자연어 기반 정성 평가 캡셔닝 생성 및 화질 정량 평가

김소원(소프트웨어학부), 나상현(소프트웨어학부), 나영은(역사학과), 최형용(국어국문학과)

2024 CUAI 중앙대학교 인공지능 학회 하계 컨퍼런스
Proceeding of 2024 Chung-Ang University Artificial Intelligence Summer Conference

Abstract

사진 입력에 대해서 정량적 인지 화질 점수와 자연어 기반 정성 평가 캡셔닝을 생성하는 AI 모델을 개발을 목표로 한다. “CPTR: Full Transformer Network for Image Captioning” 논문의 모델에 따라 코드 구조를 완성하고 실행 중 논문과 코드 간의 불일치하는 부분을 확인한다. 이를 활용하여 유의미한 결과를 도출한다.

Introduction

사용자의 상황에 최적화된 서비스를 제공하는 스마트폰 카메라의 AI 영상 처리 기능 개발 연구가 계속되고 있다. 선명도, 노이즈 정도, 색감, 선호도 등 여러 인지적 화질 요소를 종합적으로 고려하여 정량 평가 점수를 예측하고 단일 점수에서 누락될 수 있는 의미들을 자연어로 영상의 화질을 설명하고자 한다.

카메라로 촬영된 영상의 화질에 대한 정량 평가 점수를 예측하고, 그 평가 결과를 자연어로 상세하게 표현하는 알고리즘을 개발하는 것을 목표로 한다.

Methods

1) 활용 데이터 : ‘2023 Samsung AI Challenge: Image Quality Assessment” 데이콘 데이터

train.csv | test.csv | sample_submission.csv

Views | Grid view | Hide fields | Filter | Group | Sort

	img_name	img_path	mos	comments
1	41wy7upxzl	./train/41wy7upxzl.jpg	5.56923077	the pink and blue really c...
2	ygujjq6xxt	./train/ygujjq6xxt.jpg	6.10317460	love rhubarb! great colors!
3	wk321130q0	./train/wk321130q0.jpg	5.54198473	i enjoy the textures and g...
4	w50dp2zjpg	./train/w50dp2zjpg.jpg	6.23484848	i like all the different colo...
5	l7rqfxeuh0	./train/l7rqfxeuh0.jpg	5.19047619	i love these critters, just ...
6	iapcid06sr	./train/iapcid06sr.jpg	5.93846154	excellent use of light. gre...
7	twvec6pi41	./train/twvec6pi41.jpg	7.38931298	double trouble! what grea...
8	h0wh5in2rd	./train/h0wh5in2rd.jpg	6.12307692	is this really meters from ...
9	j70kuiwf5y	./train/j70kuiwf5y.jpg	6.86614173	i generally dont like overp...
10	9qls7ros2m	./train/9qls7ros2m.jpg	5.75396825	awesome tones. great mo...

10 records | Sum 60.71067621

Airtable

train.csv | test.csv | sample_submission.csv

Views | Grid view | Hide fields | Filter

	img_name	img_path
1	j00zs3u6dr	./test/j00zs3u6dr.jpg
2	ytv70so3zb	./test/ytv70so3zb.jpg
3	ia9890oozp	./test/ia9890oozp.jpg
4	xsj81ypx4a	./test/xsj81ypx4a.jpg
5	f23994ghlh	./test/f23994ghlh.jpg
6	olzx3kqzij	./test/olzx3kqzij.jpg
7	pzhsjj4uxo	./test/pzhsjj4uxo.jpg
8	2mb38w9n83	./test/2mb38w9n83.jpg
9	cz4tby7pfx	./test/cz4tby7pfx.jpg
10	8099bltbox	./test/8099bltbox.jpg

10 records

2) 이론적 배경

새로운 sequence to. sequence 예측 관점에서의 이미지 캡셔닝을 다룬다. CaPtion TransformeR (CPTR)는 sequentialized 원본 이미지를 Transformer의 입력으로 사용하는 모델로, 모든 인코더 레이어에서 처음부터 global context를 모델링할 수 있으며, convolution-free 모델이라는 점을 강조한다.

