

1. Конвертируем csv в jsonl

```
import pandas as pd
import json

df = pd.read_csv("your_data.csv", sep=';', quotechar='"',
encoding='utf-8')

required_columns = {'name', 'description', 'precondition',
'scenario', 'expected_result'}
if missing := required_columns - set(df.columns):
    raise ValueError(f"Missing columns: {missing}")

def safe_get(row, col, default=''):
    return row[col] if pd.notna(row[col]) else default

with open("ollama_dataset.jsonl", "w", encoding="utf-8") as f:
    for _, row in df.iterrows():
        prompt = f"Создай тест-кейс для: {safe_get(row,
'description')}}"

        completion = "\n".join([
            f"Название: {safe_get(row, 'name', 'Без названия')}}",
            f"Предусловия: {safe_get(row, 'precondition')}}",
            f"Сценарий: {safe_get(row, 'scenario')}}",
            f"Ожидаемый результат: {safe_get(row,
'expected_result')}}"
        ])

        entry = {
            "text": f"<s>[INST] {prompt} [/INST] {completion} </s>"
        }

        f.write(json.dumps(entry, ensure_ascii=False) + "\n")
```

2. Устанавливаем библиотеки

pip install transformers datasets peft accelerate torch

3. Подготавливаем данные (to_hf_dataset.py)

```
import pandas as pd
from datasets import Dataset
```

```
def prepare_dataset():
    df = pd.read_csv("your_data.csv", sep=';', encoding='utf-8')

    def generate_text(row):
        return f"""<s>[INST] Создай тест-кейс для:
{row['description']} [/INST]
Название: {row['name']}
Предварительные условия: {row['precondition']}
Сценарий: {row['scenario']}
Ожидаемый результат: {row['expected_result']}</s>"""

    df['text'] = df.apply(generate_text, axis=1)
    return Dataset.from_pandas(df[['text']])
```

4. получить токен в <https://huggingface.co/settings/tokens>) с правами write

5. Аутентификация в Hugging Face

Ввести huggingface-cli login

и ввести токен полученный в п4

6. Обучаем (train.py)

```
from transformers import AutoModelForCausalLM, AutoTokenizer,
TrainingArguments, Trainer, \
    DataCollatorForLanguageModeling
from peft import LoraConfig, get_peft_model
from datasets import load_from_disk
import torch
from to_hf_dataset import prepare_dataset

# Подготовка данных
dataset = prepare_dataset()
dataset.save_to_disk("hf_dataset")

# Загрузка модели (например, Mistral)
model =
AutoModelForCausalLM.from_pretrained("mistralai/Mistral-7B-Instruct-v0.2")
tokenizer =
AutoTokenizer.from_pretrained("mistralai/Mistral-7B-Instruct-v0.2"
)
tokenizer.pad_token = tokenizer.eos_token
```

```

# Настройка LoRA
peft_config = LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

# Токенизация
def tokenize_fn(examples):
    return tokenizer(examples["text"], truncation=True,
max_length=512)

dataset = dataset.map(tokenize_fn, batched=True)

# Параметры обучения
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=3,
    per_device_train_batch_size=2,
    learning_rate=1e-5,
    fp16=True,
    logging_steps=10,
    save_strategy="no"
)

# Запуск обучения
model = get_peft_model(model, peft_config)
model.print_trainable_parameters() # Должно показать ~0.1%
параметров

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    data_collator=DataCollatorForLanguageModeling(tokenizer,
mlm=False)
)

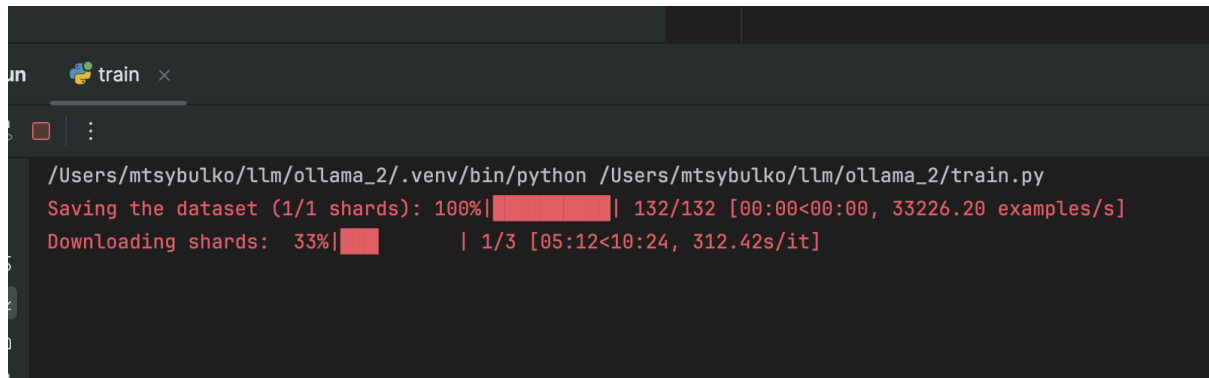
trainer.train()
model.save_pretrained("./finetuned_model")

```

7. запускаем одучение (запустить скрипт из п.4)

ОБУЧЕНИЕ ИДЕТ ОЧЕНЬ ДОЛГО!!! (надо баловаться с настройками)

Будем видеть такую вещь

A screenshot of a terminal window with a dark background. The window has a tab labeled 'train' with a small icon and a close button. The terminal shows the execution of a Python script. The first line is the command: `/Users/mtsybulko/llm/ollama_2/.venv/bin/python /Users/mtsybulko/llm/ollama_2/train.py`. The second line shows progress for saving the dataset: `Saving the dataset (1/1 shards): 100% |██████████| 132/132 [00:00<00:00, 33226.20 examples/s]`. The third line shows progress for downloading shards: `Downloading shards: 33% |███████| 1/3 [05:12<10:24, 312.42s/it]`.

```
un  train ×  
$ /Users/mtsybulko/llm/ollama_2/.venv/bin/python /Users/mtsybulko/llm/ollama_2/train.py  
Saving the dataset (1/1 shards): 100% |██████████| 132/132 [00:00<00:00, 33226.20 examples/s]  
Downloading shards: 33% |███████| 1/3 [05:12<10:24, 312.42s/it]
```

8. Конвертация в формат Ollama. После обучения создаем Modelfile:

Это докер

FROM mistral

ADAPTER /путь до finetuned_model

9. Собрать модель (команда в терминале

`ollama create mymodel -f ./Modelfile`