

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
(ФГАОУ ВО СПбПУ) Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

Отчет по практической работе
«MMBench: Является ли ваша мультимодальная модель универсальной?»

Выполнил:

Тампио Илья Сергеевич

гр. 5130904/10102

22.04.2024

Преподаватель:

Черноруцкий Игорь Георгиевич

22.04.2024

г. Санкт-Петербург
2024 г.

Оглавление

Введение.....	3
Постановка задачи.....	4
Алгоритмы.....	5
GPT-based evaluation.....	5
CircularEval.....	5
Реализация алгоритма.....	6
Общая архитектура.....	6
Входы и выходы.....	6
Код программы.....	7
Полученные результаты.....	11
Литература.....	12
Приложение.....	13

Введение

За основу работы выбрана статья:

Liu Y. et al. Mmbench: Is your multi-modal model an all-around player? //arXiv preprint arXiv:2307.06281. – 2023.

Полный текст статьи и ее перевод представлены в приложении в конце отчета.

Постановка задачи

Целью статьи MM Bench является разработать такой бенчмарк, который бы был пригоден для тщательной оценки способностей зрительно-языковых моделей.

Главные недостатки, с которыми справляется MM Bench по сравнению с другими бенчмарками: недостаточная классификация решаемых задач, небогатый набор тестовых данных, смещенность в предсказаниях модели, неточность формулировок ответов моделей.


Алгоритмы

GPT-based evaluation

– оценка на соответствие правильному варианту предсказаний оцениваемых моделей с помощью модели ChatGPT (OpenAI). Таким образом решается неточность формулировок ответов оцениваемых зрительно-языковых моделей. Использование такого алгоритма обусловлено высокой корреляцией между суждениями ChatGPT и человеческой оценкой (порядка 90%). Альтернативные решения в виде моделей с открытым исходным кодом не имеют такой же высокой корреляции. (порядка 50%)

CircularEval

– круговая оценка ответа модели на способность отвечать на вопрос. Суть заключается в переформулировании “правильности” ответа на вопрос. Для каждой задачи ответы смещаются по кругу, таким образом получая дополнительно N–1 вопрос, где N - кол-во вариантов ответов. Модель просят ответить на все вопросы, и если один из вопросов неверен, то вся задача считается выполненной неверно. Таким образом, авторы статьи решают проблему смещенности ответов вопросов модели, тем самым штрафуя модели, которые угадывают ответы на вопросы (вероятность угадать на вопрос и обмануть CircularEval для 3х вариантов ответа – в 8 раз, для 4х – в 16 раз сложнее). Ниже приведен пример.



The original VL problem:
Q: How many apples are there in the image?
A. 4; B. 3; C. 2; D. 1 GT: A

4 Passes in Circular Evaluation (choices with circular shift):

1. Q: How many apples are there in the image? Choices: A. 4; B. 3; C. 2; D. 1.	VLM prediction: A.	GT: A	✓
2. Q: How many apples are there in the image? Choices: A. 3; B. 2; C. 1; D. 4.	VLM prediction: D.	GT: D	✓
3. Q: How many apples are there in the image? Choices: A. 2; B. 1; C. 4; D. 3.	VLM prediction: B.	GT: C	✗
4. Q: How many apples are there in the image? Choices: A. 1; B. 4; C. 3; D. 2.	VLM prediction: B.	GT: B	✓

VLM failed at pass 3. Thus wrong.

Circular Evaluation

Поскольку модель не смогла ответить в третьем случае на тот же самый вопрос, задача считается решенной неверно.

Для реализации были выбраны оба эти алгоритма, но на уменьшенном датасете и проверить качество работы квантованной модели (из-за ограничений в вычислительной мощности).

Ожидается, что квантованная модель либо вовсе не сможет ответить на вопросы в случае угадывания, или будет выдавать неидеальные результаты.

Реализация алгоритма

Общая архитектура

Использовались два сервера llama.cpp для развертывания квантованных моделей LLM Solar-10.3b и VLM llava-1.6-7b.

Сначала получался ответ оцениваемой модели (llava-1.6-7b), после чего извлекалось соответствие полученного ответа правильному с помощью модели-оценщика (Solar).

Итоговым значением оценки была принята доля ответов признанных верными CircularEval, умноженная на 100.

Вопросы из реализованного бенчмарка представлены ниже.

```
8 [
9 questions = [
10 {
11     "image_link": "https://techbriefly.com/wp-content/uploads/2021/01/ddg2.jpg",
12     "question": "What animal is on the DuckDuckGo icon logo?",
13     "answers": ["a bird", "a dog", "a cat", "not an animal"],
14     "gt_answer": "a bird"
15 },
16 {
17     "image_link": "https://images6.fanpop.com/image/photos/35800000/Puppy-dogs-35894603-1920-1200.jpg",
18     "question": "What animal is on the image?",
19     "answers": ["a bird", "a dog", "a cat", "not an animal"],
20     "gt_answer": "a dog"
21 },
22 {
23     "image_link": "https://upload.wikimedia.org/wikipedia/commons/a/a5/Red_Kitten_01.jpg",
24     "question": "What animal is on the image?",
25     "answers": ["a bird", "a dog", "a cat", "not an animal"],
26     "gt_answer": "a cat"
27 },
28 {
29     "image_link": "https://i.insider.com/58fe49fb0ba0b8ea048b59e9?width=2400",
30     "question": "What animal is on the image?",
31     "answers": ["a bird", "a dog", "a cat", "not an animal"],
32     "gt_answer": "not an animal"
33 },
34 ]
```

