = 128
outputs, results = predict(model_path, max_seq_length, data, label_map)
# Print the predicted label and probilitiy of each sentiment
for idx, text in enumerate(data):
    print(
        'Data: {} \n Label: {} \n Negative prob: {} \n Positive prob: {} \n '.
        format(text, results[idx], outputs[idx][0], outputs[idx][1]))
```

该示例程序可以可以准确按 SST2 情感二分类任务要求，输出每段输入语句的分类情感标签，并获得每种情感对应的参考置信度。

```
Data: OpenVINO accelerates applications with high-performance, AI and deep learning inference deployed from edge to cloud
 Label: positive
 Negative prob: 0.0014771847054362297
 Positive prob: 0.9985228776931763

Data: The main disadvantage of padding is a bad performance due to spending time for processing dummy elements in the padding area
 Label: negative
 Negative prob: 0.9990200996398926
 Positive prob: 0.0009799102554097772

Data: Model Optimizer adjusts deep learning models for optimal execution on end-point target devices
 Label: positive
 Negative prob: 0.05097413808107376
 Positive prob: 0.9490258097648621

Data: Reduce resource demands and efficiently deploy on a range of Intel® platforms from edge to cloud
 Label: positive
 Negative prob: 0.0010080438805744052
 Positive prob: 0.9989920258522034

Data: It may have some accuracy drop
 Label: negative
 Negative prob: 0.7768430113792419
 Positive prob: 0.22315697371959686
```

小结：

作为发布至今近 4 年以来最大的一次更新，OpenVINO™ 2022.1 版本为了更好地支持 NLP 与语音相关的模型，在 CPU plugin 中已全面支持了动态 input shape，并通过与百度 PaddlePaddle 框架的深度集成，用更便捷的 API 接口，更丰富的模型支持，提升双方开发者在模型部署侧的使用体验，真正实现对 PaddleNLP 模型的**“无缝”转化与部署**。

通过本次的全流程示例，我们看到 OpenVINO™ 开发套件对 Paddle BERT 模型已经做到了很好的适配，从而加速在 Intel 平台上的推理。以下 github repository 中已为大家提前准备好了 OpenVINO™ 开发套件部署的参考实现与.pdmodel 格式的 BERT 预训练模型。

https://github.com/OpenVINO-dev-contest/openvino_notebooks/tree/PaddleBert/notebooks/005-hello-paddle-nlp

除此之外，为了方便大家了解并快速掌握 OpenVINO™ 开发套件的使用，我们还提供了一系列开源的 Jupyter notebook demo。运行这些 notebook，就能快速了解在不同场景下如何利用 OpenVINO™ 开发套件实现一系列、包括 OCR 在内的、计算机视觉及自然语言处理任务。

OpenVINO™ notebooks 的资源可以在 Github 这里下载安装：
https://github.com/openvinotoolkit/openvino_notebooks