FunQA Contest

一、数据集方案：

英文版 + 中文版

4300

3000

500个val

800个test

400个放出video和anwser

400个只放出video

二、投稿 + 写一个赛题的setting的文档

三、evaluation的代码

四、复现一遍otter的训练过程

五、确定json格式

六、确定评价标准

以下为训练与测评流程：

【注意】有关huggingface网络连接的问题，自行挂梯子可解决。

部署基本环境

```
git clone https://github.com/Luodian/Otter.git
conda env create -f environment.yml
conda activate otter
pip install -r requirement.txt
pip install wandb
```

下载预训练模型

方法一：从Onedrive链接下载。

方法二：通过huggingface下载。运行download.py。

```
from tqdm import tqdm
from otter.modeling_otter import OtterForConditionalGeneration

if __name__ == "__main__":
    model = OtterForConditionalGeneration.from_pretrained(
        "huggingface模型地址", device_map="sequential", cache_dir='自定义本地缓存地址'
    )

model = OtterForConditionalGeneration.from_pretrained(
        "luodian/OTTER-9B-INIT", device_map="sequential", cache_dir='/root/autodl-tmp/
hgcache/huggingface/hub'
    )
```

准备训练用JSON文件

我们参考 [Otter](https://github.com/Luodian/Otter) 模型的官方训练数据格式，分别为XXX.json, XXX_instructions.json, XXX_train.json。

可见例子（TVC.json, TVC_instructions.json, TVC_train.json）或小版本例子（TVC_mini.json, TVC_instructions_mini.json, TVC_train_mini.json）

XXX.json

存储图片信息，格式为：

```
["图片id": "图片信息（以base64格式储存）", "帧数id":"图片信息"]
```

例如：

```
["TVC_IMG_castle_s04e20_seg02_clip_13_00009": "iVBORw0KGgoAAA...此处省略n行",
 ""TVC_IMG_castle_s04e20_seg02_clip_13_00026": "iVBORw0KGgoAAAANSUhE...此处省略n行"]
```

XXX_instructions.json

存储每一个问答对的信息，格式为：

```
{
  "meta": {
    "version": "版本号",
    "time": "时间",
    "author": "作者"
  },
  "data": {
      # 问答对编号
    "TVC_INS_000002": {
      # 问题
      "instruction": "What does the character Castle do after standing up?",
      # 回答
      "answer": "After standing up, the character Castle walks towards the board, sugg
esting that he might be preparing to present or explain something to others.",
      # 本问题对应视频的所有帧数的图片id（与XXX.json内的需一致）
      "image_ids": [
        "TVC_IMG_castle_s04e20_seg02_clip_13_00009",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00026",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00043",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00060",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00077",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00094",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00111",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00128",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00145",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00162",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00179",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00196",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00213",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00230",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00247",
        "TVC_IMG_castle_s04e20_seg02_clip_13_00264"
      ],
      # 关联问答对（自定义，需要出现在XXX_instructions.json中）
      "rel_ins_ids": [
        "TVC_INS_000003"
      ]
    },
    ...
}
```

具体见示例。

XXX_train.json

需要准备好的XXX_instrctions.jons文件，由generation_train_json.py自动生成：

```
from tqdm import tqdm
import orjson


def make_a_train(input_file, output_file):
    # Load the JSON file
```

```
    with open(input_file, "rb") as file:
        data = orjson.loads(file.read())

    # Create a set to store seen keys
    seen_keys = set()

    # Create a new dictionary with the keys from the original JSON and rel_ins_ids as
 values
    new_dict = {}
    for key, value in tqdm(data["data"].items()):
        if key not in seen_keys:
            try:
                # Check if rel_ins_ids are in the original JSON
                valid_rel_ins_ids = [rel_ins_id for rel_ins_id in value["rel_ins_ids"]
if rel_ins_id in data["data"]]

                # Add the valid rel_ins_ids to the new_dict
                new_dict[key] = valid_rel_ins_ids
                seen_keys.update(valid_rel_ins_ids)
            except Exception as e:
                print("Error with key %s and value %s" % (key, value))

    # Write the new dictionary to a new JSON file
    with open(output_file, "wb") as file:
        file.write(orjson.dumps(new_dict))


make_a_train("XXX_instructions.json", "XXX_train.json")
```

