Направление Data Science, профессия ML-инженер NLP-направления MTS AI

Пример решения задания 1 от эксперта

```
# Установка PyTorch 2.0 (cuda 11.7)

!pip install "torch>=2.0" --extra-index-url https://download.pytorch.org/whl/cu117 --upgrade --quiet

# Установка transformers и dataset

!pip install "transformers==4.27.1" "datasets==2.9.0" "accelerate==0.17.1" "evaluate==0.4.0" tensorboard
scikit-learn # Установка git-lfs для загрузки модели и логов в hugging face hub !sudo apt-get install git-lfs
--yes

from huggingface_hub import login
login(
        token="", # ADD YOUR TOKEN HERE
        add_to_git_credential=True
)

# Загрузка датасета

from datasets import load_dataset

# Dataset id from huggingface.co/dataset

dataset_id = "massive"

# Load raw dataset

raw_dataset = load_dataset(dataset_id)

print(f"Train dataset size: {len(raw_dataset['train'])}")
print(f"Test dataset size: {len(raw_dataset['test'])}")

from transformers import AutoTokenizer

# Model id to load the tokenizer

model_id = "bert-base-uncased"

# Load Tokenizer

tokenizer = AutoTokenizer.from_pretrained(model_id)

# Tokenize helper function

def tokenize(batch):
     return tokenizer(batch['text'], padding='max_length', truncation=True, return_tensors="pt")

# Tokenize dataset

raw_dataset = raw_dataset.rename_column("label", "labels") # to match Trainer

tokenized_dataset = raw_dataset.map(tokenize, batched=True,remove_columns=["text"])
print(tokenized_dataset["train"].features.keys())

from transformers import AutoModelForSequenceClassification

# Model id to load the tokenizer
```

```
model_id = "bert-base-uncased"

# Prepare model labels - useful for inference

labels = tokenized_dataset["train"].features["labels"].names
num_labels = len(labels) label2id, id2label = dict(), dict()

for i, label in enumerate(labels):
     label2id[label] = str(i)
     id2label[str(i)] = label

# Download the model from huggingface.co/models

model = AutoModelForSequenceClassification.from_pretrained(
     model_id, num_labels=num_labels, label2id=label2id, id2label=id2label
)

import evaluate
import numpy as np

# Metric ID

metric = evaluate.load("f1")

# Metric helper method

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return metric.compute(predictions=predictions, references=labels, average="weighted")

from huggingface_hub import HfFolder
from transformers import Trainer, TrainingArguments
# ID for remote repository
repository_id = "bert-base-massive"
# Define training args
training_args = TrainingArguments(
    output_dir=repository_id,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=8,
    learning_rate=5e-5,
    num_train_epochs=3,
    # PyTorch 2.0 specifics
    bf16=True, # bfloat16 training
    torch_compile=True, # optimizations
    optim="adamw_torch_fused", # improved optimizer
    # logging & evaluation strategies
    logging_dir=f"{repository_id}/logs",
    logging_strategy="steps",
    logging_steps=200,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    # push to hub parameters
    report_to="tensorboard",
    push_to_hub=True,
    hub_strategy="every_save",
    hub_model_id=repository_id,
    hub_token=HfFolder.get_token(),
)

# Create a Trainer instance
trainer = Trainer(
     model=model,
     args=training_args,
     train_dataset=tokenized_dataset["train"],
     eval_dataset=tokenized_dataset["test"],
     compute_metrics=compute_metrics,
)
```

Пример решения задания 2 от эксперта

Описание данных и методов, с помощью которых можно доработать модель:

- Провести обогащение данных при помощи других открытых датасетов, например данными с Kaggle.
- Реализовать бейзлайн подход с использованием MaxSoftmaxProb - оценка OOD осуществляется с уже обученной моделью за счет установления порога отнесения примера к OOD по валидационной выборке.

Варианты, как можно оповещать пользователей о том, что их запрос не относится ни к одному из тех классов, которые модель умеет определять:

- Всплывающее окно с просьбой переформулировать запрос.
- Форма, куда пользователь может написать уточняющие вопросы для более четкой формулировки запроса.
- Дать ссылку на другую страницу, где собраны референсы из числа определяемых интентов, которыми можно дополнить свой запрос.