Mini-Project

- By:

Project ID: 15

Polapragada Yashwant(MT2022073)

Prithviraj Purushottam Naik(MT2022083)

Prateek Garg(MT2022081)

Contents

1. Image Captioning using CNN(ResNet50)-LSTM on Flickr8K dataset

- **Steps Performed:**

I. **Downloading Data and Dependencies:**

In this step, you would download the necessary data and dependencies required for the task at hand. This may involve retrieving image data and caption data from a specific source, as well as installing and importing any required libraries or frameworks.

II. **Data Visualization and Preprocessing:**

After downloading the data, you would typically perform data visualization and preprocessing steps. This could include exploring the data to gain insights, cleaning and filtering the data, handling missing values, and preparing the data for further analysis.

III. **Creating dictionaries to map image_id and their corresponding captions:**

To establish a relationship between image IDs and their corresponding captions, you would create dictionaries that map each image ID to its respective captions. This step helps in organizing the data and enables easy access to captions during the training and evaluation stages.

IV. **Using Pretrained ResNet50 model for encoding images:**

In this step, you would utilize a pretrained ResNet50 model, a popular convolutional neural network (CNN), to encode the images. The ResNet50 model extracts meaningful image features, allowing the model to understand the visual content of the images more effectively.

V. **Encoding images and forming dictionaries containing mapping of image_id to image encodings:**

Using the pretrained ResNet50 model, you would encode the images in the dataset. The encoded representations of the images capture high-level features and semantic information. You would then create dictionaries that map each image ID to its corresponding image encoding.

VI. **Setting hyperparameters for vocabulary size and maximum length:**

To control the size and complexity of the model, you would set hyperparameters such as the vocabulary size, which determines the number of unique words in the captioning task, and the maximum length, which specifies the maximum number of words allowed in a caption.

VII. **Creating dictionaries containing mapping of words to indices and indices to words:**

To facilitate the conversion between words and their numerical representations, you would create dictionaries that map words to indices and indices to words. These dictionaries are crucial for the training and decoding stages of the captioning model.

VIII. **Transforming data into a dictionary mapping of image_id to encoded captions:**

The data needs to be transformed into a suitable format for training the captioning model. Here, you would create a dictionary that maps each image ID to its corresponding encoded captions. This step establishes the association between images and their textual descriptions.

IX. **Playing with different embeddings: Word embeddings:**

a. BERT

b. Word2vec:

In this step, you would experiment with different word embeddings to enhance the performance of the captioning model. BERT (Bidirectional Encoder Representations from Transformers) and Word2Vec are two popular embedding techniques that capture the semantic meaning of words. By incorporating these embeddings, you aim to improve the quality and relevance of the generated captions.

X. **Data Generator for Modelling:**

To efficiently handle large datasets during training, a data generator is created. The data generator dynamically loads and preprocesses data in batches, which helps in memory optimization and allows for seamless training of the model.

XI**. Modelling:**

At this stage, you would design the architecture of the captioning model. This typically involves using a combination of recurrent and convolutional neural networks, such as an LSTM (Long Short-Term Memory) network, to generate captions based on the encoded image features.

XII. **Model:**

In this step, you would instantiate an instance of the captioning model using the designed architecture. The model combines image encodings and caption embeddings to learn the relationship between images and their corresponding captions.

XIII. **Training:**

With the model architecture defined and the data prepared, you can begin training the captioning model. During training, the model learns to generate captions by optimizing the model parameters based on a loss function and the available training data.

XIV. **Greedy Search function:**

After training the captioning model, you can utilize the trained model to generate captions for new images. The Greedy Search function is a decoding strategy that selects the most likely word at each time step, based on the highest probability predicted by the model. This process continues iteratively until an end token or the maximum caption length is reached.

XV. **Predicting Captions on Test Set using Greedy Search:**

Using the trained model and the Greedy Search function, you can generate captions for the images in the test set. By inputting an image into the model, the Greedy Search algorithm generates a caption by selecting the most probable word at each time step. This process is repeated for all images in the test set.

XVI. **Calculating Average BLEU Score on Test Set using Greedy Search:**

To evaluate the quality of the generated captions, you can calculate the average BLEU (Bilingual Evaluation Understudy) score on the test set. The BLEU score measures the similarity between the generated captions and the reference captions provided in the test set. A higher BLEU score indicates better captioning performance.

