Join_text_and_image_representation

July 8, 2020

Student name - Name : **Bhavya-deepthi.kanaparthi**

1 Join text and image representation

[1]:
```
import tensorflow
from tensorflow.keras.models import Sequential, load_model
import numpy as np
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input,
↪decode_predictions
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.imagenet_utils import preprocess_input
```

[2]:
```
resnet_model = ResNet50(weights='imagenet' , include_top=False)
```

[3]:
```
def extract_features(img_path):
    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 = resnet_model.predict(x)
    return np.expand_dims(features.flatten(), axis=0)
```

2 Image representation

[60]:
```
features =np.load('resnet50-features.10k.npy')
print(features.shape)
```

(10000, 2048)

[ ]:
```

```

3 Text Representation

[4]:
```
def load(captions_filename,features_filename): features =
    np.load(features_filename)
    images = [] texts = []
    with open(captions_filename) as fp:
        for line in fp:
            tokens = line.strip().split()
            images.append(tokens[0]) texts.append('
            '.join(tokens[1:]))
    return features, images, texts
```

[5]:
```
feature , images , texts = load( "annotations.10k.txt","resnet50-features.10k.
↪npy")
```

[6]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences tokenizer
= Tokenizer()
tokenizer.fit_on_texts(texts)
sequences =tokenizer.texts_to_sequences(texts) captions =
pad_sequences(sequences, maxlen=16)
```

[7]:
```
vocab = tokenizer.word_index
vocab['<eos>'] = 0
```

[8]:
```
import json
with open('vocab.json', 'w') as fp: # save the vocab
    fp.write(json.dumps(vocab))
```

3.1 load the pretrained embedded model

[9]:
```
import embedding
embedding_weights = embedding.load(vocab, 100,'glove.twitter.27B.100d.filtered.
↪txt')
```

loading embeddings from "glove.twitter.27B.100d.filtered.txt"

4 putting together the model

[10]:
```
from keras.layers import Input, Dense, Embedding, GRU
image_input = Input(shape=(2048,))
caption_input = Input(shape=(16,)) noise_input =
Input(shape=(16,))
```

[11]:
```
caption_embedding = Embedding(len(vocab), 100,
↪input_length=16,weights=[embedding_weights])
caption_rnn = GRU(256)
image_dense = Dense(256, activation='tanh')
```

4.1 creation of pipe line

[12]:
```
image_pipeline = image_dense(image_input)
caption_pipeline =caption_rnn(caption_embedding(caption_input))
noise_pipeline = caption_rnn(caption_embedding(noise_input))
```

4.2 compute the dot product between image and caption

[13]:
```
from keras.layers import merge
positive_pair = merge.dot([image_pipeline, caption_pipeline] , axes=1,
↪normalize=False )
negative_pair = merge.dot([image_pipeline, noise_pipeline], axes=1,
↪normalize=False)
output = merge.concatenate([positive_pair, negative_pair] )
```

[14]:
```
positive_pair ,negative_pair,output
```

[14]: (<tf.Tensor 'dot/Squeeze:0' shape=(None, 1) dtype=float32>,
<tf.Tensor 'dot_1/Squeeze:0' shape=(None, 1) dtype=float32>,
<tf.Tensor 'concatenate/concat:0' shape=(None, 2) dtype=float32>)

4.3 create multiple models

[15]:
```
from tensorflow.keras import Model
training_model = Model(inputs=[image_input, caption_input, noise_input],
↪outputs=output)
image_model = Model(inputs=image_input, outputs=image_pipeline) caption_model =
Model(inputs=caption_input,outputs=caption_pipeline)
```

5 Custom loss

[16]:
```
from keras import backend as K
def custom_loss(y_true, y_pred): positive
    = y_pred[:,0] negative = y_pred[:,1]
    return K.sum(K.maximum(0., 1. - positive + negative))
```

[17]:
```
def accuracy(y_true, y_pred): positive =
    y_pred[:,0] negative = y_pred[:,1]
```

