Project 3

20307130030 Shen Jianzhi

2023.6.14

1 Task

The task is to assign a caption to the image with novel objects. That means the model is not trained on caption-image pairs related to the objects, i.e. it has no prior idea that a zebra on the image is called "zebra", but in the shared vision and text embedding space it can find the corresponding relationship, and thus deduce the word.

2 Data

The data used is MSCOCO 2014(1), where an image has five corresponding captions. An example for the image-caption pairs is shown in Figure 1. Based on the original split (81K training, 43K test and 43K validation), this task modifies the split by removing the eight novel objects (bottle, bus, couch, microwave, pizza, racket, suitcase and zebra) from the training split, and out-of-domain samples of each novel object are selected from the original validation split, forming eight object-specific test sets and validation sets. Besides the MSCOCO dataset, we also utilize the pre-trained word embedding model GloVe(2).

3 Related Work

At the time novel object caption task was introduced, LRCN(3), which used a CNN visual encoder and an autoregressive textual decoder was the state-of-the-art image captioning model. And the encoder-decoder framework became a paradigm, even though nowadays the encoder has changed from VGGs to ResNet with object detector Faster R-CNN or ViTs, and the decoder has evolved from single-direction LSTM to ELMo or GPTs.

However, with the visual encoder and textual decoder separately learning their own knowledge in the image-caption pairs, the model is not able to properly caption an object which has not appeared in the paired data. Novel object captioning requires either the visual encoder to tell the nuance by word similarity or the textual decoder to query for image label. DCC(4) first realizes novel object captioning by transferring weights to novel objects from their closest neighbor in the dense word embedding space. NOC(5) further integrated the idea into an end-to-end model and prevent the visual encoder from "forgetting" rare objects by joint training with auxiliary loss. LSTM-C(6) took another route, which solve the out-of-vocabulary

Figure 1: Image Example

Caption:
A photo of a kitchen that has a stainless steel stove.
A neatly organized kitchen with blue counter tops.
a black oven blue counters and a brown table and chairs
A kitchen with a stove, blue counters and a large dining room table with chairs
A kitchen and dining room are empty, but a light is on.

problem by selectively copying the result of object detection. DNOC(7) and NBT(8) further polished this copying mechanism with attention.

4 Implementation

We implemented the frameworks used by DCC(4) and NOC(5).

4.1 Lexicon Learning

The lexicon, or the image encoding part, is implemented as ResNet50 with a multi-label linear head using BCE loss. DCC used nouns, adjectives, and verbs with the highest frequency. In contrast, We extract 500 terms with the highest maximal TF-IDF across all pictures in MSCOCO dataset as shown in Figure 2, so that some common words that is not image-specific can be dropped. Then an image's labels are assigned as lexicons appearing in the five annotations of the image. Unlike DCC, NOC uses all the vocabulary as multi-label. It needs specifying that the eight novel objects have to appear in the selected lexicons and learned by the lexicon part. Otherwise the f1-score is always 0.

4.2 Text Learning

The words with frequency lower than 8 are treated as "<UNK>". We tried both LSTM and GRU as auto-regressive text decoders on DCC. Their performances are close, since the texts are short and highly homogeneous as well. GRU takes less time to learn, so our NOC model is based on GRU. Since PyTorch DataLoader can only batch tensors with the same length, and the captions are length-variable, we tried two method to train the RNN decode, as shown in 3. And that do not affect the result too much either, when

```
tf = {word: 0 for word in vocab_map if word not in ('<BOS>', '<EOS>', '<UNK>')}
df = {word: 0 for word in tf}
df_temp = {word: 0 for word in tf}
tf_temp = {word: 0 for word in tf}
last_image = None
for ann_index in train_coco.anns:
    text = str(train_coco.anns[ann_index]['caption']).lower()
    token = nltk.tokenize.word_tokenize(text)
    image_id = train_coco.anns[ann_index]['image_id']
    if not image_id == last_image:
        df = {word: df[word] + df_temp[word] for word in tf}
        tf = {word: max(tf_temp[word], tf[word]) for word in tf}
        df_temp = {word: 0 for word in tf}
        tf_temp = {word: 0 for word in tf}
        last_image = image_id
    for word in token:
        if word in tf:
            tf_temp[word] += 1/ann_sum[image_id]
            df_temp[word] = 1

rank = {word: tf[word] * np.log(len(train_coco.anns) / 5 / df[word]) for word in tf}
lexicon_list = nlargest(500, rank, key=lambda x: rank[x])
lexicon_map = {lexicon: lexicon_list.index(lexicon) for lexicon in lexicon_list}
```

