AutoGluon.Multimodal 0.6.0

Installation

AutoGluon (GitHub) requires pip > 1.4 (upgrade by pip install -U pip). More installation options. AutoGluon supports Python 3.7 to 3.9. Installation is available for Linux, MacOS, and Windows.

```
pip install autogluon
```

Classification & Regression

MultiModalPredictor finetunes foundation models for solving classification and regression problems with image, text, and tabular features. Here, we use a simplified version petfinder_for_tutorial from the PetFinder dataset. MultiModalPredictor automatically analyzes the columns in the input dataframe to detect categorical, numerical, text, and images (stored as paths).

```
import pandas as pd
train_data = pd.read_csv(
    'petfinder_for_tutorial/train.csv', index_col=0)
test_data = pd.read_csv(
    'petfinder_for_tutorial/test.csv', index_col=0)
```

To train the model, just call '.fit()'. We also support customization (docs).

```
from autogluon.multimodal import MultiModalPredictor
predictor = MultiModalPredictor(
    problem_type="classification",
    label="AdoptionSpeed"
)
predictor.fit(train_data)
predictor.fit_summary() # Summarize the model
```

To evaluate on the validation data and run inference

```
predictor.evaluate(test_data)  # Evaluation
predictor.predict(test_data)  # Inference
```

Named-Entity Recognition

MultiModalPredictor supports named-entity recognition. We use MIT movies corpus to demonstrate the usage, which can be downloaded from train.csv and test.csv.

```
from autogluon.multimodal import MultiModalPredictor
predictor = MultiModalPredictor(
    problem_type="ner", label="entity_annotations"
)
# Train model
predictor.fit("train.csv")
# Evaluation
predictor.evaluate("test.csv")

# Inference
sentence = "Game of Thrones is an American fantasy "
           "drama television series created"
           "by David Benioff"
predictions = predictor.predict({
    'text_snippet': [sentence]
})
```

Object Detection

To use MultiModalPredictor for object detection, please first install additional dependencies by

```
mim install mmcv-full
pip install mmdet
pip install pycocotools-windows # if windows users
```

MultiModalPredictor supports common object detection data formats such as VOC and COCO. Here we use the dataset tiny_motorbike_coco to demonstrate how to use MultiModalPredictor. The predictor natively supports json files in the COCO-format.

```
train_path = "./Annotations/trainval_cocoformat.json"
test_path = "./Annotations/test_cocoformat.json"

from autogluon.multimodal import MultiModalPredictor
model = "faster_rcnn_r50_fpn_2x_coco"
predictor = MultiModalPredictor(
    problem_type="object_detection",
    # Set the backbone. This is optional.
    hyperparameters={
        "model.mmdet_image.checkpoint_name": model,
    },
    sample_data_path=train_path,
)
```

To train an object detector,

```
predictor.fit(train_path)
```

More options for the **fit** method (docs):

```
predictor.fit(train_path,
    # Limit the training time, in second
    time_limit=600,
    # Use a separate dataset to tune models.
    tuning_data=val_data
)
```

Once trained, we can evaluate it on the test set to obtain metrics,

```
predictor.evaluate(test_path)
```

We can also predict on any image and visualize the detected bounding boxes with its confidence scores,

```
pred = predictor.predict({"image": [test_image]})
```

Matching

MultiModalPredictor implements a flexible twin-tower architecture that can solve text-text, image-image, and text-image matching problems (docs). Example of extracting embeddings for semantic text matching:

```
!pip install ir_datasets # Install dataset package
from autogluon.multimodal import MultiModalPredictor
from autogluon.multimodal import utils
import ir_datasets
import pandas as pd
dataset = ir_datasets.load("beir/fiqa/dev")
docs_df = pd.DataFrame(dataset.docs_iter()) \
            .set_index("doc_id")
model = "sentence-transformers/all-MiniLM-L6-v2"
predictor = MultiModalPredictor(
    problem_type="text_similarity",
    hyperparameters={
        "model.hf_text.checkpoint_name": model,
    }
)
doc_embedding = predictor.extract_embedding(docs_df)
q_embedding = predictor.extract_embedding([
    "what happened when the dot com bubble burst?"
])
similarity = utils.compute_semantic_similarity(
    q_embedding, doc_embedding
)
# Get the most relevant document
docs_df['text'].iloc[similarity.argmax().numpy()]
```

In addition, you can finetune the matching model via relevance data. Here, we demonstrate the use-case via the Flickr30K image-text matching dataset preprocessed in the dataframe format: flickr30k.zip.

```
import pandas as pd
train_data = pd.read_csv("train.csv", index_col=0)
tdata = pd.read_csv("test.csv", index_col=0)
```

To finetune model, just specify the "query" and "response" keys when creating predictor and pick "image_text_similarity" as problem type.

```
from autogluon.multimodal import MultiModalPredictor
predictor = MultiModalPredictor(
            query="caption",
            response="image",
            problem_type="image_text_similarity",
        )
predictor.fit(train_data, time_limit=180)
e_i = predictor.extract_embedding(tdata["image"])
e_t = predictor.extract_embedding(tdata["caption"])
```

- MultiModalPredictor also supports features like knowledge distillation (docs), efficient finetuning (docs), and HPO (docs).
- For deployment, check Tutorial.
- For other use-cases, check TabularPredictor and TimeSeriesPredictor.
- Check the latest version of this cheat sheet.
- Any questions? Ask here
- Like what you see? Consider starring AutoGluon on GitHub and following us on twitter to get notified of the latest updates!