Practice: Using Deep Learning to Classify Profitable Movies Using the R API

The data set **MOVIE_MONEY_CLEAN** contains the cleaned text from the original **MOVIE_MONEY** data set. For the text in each review, stop words and non-letters have been removed, words have been stemmed, and all tokens were changed to lower case.

Name	Model Role	Measurement Level	Description
PROFIT	Target	Binary	1 = movie made more money than the budget 0 = otherwise
TITLE	Nominal	Text	Movie Title Cleaned
OVERVIEW	Input	Text	Description of the movie

The data set **MOVIE_MONEY_EMBED** contains the Global Vectors for Word Representation (GLOVE) for each term in the **MOVIE_MONEY_CLEAN** data set. The GLOVE was created from word-word co-occurrence statistics from the **MOVIE_MONEY_CLEAN** corpus using an unsupervised learning algorithm. The vectors of dimension 100 show the linear substructure of the word vector space.

Name	Model Role	Measurement Level	Description
VOCAB.TERM	Input	Nominal	Individual terms from the cleaned corpus
X1-X100	Input	Interval	Word representations in 100 dimensions

1. From the Jupyter Lab Home directory, select the plus symbol to open a new page and then select R under Notebook to create a new notebook.
2. Load the SWAT and repr packages. Use the options functions to change the size of the graphical output.
3. Connect to CAS and create a connection object called **conn**.
4. Load the **movie_money_clean.csv** file onto the CAS server and name it **movie_clean**. Then print the dimensions and head of the table using SWAT functionality.
5. Use the freq action from the simple action set to view the frequency of the movies that earned a profit.
6. Use the srs action from the sampling action set to partition the data into 70% training, 15% validation, and 15% for testing by adding a partition indicator to the CAS table.
7. Use the shuffle action from the table action set to randomize the observations and avoid a potential ordering bias in the deep learning model.
8. Load the **movie_money_embed.csv** onto the CAS server and view the dimensions and head of the word embedding data.
9. Use the deepLearn action set to build a gated recurrent unit neural network with one input layer, two GRU hidden layers, and an output layer.
   - Use the buildModel action to initialize the RNN and then add an input layer.
   - Connect the input layer to a GRU hidden layer with 15 neurons, set the activation function to auto, set initialization to Xavier, and set the output type to same length.
   - Connect this hidden layer to a another GRU hidden layer with the same arguments except set output type to encoding.
   - Finally, connect the second hidden layer to the output layer and set the error function to auto.
   - View the model to make sure the structure is correct using the modelInfo action.
10. Use the dlTrain action to train the GRU model using the profit variable as the target and the overview variable as the input. Train the model using the Adam optimization algorithm and a learning rate of 0.05. Use mini batch sizes of 50 and train for 20 epochs. Be sure to save the weights to score the test data after the model is built.
11. Use the dlScore action to score the test data and view the misclassification error.
12. Notice in the optimization history of the dlTrain action that the model over fit on the training data resulting in a large validation error comparatively. Regularize the previous GRU model by building the model again but include a dropout of 0.20 in each GRU hidden layer. Train the new model with the same arguments for the dlTrain action and view the changes in the optimization history.
13. Score the test data using the GRU model with regularization.
14. End the CAS session.