Figure 2: TF-IDF lexicon selection

the batch size is set to 64. The different between DCC and NOC's text embedding is that NOC uses the transpose of the embedding matrix as the output layer's weight, so that the text model is encouraged to use semantically similar words when the visual confidence is strong, while DCC just uses a plain fully connected layer.

```
def rnn_collate(data):
    data.sort(key=lambda x: len(x[0]), reverse=True)
    lengths = [len(seq[0]) for seq in data]
    data = torch.nn.utils.rnn.pad_sequence(data, batch_first=True, padding_value=-100)
    return torch.LongTensor(data), lengths
```

(a) Pad and pack

```
def random_length(self):  # Randomly choose a caption length to unify the length within batch
    length = random.sample(self.caption_lengths, 1)
    selected = np.where([self.caption_lengths[i] == length for i in range(len(self.caption_lengths))])[0]
    return selected
```

(b) Unify length in a batch

Figure 3: Two methods loading sequential data

4.3 Paired Learning

The eight novel objects are removed from the paired learning phase. At this phase, DCC freezes the image embedding part and the RNN weights, and only trains the language output layer and fine-tune the text output layer, i.e it maps the RNN hidden layer output and image multi-label classifier logit to the same vocab-size-dimension space. The predicted next-word distribution is derived as sum of the language and image outputs. After training, it transits the weight from the semantically close (based on GloVe) lexicons

to the unseen lexicons, as Figure 4 shows. It is worth noticing that the plural forms are also considered, which further improves the model's performance.

```
def transit(model):
    model.cpu()
    for o in novel_objects_with_plural:
        o_id = vocab_map[o]
        o_emb = glove_emb[o_id]
        min_dist = float("inf")
        closest_id = None
        for i, emb in enumerate(glove_emb):
            if vocab_list[i] not in lexicon_map or i == o_id:
                continue
            dist = np.linalg.linalg.norm(o_emb-emb)
            if dist < min_dist:
                min_dist = dist
                closest_id = i
        closest_lexicon_index = lexicon_map[vocab_list[closest_id]]
        with torch.no_grad():
            image_weight, image_bias = model.image_trans.parameters()
            text_weight, text_bias = model.text_trans.parameters()
            for param in [image_weight, image_bias, text_weight, text_bias]:
                param[o_id] = param[closest_id]
            image_weight[o_id, lexicon_map[o]] = image_weight[closest_id, closest_lexicon_index]
            image_weight[o_id, closest_lexicon_index] = 0
            image_weight[closest_id, lexicon_map[o]] = 0
    model.to(device)
```

Figure 4: Implementation of transition

On the other hand, NOC simultaneously trains the three parts of the model. At every step the DataLoader provides an image-lexicon pair, a caption, and an image-caption pair, and the loss is the sum of the lexicon, text and paired loss.

5 Experiment

Hyper-parameter	Lexicon	Text	Paired
Optimizer	Adam + One Cycle LR	Adam 0.01	Adam 10^{-4}
Num of Epochs	10	20	10
Batch size	64	256	64

Table 1: Hyper parameters

In DCC Text model, the hidden layer is set to 512.

5.1 Quantitative

As can be seen in the result, NOC does significantly better than DCC on identifying bus and suitcase. The reason may be that the semantic relationship around bus and suitcase may be relatively irrelevant to the vision similarity, so direct transferring these lexicons may fail, but NOC smoothes this by an attention-like mechanism, so the combination of neighbor weights is continuous, rather than one hot.

