Course 08-05-2022

VR Mini-Project

Instructor: Prof. Vishwanath

Shaikh Fahed IMT2019079,
Rohit Oze IMT2019072,
Chirag Bansal IMT2019024.

1 IMAGE CAPTIONING

Our main goal is being able to build a model which generates a descriptive caption for any provided image.

1.1 Dataset - Flickr_8k:-

- All the images belonging to training, validation and test set are located in a single folder in the **Dataset - Flickr_8k** dataset.
- There are three different files, corresponding to each type of dataset, (train, test and validation set), where each file with file_name of images included in each dataset.
  - ***Flickr_8k.trainImages.txt***
  - ***Flickr_8k.testImages.txt***
  - ***Flickr_8k.validationImages.txt***
- As an image can be captioned in multiple ways, Every image is given five differing captions corresponding by five different persons. These files are stored in a ***Flickr_8k.token.txt file.***

1.2 Models Implemented:-

- **ResNet50, VGG-16** was used as the image encoder to encode the given images, and later on, they were given as the input to the model.
- **Keras** and **TensorFlow** have also been utilized. **Keras embedding layer** was helpful for the generation of word embeddings on the captions which were encoded formerly.
- Then, the embeddings have been fed to **LSTM** (Long short-term memory) Model, post which, the image and text features were put together and passed on to a decoder network, so that, it can generate the next word.
- We also applied **Greedy Search** and **Beam Search** in the proccess to generate the captions.
- **BLEU** (Bilingual Evaluation Understudy) **Score** was used to evaluate the generated captions.

1.3 Concepts Used:-

◇ **Encoder-Decoder Architecture**

An Encoder is used to encode the input into a specific form by any model that generates sequences. Later, these encoded inputs are decoded into a sequence, word by word, with the help of a Decoder.

◇ **Attention Networks**

Attention Networks have a pretty huge spanned usage in Deep-Learning. With this, a model has the ability to choose only those parts of an encoding which are assumed to be relevant to the running task. The similar mechanism can be utilized in any model where an Encoder gives output with multiple points in space or time.

Some pixels are considered more than the others in Image Captioning. In Machine Translation task, (Sequence to Sequence) some words are considered more important than others.

◇ **Beam Search**

Beam Search is useful for any language modelling problem because it finds the most optimal sequence. The decoder cannot be lazy and simply choose the words with best score at each decode-step.

◇ **Transfer Learning**

In Transfer Learning, you borrow from an existing model with the help of the parts of it in a new model. This is considered always better than training a new model from scratch. We can always

1.4 Results:-

We can notice that when we use ***RMSprop Optimizer*** and ***relu*** as our activation function, we achieved our best ***Bleu Score***. We also tried running the code for ***SGD, Adam and Adagrad*** for different number of epochs and activation functions ***(relu/softmax/tanh)***.

The below results are deduced from varying hyperparameters:-

Model → ResNet50
Dropout = 0.2

Optimizer	***Number of Epochs***	***Activation Function***	***Loss***	***Accuracy***	***Bleu Score***
SGD	1	relu	1.5530	0.7452	0.485160
Adam	1	relu	1.4070	0.7326	0.407885
RMSprop	1	relu	1.1970	0.7956	0.598349
Adagrad	1	tanh	1.3277	0.7725	0.396555
RMSprop	5	relu	1.0563	0.8197	0.619516

1.5 Code Snippets:-

Creating Dictionaries to map imageID and their corresponding captions.

```
Creating dictionaries to map image_id and their corresponding captions

[ ] train_captions={}
    for i in tqdm(range(len(train_image_names))):
      l=[caption for caption in(image_tokens[image_tokens["img_id"]==train_image_names["img_id"].iloc[i]].img_caption)]
      train_captions[train_image_names["img_id"].iloc[i]]=l

[ ] test_captions={}
    for i in tqdm(range(len(test_image_names))):
      l=[caption for caption in(image_tokens[image_tokens["img_id"]==test_image_names["img_id"].iloc[i]].img_caption)]
      test_captions[test_image_names["img_id"].iloc[i]]=l

[ ] validation_captions={}
    for i in tqdm(range(len(val_image_names))):
      l=[caption for caption in(image_tokens[image_tokens["img_id"]==val_image_names["img_id"].iloc[i]].img_caption)]
      validation_captions[val_image_names["img_id"].iloc[i]]=l
```

Data visualization and preprocessing

```
[ ] image_tokens = pd.read_csv("all_captions/Flickr8k.lemma.token.txt",sep='\t',names=["img_id","img_caption"])

[ ] train_image_names = pd.read_csv("all_captions/Flickr_8k.trainImages.txt",names=["img_id"])

[ ] test_image_names = pd.read_csv("all_captions/Flickr_8k.testImages.txt",names=["img_id"])

[ ] validation_image_names = pd.read_csv("all_captions/Flickr_8k.devImages.txt",names=["img_id"])
```

