AiSpace

Release 0.1.0

Jul 28, 2021

Notes

AiSpace provides highly configurable framework for deep learning model development, deployment and conveniently use of pre-trained models (bert, albert, opt, etc.).

CHAPTER 1

Features

- Highly configurable, we manage all hyperparameters with inheritable Configuration files.
- All modules are registerable, including models, dataset, losses, optimizers, metrics, callbacks, etc.
- Standardized process
- Multi-GPU Training
- Integrate multiple pre-trained models, including chinese
- Simple and fast deployment using BentoML
- Integrated Chinese benchmarks CLUE

1.1 Quickstart

1.1.1 Training

```
python -u aispace/trainer.py \
    --schedule train_and_eval \
    --config_name CONFIG_NAME \
    --config_dir CONFIG_DIR \
    [--experiment_name EXPERIMENT_NAME] \
    [--model_name MODEL_NAME] \
    [--gpus GPUS]
```

1.1.2 Training with resumed model

```
python -u aispace/trainer.py \
    --schedule train_and_eval \
    --config_name CONFIG_NAME \
```

(continues on next page)

(continued from previous page)

```
    --config_dir CONFIG_DIR \
    --model_resume_path MODEL_RESUME_PATH \
    [--experiment_name EXPERIMENT_NAME] \
    [--model_name MODEL_NAME] \
    [--gpus GPUS]
```

–model_resume_path is a path to initialization model.

1.1.3 Average checkpoints

```
python -u aispace/trainer.py \
    --schedule avg_checkpoints \
    --config_name CONFIG_NAME \
    --config_dir CONFIG_DIR \
    --prefix_or_checkpoints PREFIX_OR_CHECKPOINGS \
    [--ckpt_weights CKPT_WEIGHTS] \
    [--experiment_name EXPERIMENT_NAME] \
    [--model_name MODEL_NAME] \
    [--gpus GPUS]
```

–prefix_or_checkpoints is paths to multiple checkpoints separated by comma.

–ckpt_weights is weights same order as the prefix_or_checkpoints.

1.1.4 Deployment

Generate deployment files before deployment, you need to specify the model path (–model_resume_path) to be deployed like following.

```
python -u aispace/trainer.py \
    --schedule deploy \
    --config_name CONFIG_NAME \
    --config_dir CONFIG_DIR \
    --model_resume_path MODEL_RESUME_PATH \
    [--experiment_name EXPERIMENT_NAME] \
    [--model_name MODEL_NAME] \
    [--gpus GPUS]
```

We use BentoML as deploy tool, so your must implement the ***deploy*** function in your model class.

1.1.5 Output file structure

The default output path is ***save***, which may has multiple output directories under name as:

```
{experiment_name}_{model_name}_{dataset_name}_{random_seed}_{id}
```

Where ***id*** indicates the sequence number of the experiment for the same task, increasing from 0.

Take the text classification task as an example, the output file structure is similar to the following:

```
test_bert_for_classification_119_0
├── checkpoint                  # 1. checkpoints
│   ├── checkpoint
```

(continues on next page)

(continued from previous page)

```
│   ├── ckpt_1.data-00000-of-00002
│   ├── ckpt_1.data-00001-of-00002
│   ├── ckpt_1.index
|   ...
├── deploy                      # 2. Bentoml depolyment directory
│   └── BertTextClassificationService
│       └── 20191208180211_B6FC81
├── hparams.json                # 3. Json file of all hyperparameters
├── logs                        # 4. general or tensorboard log directory
│   ├── errors.log              # error log file
│   ├── info.log                # info log file
│   ├── train
│   │   ├── events.out.tfevents.1574839601.jshd-60-31.179552.14276.v2
│   │   ├── events.out.tfevents.1574839753.jshd-60-31.profile-empty
│   └── validation
│       └── events.out.tfevents.1574839787.jshd-60-31.179552.151385.v2
├── model_saved                 # 5. last model saved
│   ├── checkpoint
│   ├── model.data-00000-of-00002
│   ├── model.data-00001-of-00002
│   └── model.index
└── reports                     # 6. Eval reports for every output or task
    └── output_1_classlabel     # For example, text classification task
        ├── confusion_matrix.txt
        ├── per_class_stats.json
        └── stats.json
```

1.2 Configuration

We use yaml to manage various configurations, and inheritance and override can be implemented between configurations The configurations of specific task inherit base configuration directly or indirectly.