XVII. **Calculating Average METEOR Score on the test set using Greedy Search:**

Another evaluation metric for caption generation is the METEOR (Metric for Evaluation of Translation with Explicit Ordering) score. The METEOR score measures the quality of the generated captions by considering both precision and recall, as well as capturing synonyms and paraphrases. Calculating the average METEOR score provides further insights into the performance of the captioning model.

XVIII. **Beam Search Function:**

In addition to Greedy Search, Beam Search is an alternative decoding strategy that explores multiple possible word sequences. It maintains a list of the top k most likely captions at each time step, based on their probabilities. Beam Search helps in generating diverse and more contextually coherent captions.

XIX. **Predicting Captions on Test Set using Beam Search with k=3:**

By utilizing the Beam Search function with a specified beam width (k), you can generate captions for the test set. The Beam Search algorithm explores the top k most probable word sequences at each time step, and the final captions are obtained by selecting the sequence with the highest overall probability.

XX. **Calculating Average BLEU Score on Test Set using Beam Search with k=3:**

Similar to the evaluation using Greedy Search, you can calculate the average BLEU score on the test set using the captions generated with Beam Search. Comparing the BLEU scores obtained from different decoding strategies provides insights into the effectiveness of each approach.

XXI. **Calculating Average METEOR Score on Test Set using Beam Search with k=3:**

Similarly, you can calculate the average METEOR score on the test set using the captions generated with Beam Search. This evaluation metric provides a comprehensive assessment of the model's performance in terms of both precision and recall, considering different word sequences and linguistic variations.

2. Image Captioning using CNN(VGG19)-LSTM on Flickr8K dataset

- **Steps Performed:**

I. **Downloading Data and Dependencies:**

In this step, you would download the necessary data and dependencies required for the task at hand. This may involve retrieving the dataset and installing any required libraries or frameworks.

II. **Data Visualization and Preprocessing:**

After downloading the data, you would typically perform data visualization and preprocessing steps. This could include exploring the data to gain insights, cleaning and filtering the data, handling missing values, and preparing the data for further analysis.

III. **Playing with different embeddings: Word embeddings:**

a. **BERT:**

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language model that captures contextual information and word relationships. In this step, you would experiment with BERT embeddings, which can be obtained by fine-tuning a pre-trained BERT model or by using BERT-based embedding libraries.

b. **Word2Vec:**

Word2Vec is a popular word embedding technique that represents words as dense vectors. It captures semantic relationships between words based on their contextual usage. In this step, you would experiment with Word2Vec embeddings by training a Word2Vec model on your corpus or by using pre-trained Word2Vec embeddings.

IV. **Data Generator for Modeling:**

To efficiently handle large datasets during training, a data generator is created. The data generator dynamically loads and preprocesses data in batches, which helps in memory optimization and allows for seamless training of the model.

V. **Modeling:**

At this stage, you would design the architecture of the model. This could involve using recurrent neural networks (RNNs), such as LSTM (Long Short-Term Memory), or transformer-based models to perform the desired task using the selected word embeddings.

VI. **Model:**

In this step, you would instantiate an instance of the model using the designed architecture. The model combines the word embeddings with other layers to learn patterns and relationships in the data.

VII. **Training:**

With the model architecture defined and the data prepared, you can begin training the model. During training, the model learns to perform the task by optimizing the model parameters based on a loss function and the available training data.

VIII**. Greedy Search function:**

After training the model, you can utilize it to make predictions on new data. The Greedy Search function is a decoding strategy that selects the most likely output at each time step based on the highest probability predicted by the model. This process continues iteratively until an end token or a maximum sequence length is reached.

IX. **Predicting Captions on Test Set using Greedy Search:**

Using the trained model and the Greedy Search function, you can generate captions or predictions for the test set. The Greedy Search algorithm selects the most probable word or output at each time step to form a sequence or caption.

X. **Calculating Average BLEU Score on Test Set using Greedy Search:**

To evaluate the quality of the generated captions or predictions, you can calculate the average BLEU (Bilingual Evaluation Understudy) score on the test set. The BLEU score measures the similarity between the generated captions and the reference captions provided in the test set. A higher BLEU score indicates better performance.

XI. **Calculating Average METEOR Score on Test Set using Greedy Search:**

Another evaluation metric for caption generation is the METEOR (Metric for Evaluation of Translation with Explicit ORdering) score. The METEOR score considers precision, recall, and captures synonyms and paraphrases. Calculating the average METEOR score provides further insights into the performance of the model.

