UDACITY

‹ Return to Classroom DISCUSS ON STUDENT HUB

Deploying a Sentiment Analysis Model

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Great job on project implementation!!!, you have correctly answered and implemented all "To Dos" in project notebook.
You have now gained good knowledge on deploying machine learning models in AWS cloud. This is very significant skill for ML professional as the success of overall project depends upon how ML application are productionalized after successfully building model.

Congratulations for finishing this project. All the best!!

Files Submitted

The submission includes all required files, including notebook, python scripts and html files.

Well done including all required files.

Preparing and Processing Data

Answer describes what the pre-processing method does to a review.

Well done including all required files.

1.Removes the html tags
2.Converts text to lower case.
3.Split string into words

4.Remove stopwords
5.Stems each word using porter stemmer.

The `build_dict` method is implemented and constructs a valid word dictionary.

Good work building word dictionary from sentences!!

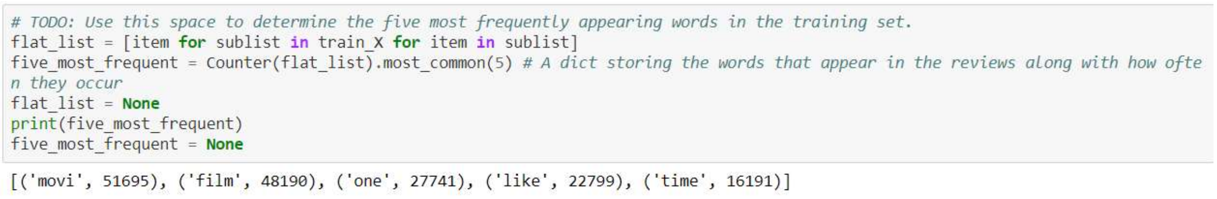
Here's another approach for building word_count

```
 word_count = {} # A dict storing the words that appear in the reviews along with
how often they occur

    for review in data:
        for word in review:
            word_count[word] = word_count.get(word, 0) + 1


    sorted_words = sorted(word_count.keys(), key=lambda x: -word_count[x])
```

Notebook displays the five most frequently appearing words.

Well done!! You have correctly evaluated five most frequent appearing words.

1: movi
2: film
3: one
4: like
5: time



```
# TODO: Use this space to determine the five most frequently appearing words in the training set.
flat_list = [item for sublist in train_X for item in sublist]
five_most_frequent = Counter(flat_list).most_common(5) # A dict storing the words that appear in the reviews along with how ofte
n they occur
flat_list = None
print(five_most_frequent)
five_most_frequent = None
```

```
[('movi', 51695), ('film', 48190), ('one', 27741), ('like', 22799), ('time', 16191)]
```

Answer describes how the processing methods are applied to the training and test data sets and what, if any, issues there may be.

Good answer provided here also.
In addition, the idea is to prevent data leakage. Data leakage occurs when data from the training set leaks to the test set.

- preprocess_data is applied per record on both the training and test sets, so there is no issue coming from it.
- convert_and_pad_data doesn't cause an issue also because word_dict which is used to transform the reviews to integers was constructed using only the training data. If the test data was also used in creating word_dict, then predictions would be biased due to the data leakage. The test data is meant to be unseen data by the model.

Build and Train the PyTorch Model

The train method is implemented and can be used to train the PyTorch model.

Well done completing the train method to train the model provided.

Use torch.nn.utils.clip_grad_norm to keep the gradients within a specific range (clip). In RNNs the gradients tend to grow very large which may cause exploding gradient problem, clipping them helps to prevent this from happening.

```
        model.zero_grad()
        output = model(batch_X)
        loss = loss_fn(output, batch_y)
        loss.backward()

        torch.nn.utils.clip_grad_norm_(model.parameters(), 2)
        optimizer.step()

        total_loss += loss.data.item()
    print("Epoch: {}, BCELoss: {}".format(epoch, total_loss / len(train_loader)))
```

The RNN is trained using SageMaker's supported PyTorch functionality.