开始train

修改train.sh文件

```
export PYTHONPATH=.

accelerate launch --config_file=./pipeline/accelerate_configs/accelerate_config_fsdp.y
aml \
pipeline/train/instruction_following.py \
--pretrained_model_name_or_path=luodian/OTTER-9B-DenseCaption  \
--mimicit_path="/root/autodl-tmp/FunQA_competition/data/TVC/TVC_instructions_mini.jso
n" \
--images_path="/root/autodl-tmp/FunQA_competition/data/TVC/TVC_mini.json" \
--train_config_path="/root/autodl-tmp/FunQA_competition/data/TVC/TVC_train_mini.json"
 \
--batch_size=1 \
--num_epochs=9 \
--report_to_wandb \
--wandb_entity=ntu-slab \
--run_name=otter9B_dense_caption \
--wandb_project=otter9B \
--workers=1 \
--cross_attn_every_n_layers=4 \
--lr_scheduler=cosine \
```

```
--learning_rate=1e-5 \
--warmup_steps_ratio=0.01
```

例如：

```
export PYTHONPATH=.

accelerate launch --config_file=./pipeline/accelerate_configs/accelerate_config_fsdp.y
aml \
pipeline/train/instruction_following.py \
--pretrained_model_name_or_path=luodian/OTTER-9B-INIT  \
--mimicit_path="path/to/DC_instruction.json" \
--images_path="path/to/DC.json" \
--train_config_path="path/to/DC_train.json" \
--batch_size=4 \
--num_epochs=9 \
--report_to_wandb \
--wandb_entity=ntu-slab \
--run_name=otter9B_dense_caption \
--wandb_project=otter9B \
--workers=1 \
--lr_scheduler=cosine \
--learning_rate=1e-5 \
--warmup_steps_ratio=0.01 \
```

修改 Otter/pipeline/train/instruction_following.py

如果在下载预模型的阶段修改了缓存地址，需要在instruction_following.py里341行的model = OtterForConditionalGeneration.from_pretrained()中添加**cache_dir**参数，例如：

```
model = OtterForConditionalGeneration.from_pretrained(
               args.pretrained_model_name_or_path,
               device_map="auto",
               local_files_only=args.offline,
               cache_dir='/root/autodl-tmp/hgcache/huggingface/hub', #（新加的行）
           )
```

一切就绪，启动train.sh脚本开始训练

```
sudo sh train.sh
```

待模型加载完毕后正式进入训练。

测评

部署环境：

```
git clone https://github.com/google-research/bleurt.git
cd bleurt
conda activate bleurt python=3.9
pip install .
pip install pycocoevalcap nltk rouge
pip install openai

# download BLEURT offical checkpoint
wget https://storage.googleapis.com/bleurt-oss-21/BLEURT-20.zip .
unzip BLEURT-20.zip
```