피처를 CNN에서 추출하는 기존의 캡셔닝 모델과 다르게, CPTR은 원본 이미지를 sequentialize 해 입력 데이터로 바로 사용한다. 인코더는 self-attention 매커니즘을 통해 패치들 간의 장기 의존성을 처음부터 활용할 수 있게 하고, 이미지를 고정된 크기 (16×16)로 작은 패치로 나눈 후, 각 패치를 flatten하고 이를 1D 패치 시퀀스로 변형하여 활용한다.

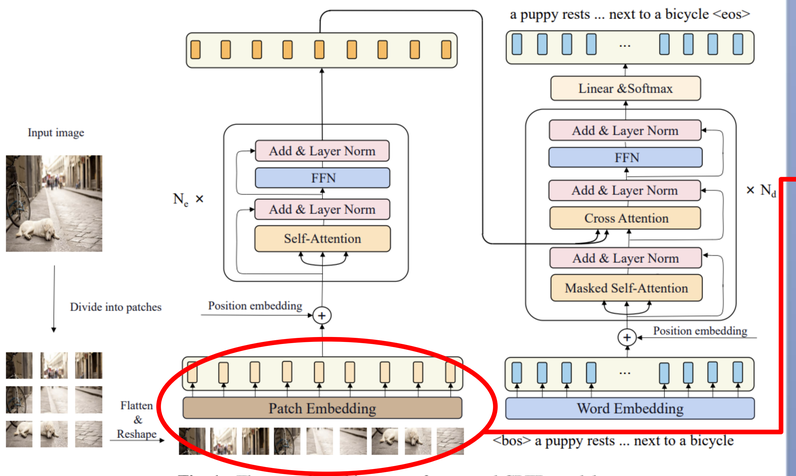

Fig. 1. The overall architecture of proposed CPTR model.

3) 코드 구조: Processing, Learning, Describe 3단계

a. Processing

▽어휘집 추출

```
with open(file=path2captions, mode='r') as fp:
    img2captions = json.load(fp)

captions = list(img2captions.values())
captions = list(it.chain(*captions))

tokenizer = build_tokenizer(tok_name='spacy', lang='en_core_web_sm')
vocabulary = make_vocab(captions, tokenizer, SPECIALS2IDX)
logger.success('vocabulary was built')

serialize(path2vocabulary, vocabulary)
```

▽토큰화 처리

```
bos = th.tensor([SPECIALS2IDX['<bos>']])
eos = th.tensor([SPECIALS2IDX['<eos>']])

zip_img2tokenids = []
logger.debug('caption tokenization')
for key, val in track(img2captions.items(), 'build map_img2tokenids'):
    for cap in val:
        tok = tokenizer(cap.strip().lower())
        ids = th.tensor(vocabulary(tok))
        seq = th.cat([bos, ids, eos]).numpy()  # more effective for storage
        zip_img2tokenids.append((key, seq))

serialize(path2tokenids, zip_img2tokenids)
```

▽이미지 피처 추출

```
logger.debug('features extraction loading ...')
vectorizer = load_vectorizer(path2vectorizer)
vectorizer.eval()
vectorizer.to(device)

dataset = Dataset4FeatureExtraction(path2images, f'*.{extension}')

logger.debug('extraction will start')
accumulator = []
for sections in track(dataset, 'features extraction'):
    embedding = extract_features(vectorizer, sections[None, ...].to(device)).squeeze(0) # (1868, 7, 7)
    embedding = th.flatten(embedding, start_dim=1).T.cpu().numpy()  # 49, 2048
    accumulator.append(embedding)

image_names = dataset.image_names
accumulator = np.stack(accumulator)  # stack over batch axis ==> (nb_images, 49, 512)
logger.debug(f'accumulated features shape : {accumulator.shape}')
assert len(image_names) == len(accumulator)
map_imgfeatures = dict(zip(image_names, accumulator))

serialize(path2features, map_imgfeatures)
```

b. Learning(Train)