Metric	Model	bottle	bus	couch	microwave	pizza	racket	suitcase	zebra	Avg.
METEOR	DCC-LSTM	17.3	20.9	23.0	21.2	22.2	20.4	18.0	21.8	20.3
	DCC-GRU	17.2	20.7	22.3	21.8	21.7	21.1	18.2	22.2	20.4
	NOC	20.1	20.4	22.0	20.9	20.7	22.3	17.9	21.3	20.7
CIDEr	DCC-LSTM	46.6	29	48.9	36.5	31.5	23.0	42.9	37.1	37.0
	DCC-GRU	46.2	29.3	48.2	37.6	33.3	21.7	43.2	37.4	37.2
	NOC	48.2	32.3	49.0	36.6	32.9	20.7	43.8	42.5	38.7
SPICE	DCC-LSTM	14.7	14.9	17.7	15.0	16.5	13.2	13.6	16.8	15.4
	DCC-GRU	14.5	14.2	17.9	14.7	16.9	14.6	12.9	16.0	15.3
	NOC	15.2	14.0	18.7	15.9	16.9	14.8	12.7	17.5	15.7
F1	DCC-LSTM	12.3	20.2	17.6	22.9	52.5	43.4	12.7	82.6	32.3
	DCC-GRU	11.9	20.4	18.2	22.8	54.3	42.6	13.0	82.7	32.5
	NOC	15.0	66.0	22.7	22.1	64.7	53.3	37.5	86.0	46.2

Table 2: Result on the eight novel objects

Some metrics and F1 are lower than stated in the papers. First, some implementation details were unclear, like in DCC how to choose the lexicons, and choose what semantic neighbors to transfer the weight from. Second, the training may be insufficient, since we train our lexicon model without a pretrained start, and the training stops after validation loss rises, and may be trapped in local minimum. Third, the automatic CIDEr metric does not work properly, since the more sentence that are tested, the higher the CIDEr is.

But some F1's of DCC, like bottle and zebra, are even higher than what is shown in the paper. Which further shows the model is sensitive to the selection of the semantic neighbors and lexicons.

The results of LSTM and GRU are close.

5.2 Qualitative

As can be seen in Figure 5, although both models correctly identified the two novel objects, DCC had some trouble giving a grammatically correct answer. The reason may be DCC links bottle to wine, since the two always appear together in a context, and the transferred weight enforced the model to output bottle when wine is detected. The third bottle may be produced by greedy search, and beam search may help prevent this phenomenon.

Figure 6 shows an example when both models fail to identify the bus. It may be because the lexicon part fail to detect it.

Figure 7 gives an explanation why DCC has relatively low F1 scores in suitcase. The five label captions contain no suitcase but luggage instead. However, DCC may link luggage to suitcase so that it tends to output suitcase. That is another evidence that DCC is not so robust compared to NOC.

References

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

Figure 5: Example with pizza and bottle

DCC:
A pizza with a bottle of bottle of bottle.
NOC:
A pizza with a bottle of wine and a glass of wine.
Label:
A pizza is seen with a glass of wine on a table.

Figure 6: Example with bus

DCC:
A group of people standing near a white house.
NOC:
A group of people playing instrument in front of a government house.
Label:
A group of guys performing on top of a bus.

Figure 7: Example with suitcase

DCC:
A man sitting in a chair with a suitcase.
NOC:
A man sitting next to a Christmas tree with a luggage.
Label:
A man sits in a chair while zipping luggage.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[3] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," 2016.

[4] L. A. Hendricks, S. Venugopalan, M. Rohrbach, R. Mooney, K. Saenko, and T. Darrell, "Deep compositional captioning: Describing novel object categories without paired training data," 2016.

[5] S. Venugopalan, L. A. Hendricks, M. Rohrbach, R. Mooney, T. Darrell, and K. Saenko, "Captioning images with diverse objects," 2017.

[6] T. Yao, Y. Pan, Y. Li, and T. Mei, "Incorporating copying mechanism in image captioning for learning novel objects," 2017.

[7] Y. Wu, L. Zhu, L. Jiang, and Y. Yang, "Decoupled novel object captioner," in *Proceedings of the 26th ACM international conference on Multimedia*, ACM, oct 2018.

[8] J. Lu, J. Yang, D. Batra, and D. Parikh, "Neural baby talk," 2018.