ResNet50 model for encoding images

```
model=ResNet50(include_top=False, weights='imagenet',pooling='avg',input_shape=(224,224,3))
model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	['input_1[0][0]']
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	['conv1_pad[0][0]']
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	['conv1_conv[0][0]']
conv1_relu (Activation)	(None, 112, 112, 64)	0	['conv1_bn[0][0]']
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	['conv1_relu[0][0]']
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	['pool1_pad[0][0]']
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	['pool1_pool[0][0]']

Encoding image encodings (features) from ResNet50 and creating dictionary train_features.

```
[ ] path="all_images/Flicker8k_Dataset/"
    train_features={}
    c=0
    for image_name in tqdm(train_captions):
      img_path = path + image_name
      img=image.load_img(img_path,target_size=(224,224))
      x = image.img_to_array(img)
      x = np.expand_dims(x, axis=0)
      x = preprocess_input(x)
      features = model.predict(x)
      train_features[image_name]=features.squeeze()
```

Greedy search function

```
[ ] def greedy_search(photo):
      photo=photo.reshape(1,2048)
      in_text='<start>'
      for i in range(max_length):
        sequence = [words_to_indices[s] for s in in_text.split(" ") if s in words_to_indices]
        sequence = pad_sequences([sequence], maxlen=max_length, padding='post')
        y_pred = model.predict([photo,sequence],verbose=0)
        y_pred = np.argmax(y_pred[0])
        word = indices_to_words[y_pred]
        in_text += ' ' + word
        if word == '<end>':
          break
      final = in_text.split()
      final = final[1:-1]
      return final
```

Predicting captions on test set using greedy search

```
[ ] i=0
    tot_score=0
```

```
[ ] for img_id in tqdm(test_features):
      i+=1
      photo=test_features[img_id]
      reference=[]
      for caps in test_captions[img_id]:
        list_caps=caps.split(" ")
        list_caps=list_caps[1:-1]
        reference.append(list_caps)
      candidate=greedy_search(photo)
      score = sentence_bleu(reference, candidate)
      tot_score+=score
```

```
[ ] avg_score=tot_score/i
    print()
    print("Bleu score on Greedy search")
    print("Score: ",avg_score)
```

2 Language Bias of the System

2.1 Stereotype-driven descriptions

Stereotypes are ideas about how other (groups of) people commonly behave and what they are likely to do. These ideas guide the way we talk about the world. I distinguish two kinds of verbal behavior that result from stereotypes:

1. Linguistic bias
2. Unwarranted inferences.

Linguistic bias

A systematic asymmetry in word choice as a function of the social category to which the target belongs is called the Linguistic Bias.

In data, adjectives are sometimes used to generate 'More narrow labels (or subtypes) for individuals who do not fit with general social category expectations.

Unwarranted inferences

Unwarranted inferences are statements about the subject(s) of an image that go beyond what the visual data alone can tell us. They are based on additional assumptions about the world.

- Many people who look Asian are captioned Chinese or Japanese.
- Almost all short haired people are tagged as men.
- In India, Most of the dark-skinned people are assumed to be South Indian, irrespective of where they belong.

All these are caused due to the bias data.

Picture 1 - *A thinking Asian Man*

Picture 2 - *A young South Indian Man*

Objective of System 1 - Modified is to search for language bias and reduce it as far as possible.

2.2 Measures to reduce Machine Learning Bias

In order to reduce Machine Learning Bias, one can -

- Test and validate to make sure that the results of the machine learning systems do not display bias due to algoriths or the data sets.
- Select training data that is correspondingly representative and large enough to counteract common types of Machine Learning Bias, for example, sample bias and prejudice bias.
- Keep a note of machine learning systems as they perform their tasks to ensure biases don't creep in overtime as the systems continue to learn as they work. Subpopulation analysis is one of the smarter ways to do monitor model performance over time.

2.3 Attributes of the Modified Model

- ***Representativeness***

  We are supposed to get used to the fact that crowdsourced descriptions of images are biased. Acknowledging this statement is a vital step in designing models that can accomodate data based on a mixture of facts and stereotypes around the world.

- ***Stereotypes and Interpretation***

  While stereotypes might be beneficial to systems that have to interpret human descriptions and determine likely referents of those descriptions.

- ***Neutralizing Stereotypes for production***

  One way to improve is to work with multi-lingual data. We propose a model that generates image descriptions given data from multiple languages. (German and English)