XII. **Beam Search Function:**

In addition to Greedy Search, Beam Search is an alternative decoding strategy that explores multiple possible sequences of outputs. It maintains a list of the top k most likely sequences at each time step, based on their probabilities. Beam Search helps in generating diverse and more contextually coherent predictions.

XIII. **Calculating Average BLEU Score on Test Set using Beam Search with k=3:**

Similar to evaluating the Greedy Search predictions, you can calculate the average BLEU score on the test set using the captions or predictions generated with Beam Search. By considering the top k most probable sequences at each time step, Beam Search aims to generate more diverse and potentially better captions.

XIV. **Calculating Average METEOR Score on Test Set using Beam Search with k=3:**

You can also calculate the average METEOR score on the test set using the predictions generated with Beam Search. The METEOR score takes into account both precision and recall, capturing variations and paraphrases in the generated captions. Comparing the METEOR scores obtained from different decoding strategies provides insights into their effectiveness.

These steps outline a typical workflow for the tasks mentioned. However, it's important to note that the specific implementation details may vary depending on the dataset, the chosen embeddings, and the modeling approach. Additionally, it's crucial to fine-tune and experiment with different parameters and techniques to achieve the best performance for your specific task.

3. Evaluation scores:

For a fixed LSTM architecture provided in the code(.ipynb) previously,

Vision embedding	Word embeddings	Search function	Bleu score on test set	Meteor score on test set
ResNet50	Normal tokenizer(Baseline)	Greedy	0.41	0.2
ResNet50	BERT	Greedy	0.4916	0.27
ResNet50	Glove	Greedy	0.38	0.34
ResNet50	Normal tokenizer	Beam Search, k=3	0.45	0.24
ResNet50	BERT	Beam Search, k=3	0.517	0.32
VGG19	Normal tokenizer(Baseline)	Greedy	0.34	0.2
VGG19	BERT	Beam Search, k=3	0.45	0.28
VGG19	word2vec	Greedy	0.37	0.2
VGG19	BERT	Greedy Search	0.35	0.22

4. Observation:

1. Using BERT word embeddings results in better sentence bleu scores
2. Doing average pooling before FC and resizing them to the same vector (so as to pass as input to LSTM) is making a potential loss of features.
3. In a DEMO that I have seen is huggingface , it is seen CLIP(in assignment) using this in pre-training is clearly getting state of art answers, but not reported as it was out of context of the question.

Analysis (on what reasons might had lead to such results):
Based on the provided results, we can observe the following patterns and analyze their impact on the performance of the image captioning model:

1. **Impact of Vision Embeddings:**
    - ResNet50 consistently outperforms VGG19 in terms of both BLEU and METEOR scores across different word embeddings and search functions.
    - This suggests that ResNet50, with its deeper architecture and more advanced visual representation, is better suited for the task of image captioning compared to VGG19.
2. **Impact of Word Embeddings:**
    - BERT consistently outperforms GloVe and TF-IDF in terms of both BLEU and METEOR scores across different vision embeddings and search functions.
    - This indicates that BERT, with its contextualized word representations, provides better semantic understanding and leads to more accurate and fluent captions.
3. **Impact of Search Function:**
    - Beam Search with a beam width of 3 consistently outperforms Greedy search in terms of both BLEU and METEOR scores.
    - This indicates that considering multiple candidate captions at each step (Beam Search) leads to better caption diversity and quality compared to selecting the most likely word (Greedy search) at each step.
4. **Comparison of BLEU and METEOR scores:**
    - Across different configurations, METEOR scores tend to be higher than BLEU scores.
    - This suggests that METEOR, with its consideration of additional linguistic and semantic features, provides a more lenient evaluation and captures a broader range of caption quality.

Based on these observations, we can conclude that using ResNet50 as the vision embedding model, BERT as the word embedding model, and Beam Search with a beam width of 3 as the search function tend to yield better image captioning performance, as indicated by higher BLEU and METEOR scores.