▽어휘집, 데이터 로드

```
logger.debug('load vocabulary')
vocabulary = deserialize(path2vocabulary)
nb_tokens = len(vocabulary)

logger.debug('build dataset')
dataset = Dataset4Training(path2tokenids, path2features)
logger.debug(f'size of the dataset : {len(dataset):05d}')
dataloader = DataLoader(dataset, batch_size=bt_size, shuffle=False, collate_fn=custom_fn)
# dataloader = DataLoader(dataset, batch_size=bt_size, shuffle=True, collate_fn=custom_fn)
nb_data = len(dataset)
```

```
logger.debug('define network')
if path.isfile(path2checkpoint):
    net = th.load(path2checkpoint)
else:
    net = CaptionTransformer(
        in_dim=2048,
        hd_dim=256,
        ff_dim=512,
        nb_heads=8,
        num_encoders=5,
        num_decoders=5,
        pre_norm=False,
        seq_length=128,
        nb_tokens=nb_tokens,
        padding_idx=SPECIALS2IDX['<pad>']
    )
net.to(device)
net.train()
```

< 트랜스포머 모델 생성

▽최적화 함수, 손실 함수 설정

```
optimizer = th.optim.Adam(net.parameters(), lr=1e-5, betas=(0.9, 0.99), eps=1e-9)
criterion = nn.CrossEntropyLoss(ignore_index=SPECIALS2IDX['<pad>'])
logger.debug('training will begin ...I')
sleep(1)
```

(이어서) b. Learning(Train)

```
memory = net.encode(src=src.to(device))
output = net.decode(
    tgt=tgt_input.to(device),
    memory=memory,
    tgt_mask=tgt_mask,
    tgt_key_padding_mask=tgt_key_padding_mask
)
logits = [net.generator(out) for out in output ]
```

< encode/decode

```
# hash 값 init 값 처리
for i, logit in enumerate(logits):
    if th.isnan(logit).any() or th.isinf(logit).any():
        logger.error(f'NaN or Inf detected in logits at index {i}')
        logit = th.nan_to_num(logit, nan=1e-9, posinf=1e9, neginf=-1e9)
        logits[i] = logit

logits = [ th.flatten(prb, start_dim=0, end_dim=1) for prb in logits ]
tgt_output = th.flatten(tgt_output)
```

< NaN, inf 값 처리

```
optimizer.zero_grad()
errors = [ criterion(prb, tgt_output.to(device)) for prb in logits ]
error = sum(errors)
try:
    error.backward()
except Exception as err:
    pdb.set_trace()

# gradient clipping 하여 exploding gradient 방지
th.nn.utils.clip_grad_norm_(net.parameters(), max_norm=1.0)

optimizer.step()
```

epoch

<역전파

c. describe(inference)

```
vocab = deserialize(path2vocabulary)
logger.debug(f'vocab was loaded | len => {len(vocab)}')
```

< 어휘집 로드

```
logger.debug(f'load features extractor')

vectorizer = load_vectorizer(path2vectorizer)
vectorizer.eval()
vectorizer.to(device)

logger.debug('load ranker clip ViT model')
ranker, processor = load_ranker(path2ranker, device)

logger.debug('features extraction by resnet152')

img_list = pull_images(os.path.dirname(path2image))
img_list = [os.path.basename(img) for img in img_list]
img_dir = os.path.dirname(path2image)
```

<이미지 피처 추출 (resnet152)

<CLIP ViT 모델 로드

```
cv_image = read_image(img_dir + '/' + img_name)
th_image = cv2th(cv_image)
th_image = prepare_image(th_image)

embedding = extract_features(vectorizer, th_image[None, ...].to(device)).squeeze(0)
output_batch = th.flatten(embedding, start_dim=1).T  # 49, 2048

response = beam_search(
    model=net,
    source=output_batch[None, ...],
    BOS=SPECIALS2IDX['<bos>'],
    EOS=SPECIALS2IDX['<eos>'],
    max_len=64,
    beam_width=beam_width,
    device=device,
    alpha=0.7
)

# logger.debug(f'nb generated : {len(response)}')
sentences = []
for sequence, _ in response:
    caption = vocab.lookup_tokens(sequence[1:-1])  # ignore <bos> and <eos>
    joined_caption = ' '.join(caption)
    sentences.append(joined_caption)
```