Well done!! BCELoss decreases with subsequent epochs shows model has trained well.

```
Using device cuda.
Get train data loader.
Model loaded with embedding_dim 32, hidden_dim 200, vocab_size 5000.
Epoch: 1, BCELoss: 0.6711890843449807
Epoch: 2, BCELoss: 0.5896885468035328
Epoch: 3, BCELoss: 0.4996530219000213
Epoch: 4, BCELoss: 0.4245543315702555
Epoch: 5, BCELoss: 0.40212314165368374
Epoch: 6, BCELoss: 0.35693952319573385
Epoch: 7, BCELoss: 0.3198749082429068
Epoch: 8, BCELoss: 0.2937151847445235
Epoch: 9, BCELoss: 0.27433232658979845
```

Deploy the Model for Testing

The trained PyTorch model is successfully deployed.

Good work deploying model to 'ml.m4.xlarge' instance.

```
# TODO: Deploy the trained model
predictor = estimator.deploy(initial_instance_count = 1, instance_type = 'ml.m4.xlarge')
```

Use the Model for Testing

Answer describes the differences between the RNN model and the XGBoost model and how they perform on the IMDB data.

Excellent!! RNN or LSTM are sequence based model as they store context of the sentence in the cell state. When further optimized and fine tuned they can outperform XGBoost for sentiment based classification.

The test review has been processed correctly and stored in the `test_data` variable.

Well done preprocessing test_review data by applying review_to_words and convert_and_pad!!

```
# TODO: Convert test_review into a form usable by the model and save the results in test_data
test_data, length = convert_and_pad(word_dict, review_to_words(test_review), pad=500)
test_data = np.hstack(([length], test_data))
test_data = test_data.reshape(1, -1)
test_data, length
```

The `predict_fn()` method in `serve/predict.py` has been implemented.

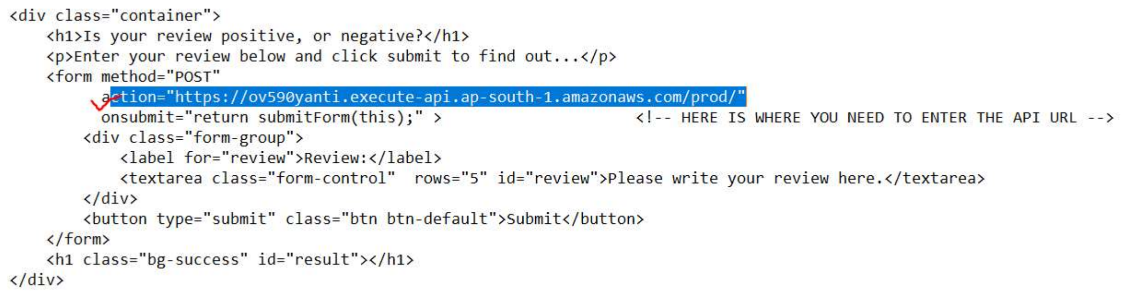
Well done!! Same method has been used in script file to preprocess test review.

```
data_X, data_len = convert_and_pad(model.word_dict, review_to_words(input_data), pad=500)
```

Deploying the Web App

The model is deployed and the Lambda / API Gateway integration is complete so that the web app works (make sure to include your modified `index.html`).

Well done creating lambda function and integrating with API endpoint.



```
<div class="container">
    <h1>Is your review positive, or negative?</h1>
    <p>Enter your review below and click submit to find out...</p>
    <form method="POST"
          action="https://ov590yanti.execute-api.ap-south-1.amazonaws.com/prod/"
          onsubmit="return submitForm(this);" >                    <!-- HERE IS WHERE YOU NEED TO ENTER THE API URL -->
        <div class="form-group">
            <label for="review">Review:</label>
            <textarea class="form-control"  rows="5" id="review">Please write your review here.</textarea>
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
    </form>
    <h1 class="bg-success" id="result"></h1>
</div>
```

Answer gives a sample review and the resulting predicted sentiment.

Good work on test prediction and considering review which are ambiguous and hard for model to predict sentiment specifically review which consists of sarcasm. This gives an idea about model shortcomings or limitations.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START