在此路径下导入ground_truth.json和submission.json，

修改eval.py中的44行，改为自己的openai key，运行eval.py脚本。

eval.py的参数意义为：

—submission 提交json文件路径

—answer 官方答案json文件路径

—total_score 最终打分文件生成路径

—run_time BLEURT打分时间（默认60）

```
from pycocoevalcap.cider.cider import Cider
from pycocoevalcap.tokenizer.ptbtokenizer import PTBTokenizer
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
from rouge import Rouge
from nltk.translate.meteor_score import meteor_score
import nltk
import pandas as pd
from tqdm import tqdm
# nltk.download('wordnet')
import os
import time
import openai
import json
import re


def calculate_bleu4(reference, hypothesis):
    # 将字符串分词为列表
    ref_tokens = reference.split()
    hyp_tokens = hypothesis.split()
    blue4_score = sentence_bleu([ref_tokens], hyp_tokens, weights=(0.25, 0.25, 0.25,
 0.25), smoothing_function=SmoothingFunction().method1)
```

```
    return blue4_score
def calculate_rouge(reference, hypothesis):
    rouge = Rouge()
    rouge_score = rouge.get_scores(hypothesis, reference, avg=True)
    return rouge_score['rouge-l']['f']


def compute_cider_score(reference, candidate):
    gts = {}
    res = {}
    for idx in range(len(reference)):
        gts[idx]=[reference[idx]]
        res[idx]=[candidate[idx]]

    # 初始化CIDEr评估器
    cider_scorer = Cider()

    # 计算CIDEr分数
    cider_score, _ = cider_scorer.compute_score(gts, res)
    return cider_score

openai.api_key ='' # 改为自己的
def extract_last_integer(string):
    # 使用正则表达式匹配最后一个整数
    match = re.search(r'\d+$', string)

    if match:
        last_integer = int(match.group())
        return last_integer
    else:
        # 如果字符串中没有整数，则返回None或其他默认值
        print('No integers in this string.')
        return 0

def get_score(output,ground_truth,task):

    system_messages = {
        'des':'''
        You will be given two text segments in the following format: [text1][text2]. T
hese two texts will be descriptions of a counterintuitive (humorous, creative, or magi
cal) video. For text2, your task is to provide a score based on the following criteri
a:
        1. Content: Score out of 20 points. If the content is nearly identical, award
 20 points. If the content differs slightly, deduct 5 points. If the content differs s
ignificantly, deduct 10 points. If the content differs greatly, deduct 15 points. If t
he content is completely different, deduct 20 points.
        2. Details: Score out of 50 points. Describe the video's details, including ch
aracters, scenes, actions, dialogues, etc. Deduct 5 points for each differing detail.
 Clearly identify and count the differing details to calculate the final score.
        3. Logic: Score out of 20 points. The description should be logically consiste
nt without any unreasonable situations. If the logic is nearly identical, award 20 poi
nts. If the logic is generally consistent but differs in details, award 15 points. If
 there are some differences in logic but still similar overall, award 10 points. If th
ere are significant differences in logic, award 5 points.
        4. Language Expression: Score out of 10 points. Evaluate the fluency and word
 usage of the text. If the language expression is at a consistent level, award 10 poin
ts. If there are minor differences in language expression, award 5 points. If there ar
e significant differences in language expression, award 0 points.
```

```
        Note: If the content differs significantly, multiply the total score by 0.5. I
f the content differs greatly, multiply the total score by 0.25.
        ''',
        'exp':'''
        You will be given two text segments in the following format: [text1][text2]. T
hese two texts will be explanations for a counterintuitive video (humorous, creative,
 or magical). For text2, your task is to provide a score based on the following criter
ia:
        1. Language Expression: Score out of 5 points. Evaluate the fluency and word u
