DL Kiso VizWiz_VQA Solution

工夫点

- 学習データの選択
- モデルの選択
- 結果のアンサンブル
- gdownを利用し、Google Driveからインスタンス(ランタイム)直下にtrain.zipやvalid.zipをダウンロードし、unzipで展開し利用した。

学習データの作成・選択

まず回答のデータセットを作成。trainデータセットの中で3つ以上一致して且つ確信度がyesのものが多いものを回答とした。→決まらない場合は2つ以上一致して且つ確信度がyesのものが多いもの→それでも決まらないものは学習データに加えない

作成したjson

```
[
    {
        "image": "train_00000.jpg",
        "question": "What is this?",
        "answer": "beef chuck steak"
    },
    {
        "image": "train_00001.jpg",
        "question": "maybe it's because you're pushing it down instead",
        "answer": "unanswerable"
    },
    ...
```

llavaの学習形式のjsonなど必要な形式に適宜変換した。(https://github.com/haotian-liu/LLaVA/blob/main/docs/Finetune_Custom_Data.md　を参考に)

```
[
  {
    "id": "train_00000",
    "image": "train_00000.jpg",
    "conversations": [
      {
        "from": "human",
        "value": "<image>\nWhat is this?"
      },
      {
        "from": "gpt",
        "value": "beef chuck steak"
      }
    ]
  },
  {
    "id": "train_00001",
    "image": "train_00001.jpg",
```

```
    "conversations": [
      {
        "from": "human",
        "value": "<image>\nmaybe it's because you're pushing it down instead"
      },
      {
        "from": "gpt",
        "value": "unanswerable"
      }
    ]
  },
  ...
```

モデルの選択

最初に試した方法 ： `GLM-4V-9B` （Github / HuggingFace）を使用したLoRAのファインチューニングを実施。

LLMのパラメーターが9B、Vision encoderのパラメーターが4Bで計13Bの非常に大きいモデル

H100(VRAM 80GB)で３時間(2epoch)ほど、今回のデータセットに合わせて編集した。

`finetune_vision.py`, `lora.yaml`を使いLoRAでファインチューニング。

`nohup python finetune_vision.py . THUDM/glm-4v-9b configs/lora.yaml > finetune.log 2>&1 &`

`trainable params: 6,397,952 || all params: 13,912,718,848 || trainable%: 0.0460`

出力が崩壊（アラビア語やロシア語が混ざってしまう）してしまう現象に遭遇した。

理由として考えられるもの

- lora_rank=8で小さすぎたため
- 学習不足のため
- 推論時のパラメーター(top_pなど)をあまり試せなかったため

結果 0.48

そもそも推論だけで２時間ほどかかり、お金がもったいないため断念。よって、方針を変更し、もう少し小さいモデルで行い、実験を繰り返すことにした。

`Phi-3-vision`のファインチューニング(LoRA)　HuggingFace　ファインチューニングコードは https://github.com/GaiZhenbiao/Phi3V-Finetuning[1]より引用、改変。

A100(40GB)を使用。

DeepSpeed ZeRO-2を使用。lr=1e-5, max_seq_length=100に変更。6epoch実施。時間は８時間ほど。

以下実験結果をもとに推論のパラメーターを色々変えて試したもの。

黄色のものをメインにアンサンブルし、一番良いものを提出した。

n		max_new_token	top_p	temperature	do_sample	result
2_2	2epoch学習	10(\n 記号より前のみ抽出)			False	0.66949
2_3	2epoch学習	10(\n 記号より前のみ抽出)	0.9	0.7	True	0.62747
2_6	2epoch学習	10(\n 記号より前のみ抽出)	0.3	0.3	True	0.66951
2_6	2epoch学習	10(\n 記号より前のみ抽出)	0.3	0.3	True	0.664
2_7	2epoch学習	10(\n 記号より前のみ抽出)	0.2	0.2	True	0.66864
3	3epoch学習	4	0.9	0.7	True	0.59746

3_1	3epoch学習	10(\n 記号より前のみ抽出)			False	0.65313
3_2	3epoch学習	10(\n 記号より前のみ抽出)	0.3	0.3	True	0.65259
4_1	4epoch学習	10(\n 記号より前のみ抽出)			False	0.64152
5	5epoch学習	10(\n 記号より前のみ抽出)	0.9	0.7	True	0.61463
5_1	5epoch学習	10(\n 記号より前のみ抽出)			False	0.6522
5_2	5epoch学習	10(\n 記号より前のみ抽出)	0.3	0.3	True	0.65258
6	6epoch学習	4	0.9	0.7	True	0.59775
6_1	6epoch学習	10(\n 記号より前のみ抽出)			False	0.65118
6_2	6epoch学習	10(\n 記号より前のみ抽出)	0.3	0.3	True	0.64959
ensemble						0.67

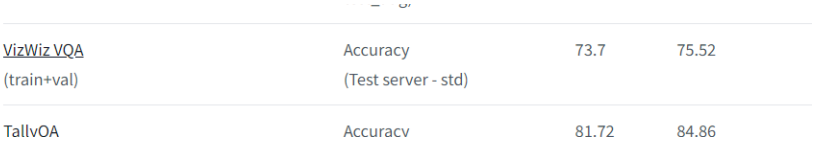
max_new_tokenを大きくするとなどにすると、`no \nunanswerable \n` こういう回答ばかりになっていた。それで、まず`unanswerable` が出る4にしたが、`chicken noo` などのtoken数が足りないことによる回答があったため、max_new_token:10にして、\n以下を切れば望ましい答えが出力させた。

また、2_2(do_sample=False)と4の`unanswerable` の比率を比較すると、2_2が51%、4が39%

一般的に一貫性を高め、創造性を下げようとすると、データの多様性がなくなり、同じ回答ばかり出力される。そのバランスが難しかった。結局アンサンブルしてもっともよい精度の提出した。

反省点

調査不足で、Phi-3 visionと同時期に発表されたGoogleのPaliGemmaでファインチューニングすれば73~75点ほどが出るらしい。Huggingfaceにモデルあり。提出前日まで気付かなかった。



PaligemmaでFinetuneすればもっと精度があげられると思うが今回は実施しなかった。