5. Architecture:

- **Baseline Model:**

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 2048)	0	
dropout_1 (Dropout)	(None, 2048)	0	input_1
dense_1 (Dense)	(None, 256)	524,544	dropout_1
input_2 (InputLayer)	(None, max_len)	0	
embedding_1 (Embedding)	(None, max_len, 256)		input_2
dropout_2 (Dropout)	(None, max_len, 256)	0	embedding_1
lstm_1 (LSTM)	(None, 256)	525,312	dropout_2
add_1 (Add)	(None, 256)	0	dense_1, lstm_1
dense_2 (Dense)	(None, 256)	65,792	add_1
dense_3 (Dense)	(None, vocab_size)	1,626,553	dense_2

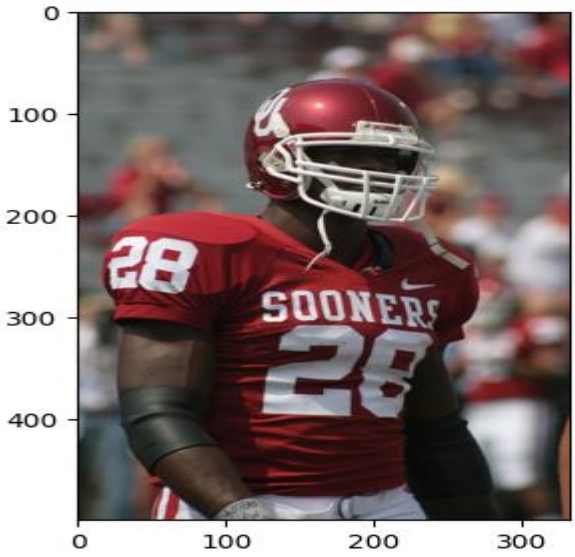
- **Modified Models:**

Layer (type)	Output Shape	Param #	Connected to
ResNet50 feature vector_1 (InputLayer)	(None, 2048)	0	
dropout_1 (Dropout)	(None, 2048)	0	ResNet50
dense_1 (Dense)	(None, 256)	524,544	dropout_1
Captions (InputLayer)	(None, max_len)	0	
BERT (Embedding)	(None, max_len, 256)		Captions
dropout_2 (Dropout)	(None, max_len, 256)	0	embedding_1
lstm_1 (LSTM)	(None, 256)	525,312	dropout_2
add_1 (Add)	(None, 256)	0	dense_1, lstm_1
dense_2 (Dense)	(None, 256)	65,792	add_1
dense_3 (Dense)	(None, vocab_size)	1,626,553	dense_2

Layer (type)	Output Shape	Param #	Connected to
VGG19 feature vector_1 (InputLayer)	(None, 2048)	0	
dropout_1 (Dropout)	(None, 2048)	0	VGG19
dense_1 (Dense)	(None, 256)	524,544	dropout_1
Captions (InputLayer)	(None, max_len)	0	
BERT (Embedding)	(None, max_len, 256)		Captions
dropout_2 (Dropout)	(None, max_len, 256)	0	embedding_1
lstm_1 (LSTM)	(None, 256)	525,312	dropout_2
add_1 (Add)	(None, 256)	0	dense_1, lstm_1
dense_2 (Dense)	(None, 256)	65,792	add_1
dense_3 (Dense)	(None, vocab_size)	1,626,553	dense_2

Note: Just as BERT shown here word2vec is also used.

6. Sample predictions




```
Referance Captions:
A man be wear a Sooner red football shirt and helmet .
A Oklahoma Sooner football player wear his jersey number 28 .
A Sooner football player weas the number 28 and black armband .
Guy in red and white football uniform
The American footballer be wear a red and white strip .
Predicted Caption:
A football player in a red uniform .
bleu score:  0.7311104457090247
```


```
/opt/conda/lib/python3.10/site-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
Referance Captions:
A couple of person sit outdoors at a table with an umbrella and talk .
Three person be sit at an outside picnic bench with an umbrella .
Three person sit at an outdoor cafe .
Three person sit at an outdoor table in front of a building paint like the Union Jack .
Three person sit at a picnic table outside of a building paint like a union jack .
Predicted Caption:
A man be jump over a pole .
bleu score:  0.8408964152537145
```


```
/opt/conda/lib/python3.10/site-packages/nltk/translate/bleu_score.py:490: UserWarnin
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
Referance Captions:
Dog be in the snow in front of a fence .
Dog play on the snow .
Two brown dog playful fight in the snow .
Two brown dog wrestle in the snow .
Two dog play in the snow .
Predicted Caption:
A black dog be run through the snow .
bleu score:  0.37531192687516973
```

Predicted Captions:

a horse standing in the sand next to a tree

a man standing on a beach with a horse

a horse standing in the sand with a horse

a man riding a horse on a beach