sage of the text. If the language expression is at a consistent level, award 5 points.
If there are significant differences in language expression, award 0 points.
        2. Logic: Score out of 10 points. The explanation should be logically sound, p
referably with logical words and cause-effect relationships. If the logic is nearly id
entical, award 10 points. If the logic is generally consistent but differs in details,
award 5 points. If there are some differences in logic but still similar overall, awar
d 5 points. If there are significant differences in logic, award 0 points.
        3. Common Sense Errors: Score out of 10 points. The explanation should not con
tain any obvious common sense errors. Deduct 5 points for each occurrence of a common
 sense error.
        4. Understanding of Humor, Creativity, or Magic: Score out of 40 points. If th
e explanation focuses on the same key points as the reference answer, award 35 points
 or above. If the explanation provides reasons for the counterintuitive phenomenon but
differs from the reference answer, award between 15-35 points based on the difference.
If the explanation provides reasons for the counterintuitive phenomenon but differs gr
eatly from the reference answer, award between 0-15 points.
        5. Details: Score out of 35 points. While providing the explanation, include v
ideo details that contribute to the humor, creativity, or magical effect. Deduct 5 poi
nts for each additional or missing detail compared to the reference answer.
        6. If the explanation differs significantly from the reference answer and incl
udes descriptive details not mentioned in the reference answer, multiply the total sco
re by 0.5.
        7. The minimum score is 0, and the maximum score is 100.
        ''',
        'title':'''
        You will be given four text segments in the following format: [Description][Ex
planation][text1][text2]. The first two texts are descriptions of a video and its expl
anation, respectively. The third text is a reference title. Your task is to evaluate w
hether the fourth text is a good title. Note that the fourth text may not be a title b
ut a statement including the video. In that case, extract the actual title and evaluat
e it. Consider the following points while assigning a score:
        1. The title should mention the content of the video.
        2. A title with a certain level of humor or creativity is preferable.
        Provide a score ranging from 0 to 100, considering the above criteria.
        ''',
    }
    task_mapping = ['','','des','exp','title']
    gpt_input = '[' + ground_truth + '] [' + output +']'
    for _ in range(3):
        try:
            response = openai.ChatCompletion.create(
                model='gpt-3.5-turbo',
                messages=[{
                    'role': 'system',
                    'content': system_messages[task_mapping[int(task[-1])]],
                }, {
                    'role': 'user',
                    'content': gpt_input,
                }],
```

```
                temperature=0.7,
                max_tokens=1024,
                top_p=0.95,
                frequency_penalty=0,
                presence_penalty=0,
                stop=None,
            )
            ans = response['choices'][0]['message']['content']
            score = extract_last_integer(ans)
            return score
        except Exception as e:
            print('[ERROR]', e)
            ans =  '#ERROR#'
            time.sleep(1)
    return score

def eval_gpt(submission_json,ground_truth_json):
    scores = {
        'H2':0,
        'H3':0,
        'H4':0,
        'C2':0,
        'C3':0,
        'C4':0,
        'M2':0,
        'M3':0,
    }
    cnt = {
        'H2':0,
        'H3':0,
        'H4':0,
        'C2':0,
        'C3':0,
        'C4':0,
        'M2':0,
        'M3':0,
    }
    with open(submission_json) as f:
        submission = json.load(f)
    with open(ground_truth_json) as f:
        ground_truth = json.load(f)
    for i,j in zip(submission,ground_truth):
        this_score = get_score(i['output'],j['output'],i['task'])
        scores[i['task']] += this_score
        cnt[i['task']] += 1
    for i in scores:
        scores[i] /= cnt[i]
    return scores

def eval(submission_file, answer_file, total_score_path, run_time):
    print('Validating...')
    col = ['Task', 'H2', 'H3', 'H4', 'C2', 'C3', 'C4', 'M2', 'M3']
    with open(submission_file) as f:
        submission = json.load(f)
    with open(answer_file) as f:
        answer = json.load(f)

    chk_answer = []
```

```
    for data in answer:
        chk_answer.append(
            {'task': data['task'], 'output': data['output'], "instruction": data['inst
ruction'], "ID": data['ID']})

    diff = False
    for data in submission:
        if {'task': data['task'], 'output': data['output'], "instruction": data['instr
uction'], "ID": data['ID']} not in chk_answer:
            diff = True
            break

    assert diff == False, "Submission file is not valid"
    print('File is valid! Loading File...')

    submission = sorted(submission, key=lambda x: x['ID'])
    answer = sorted(answer, key=lambda x: x['ID'])

    can_path = 'bleurt/test_data/candidates'
    ref_path = 'bleurt/test_data/references'
    with open(can_path, 'w') as f_can:
        f_can.close()
    with open(ref_path, 'w') as f_ref:
        f_ref.close()

    bleurt_score_path = 'score.txt'
    eval_csv = pd.DataFrame(columns=['pre_output', 'gt', 'Task', 'bleurt_score'])
    for i in tqdm(range(len(submission))):
        pre_output = submission[i]['output']
        gt = answer[i]['output']

        with open(can_path, 'a') as f_can:
            f_can.write(pre_output + '\n')
        with open(ref_path, 'a') as f_ref:
            f_ref.write(gt + '\n')
        task = answer[i]['task']
        eval_csv = pd.concat([eval_csv, pd.DataFrame([[pre_output, gt, task, 0]], colu
mns=['pre_output', 'gt', 'Task', 'bleurt_score'])])

    f_can.close()
    f_ref.close()

    os.system('python -m bleurt.score_files   -candidate_file={}  -reference_file={}
-bleurt_checkpoint=BLEURT-20   -scores_file={}'
              .format(can_path, ref_path, bleurt_score_path))

    from time import sleep
    sleep(run_time)

    with open(bleurt_score_path) as f:
        eval_csv['bleurt_score'] = [i[:-1] for i in f.readlines()]

    rouge_score, bleu_score, bleurt_score, cider_score = {}, {}, {}, {}
    gpt_score = eval_gpt(submission_file, answer_file)
    bleurt_score['H2'], bleurt_score['H3'], bleurt_score['H4'], bleurt_score['C2'], bl
eurt_score['C3'], bleurt_score['C4'], bleurt_score['M2'], bleurt_score['M3'] = [], [],
[], [], [], [], [], []
    rouge_score['H2'], rouge_score['H3'], rouge_score['H4'], rouge_score['C2'], rouge_
```

```
score['C3'], rouge_score['C4'], rouge_score['M2'], rouge_score['M3'] = [], [], [], [],
[], [], [], []
    bleu_score['H2'], bleu_score['H3'], bleu_score['H4'], bleu_score['C2'], bleu_score
['C3'], bleu_score['C4'], bleu_score['M2'], bleu_score['M3'] = [], [], [], [], [], [],
[], []

    for index, row in eval_csv.iterrows():

        if row['pre_output'] == '':
            bleu_s = 0.
            rouge_s = 0.
            bleurt_s = 0.
        else:
            groudtruth_value = str(row['gt'])
            groudtruth_value.lower()
            result_value = str(row['pre_output'])
            result_value.lower()

            bleu_s = calculate_bleu4(groudtruth_value, result_value)
            rouge_s = calculate_rouge(groudtruth_value, result_value)
            bleurt_s = float(row['bleurt_score'])

        if row['Task'][0] == 'H':
            if row['Task'][-1] == '2':
                bleu_score['H2'].append(bleu_s)
                rouge_score['H2'].append(rouge_s)
                bleurt_score['H2'].append(bleurt_s)
