UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Deploying a Sentiment Analysis Model

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations on project completion. Very well done. Just a few points
1) XGBOOST might have given slightly better results in this case. A few more LSTM layers along with hyperparameter tuning will help. RNNs work better on data with some context like textual data. Therefore, it will work better than any machine learning algorithm .

2) nested for loops work really slow. If you are looking for word within a certain set of words, try set(word_list1).intersection(set(word_list2)). This will give you the the common words in the two lists. Look at this blog -- https://www.geeksforgeeks.org/intersection-function-python/

3) If you wish to look at the tutorials from amazon, you can have a look at this link https://docs.aws.amazon.com/sagemaker/latest/dg/gs.html

4) for more pre processing, check this link http://snowball.tartarus.org/algorithms/porter/stemmer.html

5) for more understanding and comparisons, read https://datascience.stackexchange.com/questions/2504/deep-learning-vs-gradient-boosting-when-to-use-what

I wish you all the best. Happy learning

Files Submitted

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

Thanks for submitting all the files

Preparing and Processing Data

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

You are right . Here are all the points

Removes punctuation, ensuring only alphabets and numbers are present.
Ensures all alphabets are in lower case.
Removes stopwords such as "the", since they would not add much information about the reviews.
Returns a list of the stemmed words.
removing HTML tags and tokenizing the words in the review,

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

Your intuition is correct. It makes sense to construct a dictionary for storing the converted words into integer representation . Also check out this link for better reference https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/
Try to use python comprehensions for better performance. https://www.geeksforgeeks.org/comprehensions-in-python/. You can also use the Counter library package.
https://docs.python.org/2/library/collections.html#collections.Counter

Notebook displays the five most frequently appearing words.

Your answer is absolutely correct. It is also predictable that these would be the most occurring words as one tends to use these words a lot i.e some specific words while writing reviews. Though it will be interesting to check the least occurring words and the context they are used in

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

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

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 a PyTorch Model

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

The train method is implemented correctly. The model is initialized to train mode, batches are loaded, the optimizer is initialized, the model is called with X values, the loss function is called with output and Y values and the loss is back propagated correctly. Well done.

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

Deploy a Model for Testing

The trained PyTorch model is successfully deployed.

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.

XGBoost is a tree-based algorithm which has no idea of sequential structure whereas the RNN/LSTM is specifically designed to take this into account as it possesses an internal state variable that can be thought of as a memory of the model. RNN with LSTM cells are designed to take a sequential order into account which in theory should make them superior to approach tasks where the order (i.e. words or time) is crucial. I would think that in general when it comes to speech and text processing this is very important so I think RNN with LSTM cells is better for sentiment analysis.

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

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

Deploying a 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`).

Answer gives a sample review and the resulting predicted sentiment.

Very well done, The predictions have come out to be correct in this case. You should try more confusing reviews and check the functionality of the trained model
Example - The movie was good in the initial half but turned to be boring in the end. The ending could have been better.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START