Входы и выходы

На входе программы указываются вопросы, которые состоят из изображения, вопроса, вариантов ответа и правильного варианта ответа. Так же указываются URL серверов, на которых развернуты модель-оценщик и оцениваемая модель.

На выходе программа выдает количество вопросов, на которые правильно ответила модель по критерию CircularEval, общее количество вопросов и полученный моделью балл по бенчмарку.

Код программы

Программа реализована на языке Python 3.10, с использованием библиотек OpenAI для вызовов к моделям, а так же tqdm для отслеживания процесса выполнения программы.

```
import os
import typing as tp
from pprint import pprint

from openai import OpenAI
from tqdm import tqdm

questions = [
    {
        "image_link": "https://images6.fanpop.com/image/photos/35800000/Puppy-
dogs-35894603-1920-1200.jpg",
        "question": "What animal is on the image?",
        "answers": ["a bird", "a dog", "a cat", "not an animal"],
        "gt_answer": "a dog"
    },
    {
        "image_link":
"https://upload.wikimedia.org/wikipedia/commons/a/a5/Red_Kitten_01.jpg",
        "question": "What animal is on the image?",
        "answers": ["a bird", "a dog", "a cat", "not an animal"],
        "gt_answer": "a cat"
    },
    {
        "image_link": "https://i.insider.com/58fe49fb0ba0b8ea048b59e9?
width=2400",
        "question": "What animal is on the image?",
        "answers": ["a bird", "a dog", "a cat", "not an animal"],
        "gt_answer": "not an animal"
    },
]

def get_sample(q_text: str, answers: tp.List[str], gt: str, image_link:
tp.Optional[str] = "") -> tp.Tuple[str, str]:
    LETTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

    assert gt in answers, "GT is not in answers."

    gt_letter = LETTERS[answers.index(gt)]
    choices = [f'{LETTERS[i]}. {answer}' for i, answer in enumerate(answers)]
    choices_text = "; ".join(choices) + '.'
    sample = f'Q: {q_text}\nChoices: {choices_text}'

    return {
        "prompt": sample,
        "q_text": q_text,
        "options_text": choices_text,
        "gt_letter": gt_letter,
        "image_link": image_link
    }

def get_shifted_answers(answers) -> tp.List[tp.List[str]]:
    results = []
```

```

answers_len = len(answers)
for i in range(1, answers_len):
    shifted_answers = []
    for j in range(i, answers_len):
        shifted_answers.append(answers[j])
    for j in range(0, i):
        shifted_answers.append(answers[j])
    results.append(shifted_answers)
results = results[:-1]
results = [answers, ] + results
return results

def extract_model_choice(question: str, options: str, prediction: str,
evaluator_client):
    gpt_query_template = (
        "You are an AI assistant to help me matching an answer with several\n"
        "options of a multiple choice question. "\n"
        "You are provided with a question, several options, and an answer, "\n"
        "and you need to find which option is most similar to the answer. "\n"
        "If the meaning of all options are significantly different from the\n"
        "answer, output X. "\n"
        "Your should output a single uppercase character in A, B, C, D (if they\n"
        "are valid options), and X. \n"
        "Example 1: \n"
        "Question: What is the main object in image?\nOptions: A. teddy bear B.\n"
        "rabbit C. cat D. dog\n"
        "Answer: a cute teddy bear\nYour output: A\n"
        "Example 2: \n"
        "Question: What is the main object in image?\nOptions: A. teddy bear B.\n"
        "rabbit C. cat D. dog\n"
        "Answer: Spider\nYour output: X\n"
        "Example 3: \n"
        f"Question: {question}? \nOptions: {options} \nAnswer: {prediction} \nYour\n"
        "output: "
    )
    prompt = gpt_query_template.replace('{question}',
question).replace('{options}', options).replace('{prediction}', prediction)
    completion = evaluator_client.completions.create(
        model="i don't care",
        prompt=prompt,
        max_tokens=10, # because we await A, B, C, D or X.
        temperature=0.1,
        stop='\n',
    )
    answer = completion.content.strip()
    assert len(answer) >= 1
    return answer[0]

def get_model_response(question: str, options: str, image_url: str,
evaluator_client) -> str:
    template = (
        "You are an AI assistant to be evaluated by a benchmark. "\n"
        "You are provided with a question, several options, "\n"
        "and you need to choose an option which is best answers a questions\n"
        "based on image and question you get."
        "Your should output a single uppercase character in A, B, C, D (if they\n"
        "are valid options), and X. \n"
        f"Question: {question}? \nOptions: {options} \nYour answer: "
    )