            elif row['Task'][-1] == '3':
                bleu_score['H3'].append(bleu_s)
                rouge_score['H3'].append(rouge_s)
                bleurt_score['H3'].append(bleurt_s)
            elif row['Task'][-1] == '4':
                bleu_score['H4'].append(bleu_s)
                rouge_score['H4'].append(rouge_s)
                bleurt_score['H4'].append(bleurt_s)

        elif row['Task'][0] == 'C':
            if row['Task'][-1] == '2':
                bleu_score['C2'].append(bleu_s)
                rouge_score['C2'].append(rouge_s)
                bleurt_score['C2'].append(bleurt_s)
            elif row['Task'][-1] == '3':
                bleu_score['C3'].append(bleu_s)
                rouge_score['C3'].append(rouge_s)
                bleurt_score['C3'].append(bleurt_s)
            elif row['Task'][-1] == '4':
                bleu_score['C4'].append(bleu_s)
                rouge_score['C4'].append(rouge_s)
                bleurt_score['C4'].append(bleurt_s)

        elif row['Task'][0] == 'M':
            if row['Task'][-1] == '2':
                bleu_score['M2'].append(bleu_s)
                rouge_score['M2'].append(rouge_s)
                bleurt_score['M2'].append(bleurt_s)
            elif row['Task'][-1] == '3':
                bleu_score['M3'].append(bleu_s)
                rouge_score['M3'].append(rouge_s)
```

```
                bleurt_score['M3'].append(bleurt_s)

    bleu_score_ls, rouge_score_ls, bleurt_score_ls, cider_score_ls, gpt_score_ls, weig
hted_avg_ls = [], [], [], [], [], []
    for task in col[1:]:
        df = eval_csv[eval_csv['Task'] == task]
        cider_score_ls.append(compute_cider_score(list(df['gt']), list(df['pre_outpu
t'])) * 10)
        bleu_score_ls.append(sum(bleu_score[task])/len(bleu_score[task]) * 100)
        rouge_score_ls.append(sum(rouge_score[task])/len(rouge_score[task]) * 100)
        bleurt_score_ls.append(sum(bleurt_score[task])/len(bleurt_score[task]) * 100)
        gpt_score_ls.append(gpt_score[task])
        weighted_avg_ls.append(bleu_score_ls[-1] * 0.1 + rouge_score_ls[-1] * 0.1 + ci
der_score_ls[-1] * 0.1 + bleurt_score_ls[-1] * 0.2 + gpt_score_ls[-1] * 0.5)


    score_df = pd.DataFrame(columns=['Eval'] + col[1:])
    score_df.loc[len(score_df)] = ['BLEU-4'] + bleu_score_ls
    score_df.loc[len(score_df)] = ['ROUGE-L'] + rouge_score_ls
    score_df.loc[len(score_df)] = ['BLEURT'] + bleurt_score_ls
    score_df.loc[len(score_df)] = ['CIDEr'] + cider_score_ls
    score_df.loc[len(score_df)] = ['GPT-4'] + gpt_score_ls
    score_df.loc[len(score_df)] = ['Weighted Avg'] + weighted_avg_ls
    score_df.loc[len(score_df)] = ['Final Score'] + [list(score_df['H2'])[-1] * 0.15 +
list(score_df['H3'])[-1] * 0.15 + list(score_df['H4'])[-1] * 0.05 + list(score_df['C
2'])[-1] * 0.15 + list(score_df['C3'])[-1] * 0.15 + list(score_df['C4'])[-1] * 0.05 +
 list(score_df['M2'])[-1] * 0.15 + list(score_df['M3'])[-1] * 0.15] + [''] * 7
    score_df.to_csv(total_score_path, index=False)

    print('Finish! Score File was save in ' + total_score_path)

if __name__ == '__main__':
    eval("subbmision.json", "funqa_test.json", "total_score.csv", 60)
```

最终total_score.csv为：

```
Eval,H2,H3,H4,C2,C3,C4,M2,M3
BLEU-4,100.0,100.0,97.40151736066252,100.0,100.0,100.0,100.0,100.0
ROUGE-L,99.99999949999972,99.99999949999972,99.99999949999972,99.99999949999997,99.999
99949999997,99.99999949999997,99.99999949999982,99.99999949999983
BLEURT,98.0084100394573,97.7274618546168,97.9510729511579,94.2682933807373,95.94981312
75177,97.58472561836243,97.2402961748951,97.09363679091135
CIDEr,100.0,100.0,98.33333333333334,100.0,100.0,100.0,100.0,100.0
GPT-4,30,20,10,30,20,10,20,10
Weighted Avg,49.60168195789144,49.54549232092334,49.163699609631145,48.85365862614746,
49.189962575503536,49.51694507367248,49.448059184979,49.418727308182255
Final Score,49.342669530209236,,,,,,,
```