```
    return K.mean(positive >negative)
```

[18]:
```
training_model.compile(loss=custom_loss,optimizer='adam',metrics=[accuracy])
```

6 Training

6.1 spiting the model

[61]:
```
noise = np.copy(captions)
fake_labels = np.zeros((len(features), 1))
X_train =       ( [features[:9000], captions[:9000], noise[:9000]])Y_train = (
fake_labels[:9000])
X_valid = [features[-1000:], captions[-1000:],noise[-1000:]] Y_valid =
fake_labels[-1000:]
```

[62]:
```
### actual trainig
```

[63]:
```
for epoch in range(10):
    np.random.shuffle(noise) # don't forget to shuffle mismatched captions
    training_model.fit(X_train, Y_train,validation_data=[X_valid, Y_valid],
↪epochs =1,batch_size=64)
```

```
141/141[==============================]  - 20s 141ms/step - loss:  4.3636  -
accuracy: 0.9726 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 142ms/step - loss:  4.5233  -
accuracy: 0.9720 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 141ms/step - loss:  4.1314  -
accuracy: 0.9740 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 141ms/step - loss:  4.4201  -
accuracy: 0.9725 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 22s 153ms/step - loss:  4.4955  -
accuracy: 0.9735 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 142ms/step - loss:  4.3632  -
accuracy: 0.9743 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 143ms/step - loss:  3.9072  -
accuracy: 0.9767 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 142ms/step - loss:  3.9071  -
accuracy: 0.9771 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 142ms/step - loss:  3.8084  -
accuracy: 0.9769 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
141/141[==============================]  - 20s 141ms/step - loss:  3.9775  -
accuracy: 0.9743 - val_loss: 0.0000e+00  - val_accuracy: 0.0000e+00
```

6.2 saving model and representation

[64]:
```
image_model.save('model.image') caption_model.save('model.caption')

np.save('caption-representations',caption_model.predict(captions)) np.save('image-
representations', image_model.predict(feature))
```

```
INFO:tensorflow:Assets written to: model.image/assets
INFO:tensorflow:Assets written to:model.caption/assets
```

7 Captioning novel images

[65]:
```
from keras.models import load_model
image_model = load_model('model.image',compile=False)
caption_model =load_model('model.caption',compile=False)
```

7.1 load representation

[66]:
```
import numpy as np
caption_representations =np.load('caption-representations.npy')
image_representations = np.load('image-representations.npy')
```

[67]:
```
from keras.preprocessing.sequence import pad_sequences
import json
vocab = json.loads(open('vocab.json').read())
def preprocess_texts(texts): output = []
    for text in texts:
        output.append([vocab[word] if word in vocab else 0 for word in text.
↪split()])
    return pad_sequences(output, maxlen=16)
```

[68]:
```
def generate_caption(image_filename, n=10):
    # generate image representation for new image
    image_representation = image_model.predict(extract_features(image_filename))
    # compute score of all captions in the dataset
    scores = np.dot(caption_representations,image_representation.T).flatten()
    # compute indices of n best captions
    indices = np.argpartition(scores, -n)[-n:]

    indices = indices[np.argsort(scores[indices])]
    # display them
    for i in [int(x) for x inreversed(indices)]: print(scores[i],
        texts[i])
```

[75]:
```
#generate_caption('images.jpg')
```

8 searching for images

[70]:
```
def search_image(caption, n=10):
    caption_representation =caption_model.predict(preprocess_texts([caption])) scores =
    np.dot(image_representations, caption_representation.T).flatten() indices =
    np.argpartition(scores, -n)[-n:]
    indices = indices[np.argsort(scores[indices])]
    for i in [int(x) for x inreversed(indices)]: print(scores[i],
        images[i])
```

[71]:
```
search_image('a man in the snow on someskis')
```

15.56558 COCO_val2014_000000022626.jpg
13.151729 COCO_val2014_000000007938.jpg
11.980955 COCO_val2014_000000276893.jpg
11.921771 COCO_val2014_000000380906.jpg
11.8630085 COCO_val2014_000000392575.jpg
11.753822 COCO_val2014_000000012946.jpg
11.510327 COCO_val2014_000000134343.jpg
11.507981 COCO_val2014_000000119773.jpg
11.371106 COCO_val2014_000000204360.jpg
11.25193 COCO_val2014_000000112769.jpg

[71]:
```

```