1.2.1 Base

```
configs/base.yml
```

This is the most basic configuration file, including the default configuration about training, logging, etc.

You can read this configuration carefully to understand the possibility of configurability.

1.2.2 Pretrain

```
configs/pretrain
```

This kind of configuration file adds pretrained item compared to other configurations mainly, and includes ***base.yml***.

1.2.3 Specific task

For example:

```
configs/glue_zh/tnews.yml
```

1.3 Dataset

1.3.1 BaseDataset

The base class ***BaseDataset*** inherits the ***tfds.core.GeneratorBasedBuilder*** and ***Registry***, which makes subclasses registerable.

1.3.2 Custom dataset

Take the glue_zh dataset as an example as following:

```
@BaseDataset.register("glue_zh")
class GlueZh(BaseDataset):
    ...
```

The development follows ***tensorflow_dataset's*** specification.

1.4 Model

1.4.1 BaseModel

The base class ***BaseModel*** inherits the ***tf.keras.Model*** and ***Registry***, which makes subclasses registerable. It also implements deploy method for helping generate deployment files.

1.4.2 Custom Models

Take the bert_for_classification model as an example as following:

```
@BaseModel.register("bert_for_classification")
class BertForSeqClassification(BaseModel):
    ...
```

The registered name of the model BertForSeqClassification is bert_for_classification. And the implementation of other functions follows ***tf.keras.Model's*** specification.

1.5 Deployment

We use BentoML as deploy tool, so your must implement the ***deploy*** function in your model class and a **bentoml service class**.

Take model ***bert_for_classification*** as an example:

1.5.1 Custom bentoml service

For more detailed information, please visit BentoML.

```
__all__ = [
    "BertTextClassificationService"
]

import os, sys
import tensorflow as tf

sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(os.path.abspath(__
↪file__)), "../" * 4)))

from bentoml import api, env, BentoService, artifacts
from bentoml.artifact import TensorflowSavedModelArtifact, PickleArtifact
from bentoml.handlers import JsonHandler

import numpy as np
from scipy.special import softmax

from aispace.datasets.tokenizer import BertTokenizer
from aispace.utils.hparams import Hparams

@artifacts([
        TensorflowSavedModelArtifact('model'),
        PickleArtifact('tokenizer'),
        PickleArtifact("hparams"),
    ])
@env(pip_dependencies=['tensorflow-gpu==2.0.0', 'numpy==1.16', 'scipy==1.3.1',
↪"tensorflow-datasets==1.3.0"])
class BertTextClassificationService(BentoService):

    def preprocessing(self, text_str):
        input_ids, token_type_ids, attention_mask = self.artifacts.tokenizer.
↪encode(text_str)
        return input_ids, token_type_ids, attention_mask

    def decode_label_idx(self, idx):
        return self.artifacts.hparams.dataset.outputs[0].labels[idx]

    @api(JsonHandler)
    def title_predict(self, parsed_json):
        input_data = {
            "input_ids": [], "token_type_ids": [], "attention_mask": []
        }
        if isinstance(parsed_json, (list, tuple)):
            pre_input_data = list(zip(*list(map(self.preprocessing, parsed_json))))
            input_data['input_ids'].extend(pre_input_data[0])
            input_data['token_type_ids'].extend(pre_input_data[1])
            input_data['attention_mask'].extend(pre_input_data[2])
        else:  # expecting type(parsed_json) == dict:
            pre_input_data = self.preprocessing(parsed_json['text'])
            input_data['input_ids'].append(pre_input_data[0])
            input_data['token_type_ids'].append(pre_input_data[1])
            input_data['attention_mask'].append(pre_input_data[2])

        input_data['input_ids'] = tf.constant(input_data['input_ids'], name="input_ids
↪")
```

(continues on next page)