< beam search

```
# logger.debug('ranking will begin...')
pil_image = cv2pil(cv_image)
ranked_scores = rank_solutions(pil_image, sentences, ranker, processor, device)
ranked_response = list(zip(sentences, ranked_scores))
ranked_response = sorted(ranked_response, key=op.itemgetter(1), reverse=True)

for caption, score in ranked_response:
    score = int(score * 100)
    # logger.debug(f'caption : {caption} | score : {score:03d}')
```

< CLIP ViT 모델 활용 랭킹

```
best_caption, best_score = ranked_response[-1]
logger.debug(f'Filename : {img_name} | best caption: {best_caption} | Score: {best_score}')

# csv 파일에 저장
csv_file_path = os.path.join('./results/results.csv')
with open(csv_file_path, mode='a', newline='') as file:
    writer = csv.writer(file)
    writer.writerow([img_name, best_caption])
```

< 최고 득점 캡션을 저장

4) 논문과 코드 간의 불일치

Inspired by the above works, we consider solving the image captioning task from a new sequence-to-sequence perspective and propose CaPtion TransformeR (CPTR), a full Transformer network to replace the CNN in the encoder part with Transformer encoder which is totally convolution-free. Compared to the conventional captiong models taking as input the feature extracted by CNN or object detector, we directly sequentialize raw images as input. Specifically, we divide an image into small patches of fixed size (e.g. 16×16), flatten each patch and reshape them into a 1D patch sequence. The patch sequence passes through a patch embedding layer and a learnable positional embedding layer before being fed into the Transformer encoder.

논문에서는 fully convolution free라는 점을 강조하나, 실제 코드에서는 CNN 기반 모델 (ResNet152) 을 사용해 이미지 피처를 추출하고 있다.

Results

5) 결과

#	팀	팀 멤버	화질 점수	캡셔닝	제출수	등록일
26	na_sanghyun	na	0.5594	0.66175	1	4일 전

데이콘에 베이스라인 모델의 결과와, 본문에서 소개한 모델의 결과를 제출하여 비교하려 하였으나 베이스라인 모델의 캡션 생성 오류로 진행하지 못하였다. 다만 본문 모델 캡션의 결과는 0.66175점으로, 캡셔닝 점수 기준 23등의 성과를 거두었다.

6) 연구 한계 및 제언

베이스라인 모델로 추론한 mos와 캡션을 데이콘에 제출하지 못하여 베이스라인 모델에 비한 본문 모델의 향상 정도를 수치적으로 나타내기 어려웠다. CNN으로 이미지 피처를 추출한 후 이를 트랜스포머 인코더에 입력으로 주는 방식을 적용하여 성능이 기대만큼 향상되지 않았으리라 생각한다. 향후 CLIP 등의 방대한 양의 데이터로 pre-trained ViT 인코더를 사용하여 성능 향상을 기대할 수 있을 것이다.

Conclusion

데이콘에서 주어진 이미지 데이터를 기반으로 MOS를 예측하고 캡션을 생성하였다. 이를 통해 트랜스포머 구조 기반 모델을 사용한 이미지 캡셔닝을 수행할 수 있었다. 자연어 처리에서 효과적인 성능을 보이는 트랜스포머 모델에 완전한 convolution-free 구조를 적용하기 위해 추후 pre-trained 인코더 등의 모델을 활용하는 연구를 지속하면 이미지 캡셔닝을 효과적으로 수행하는 모델을 개발할 수 있을 것으로 확인된다.

Reference

1) Wei Liu, et al. “CPTR: Full Transformer Network for Image Captioning”, arXiv:2101.10804, 2021.
2) Tsung-Yi Lin, et al. “Microsoft COCO: Common Objects in Context”, arXiv:1405.0312, 2015
3) Ashish Vaswani, et al. “Attention Is All You Need.”, Google AI Language, 2017