```



```

    )
    prompt = template.replace("{question}", question).replace('{options}',
options)
    completion = evaluatee_client.chat.completions.create(
        model="i don't care (VL)",
        messages=[
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": prompt },
                    {
                        "type": "image_url",
                        "image_url": {
                            "url": image_url,
                        },
                    },
                ],
            },
        ],
        max_tokens=300,
    )
    answer = completion.choices[0].message.content
    return answer

def circular_eval_single(q: dict, evaluatee_client, evaluator_client, verbose:
bool = True) -> bool:
    """
        Performs circular eval on a question of format
        question = {
            "image_link":
"https://techbriefly.com/wp-content/uploads/2021/01/ddg2.jpg",
            "quesiton": "What animal is displayed on DuckDuckGo logo?",
            "answers": ["a bird", "a dog", "a cat", "not an animal"]
            "gt_answer": "a bird"
        }

        If necessary, it loads the image and then feeds it to the multi-modal
model.
    """
    assert 'question' in q
    assert 'answers' in q
    assert 'gt_answer' in q
    assert 'image_link' in q
    assert q['gt_answer'] in q['answers']
    assert len(q['answers']) > 1 and len(q['answers']) < 5

    image_link = q.get('image_link', "")
    q_text = q['question']
    answers = q['answers']
    gt_answer = q['gt_answer']

    shifted_answers_lists = get_shifted_answers(answers)

    passes = []
    for shifted_answers in shifted_answers_lists:
        pass_item = get_sample(q_text, shifted_answers, gt_answer, image_link)
        passes.append(pass_item)

    for i, p in enumerate(tqdm(passes, desc='passes', disable=not verbose)):

```

```

    q_text = p['q_text']
    options_text = p['options_text']
    gt_letter = p['gt_letter']
    image_link = p['image_link']

    if verbose:
        print("="*80)
        print(f"Q: {q_text}\nImage link:{image_link}\nOptions:
{options_text}\nGT: {gt_letter}")

    model_response = get_model_response(q_text, options_text, image_link,
evaluatee_client)
    if verbose:
        print(f"Model Raw: {model_response}")

    model_evaluated = extract_model_choice(q_text, options_text,
model_response, evaluator_client)

    if verbose:
        print(f"Extracted model choice: {model_evaluated}")

    assert model_evaluated in ["A", "B", "C", "D", "X"], model_evaluated

    if gt_letter.lower() != model_evaluated.lower():
        if verbose:
            print(f"Failed at pass {i}")

        return False

    return True

if __name__ == "__main__":
    OAI_EVALUATOR_API = os.getenv("OAI_EVALUATOR_API",
"http://localhost:6969/v1")
    OAI_EVALUATEE_API = os.getenv("OAI_EVALUATEE_API",
"http://localhost:4242/v1")

    evaluatee_client = OpenAI(base_url=OAI_EVALUATEE_API,
api_key="i_dont_care")
    evaluator_client = OpenAI(base_url=OAI_EVALUATOR_API,
api_key="i_dont_care")

    correct = 0
    for q in tqdm(questions, desc='Questions'):
        answer = circular_eval_single(q, evaluatee_client, evaluator_client)
        print(f'CircularEval verdict: {answer}')
        if answer:
            correct += 1
    print("="*80)
    print("SUMMARY")
    print(f"\tTotal: {len(questions)}")
    print(f"\tCorrect: {correct}")
    print(f"\tScore: {100 * correct/len(questions):.2f}")

```

--

Полученные результаты

В результате оценки квантованной модели `llava-v1.6-mistral-7b-q4_k_m.gguf` с помощью `CircularEval` модель смогла успешно ответить на 3/4 вопросов из тестового датасета, тем самым получая оценку в 75 баллов.

Стоит отметить, что 4 вопроса для полноценного бенчмаркинга, неоспоримо, недостаточно. Однако поскольку проект делался в учебных целях, количеством вопросов можно пренебречь, а метод считать работающим исправно.

Литература

1. Оригинальная статья MMBench: <https://arxiv.org/abs/2307.06281>
2. Репозиторий MMBench: <https://github.com/open-compass/mmbench>
3. Лидерборд MMBench: <https://mmbench.opencompass.org.cn/leaderboard>
4. Репозиторий VLMEvalKit: <https://github.com/open-compass/VLMEvalKit>
5. Репозиторий llama.cpp: <https://github.com/ggerganov/llama.cpp>
6. Лидерборд VLM Huggingfaces:
https://huggingface.co/spaces/opencompass/open_vlm_leaderboard
7. Репозиторий оцениваемой модели: <https://huggingface.co/cjpais/llava-1.6-mistral-7b-gguf>
8. Репозиторий модели-оценщика: <https://huggingface.co/TheBloke/SOLAR-10.7B-Instruct-v1.0-uncensored-GGUF/tree/main>

Приложение

Далее идёт полный текст статьи, а затем – её перевод.