(continued from previous page)

```
        input_data['token_type_ids'] = tf.constant(input_data['token_type_ids'], name=
↪"token_type_ids")
        input_data['attention_mask'] = tf.constant(input_data['attention_mask'], name=
↪"attention_mask")
        prediction = self.artifacts.model(input_data, training=False)
        prediction_normed = softmax(prediction[0].numpy(), -1)
        prediction_idx = np.argmax(prediction_normed, -1).tolist()
        prediction_confidence = np.max(prediction_normed, -1).tolist()
        ret = {
            "predictions": []
        }
        for idx, confidence in zip(prediction_idx, prediction_confidence):
            cur_label = self.decode_label_idx(idx)
            new_ret = {
                "label": cur_label,
                "confidence": confidence
            }
            ret["predictions"].append(new_ret)

        return ret
```

1.5.2 Deploy function

Deploy function in model class as following

```
def deploy(self):
    """Return path of deployment files"""
    from aispace.datasets.tokenizer import BertTokenizer
    from .bento_services import BertTextClassificationService
    tokenizer = BertTokenizer(self._hparams.dataset.tokenizer)
    bento_service = \
        BertTextClassificationService.pack(
            model=self,
            tokenizer=tokenizer,
            hparams=self._hparams,
        )
    saved_path = bento_service.save(self._hparams.get_deploy_dir())
    return saved_path
```

1.5.3 Generate deployment files

To generate deployment files, you need to specify the model path (–model_resume_path) to be deployed and run following script.

```
python -u aispace/trainer.py \
    --schedule deploy \
    --config_name CONFIG_NAME \
    --config_dir CONFIG_DIR \
    --model_resume_path MODEL_RESUME_PATH \
    [--experiment_name EXPERIMENT_NAME] \
    [--model_name MODEL_NAME] \
    [--gpus GPUS]
```

1.6 Examples

1.6.1 glue_zh/tnews

Tnews is a task of Chinese GLUE, which is a short text classification task from ByteDance.

Run Tnews classification

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --config_name tnews \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 2 3  \
```

Specify different pretrained model, please change ***includes*** and ***pretrained.name*** in config file.

NOTE: The hyper-parameters used here have not been fine-tuned.

1.6.2 glue_zh/cmrc2018

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_qa \
    --schedule train_and_eval \
    --enable_xla False \
    --config_name cmrc2018 \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 2 3 \
    > err.log 2>&1 &
```

1.6.3 glue_zh/csl

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --config_name csl \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > csl_err.log 2>&1 &
```

1.6.4 glue_zh/drcd

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_qa \
    --schedule train_and_eval \
    --enable_xla False \
    --config_name drcd \
```

(continues on next page)

(continued from previous page)

```
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > drcd_err.log 2>&1 &
```

1.6.5 glue_zh/afqmc

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --config_name afqmc \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > afqmc_err.log 2>&1 &
```

1.6.6 glue_zh/iflytek

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --config_name iflytek \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > iflytek_err.log 2>&1 &
```

1.6.7 glue_zh/cmnli

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --config_name cmnli \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > cmnli_err.log 2>&1 &
```

1.6.8 glue_zh/wsc

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_relation_extract \
    --schedule train_and_eval \
    --config_name wsc \
    --config_dir ./configs/glue_zh \
    --gpus 0 1 \
    > wsc_err.log 2>&1 &
```

1.6.9 dureader/robust

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_qa \
    --schedule train_and_eval \
    --enable_xla False \
    --config_name dureader_robust \
    --config_dir ./configs/qa \
    --gpus 0 1 \
    > err.log 2>&1 &
```

|Model|F1|EM| |—|—|—| |bert-base-chinese-huggingface|66.624|51.856| |chinese_wwm|67.007|53.434| |chinese_roberta_wwm_ext|65.521|50.274| |ERNIE_stable-1.0.1|75.268|61.675| |ERNIE_1.0_max-len-512|**83.609**|**72.328**|

1.6.10 dureader/yesno

```
python -u aispace/trainer.py \
    --experiment_name test \
    --model_name bert_for_classification \
    --schedule train_and_eval \
    --enable_xla False \
    --config_name dureader_yesno \
    --config_dir ./configs/qa \
    --gpus 0 1 \
    > err.log 2>&1 &
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search