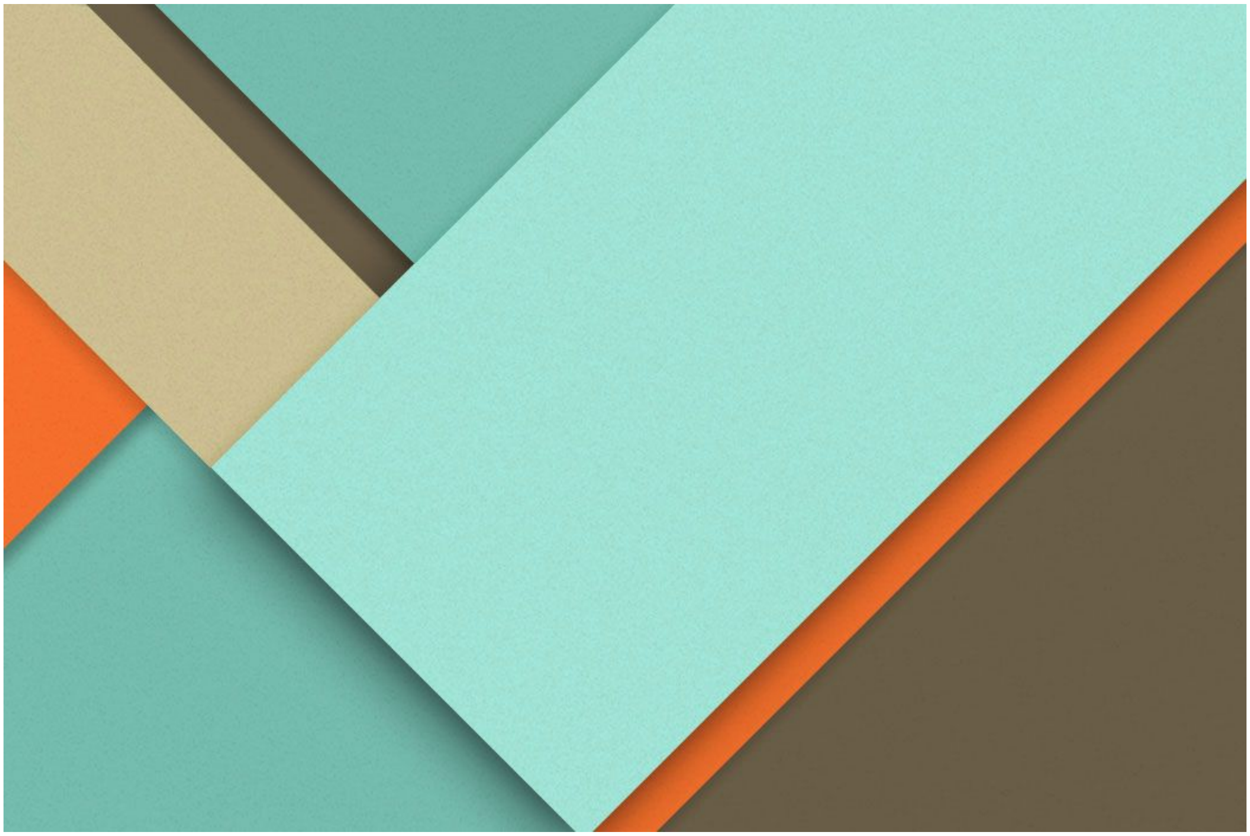


Assignment 2 Report

Text Classification Using Deep Learning

CS6301.004 Data Science with R

Sankalpa Rath (sxr173830)

Sushrut Patnaik (sxp175331)

What is Text Classification?

Text classification (a.k.a. text categorization or text tagging) is the task of assigning a set of predefined categories to free-text. Text classifiers can be used to organize, structure, and categorize pretty much anything. For example, new articles can be organized by topics, support tickets can be organized by urgency, chat conversations can be organized by language, brand mentions can be organized by sentiment, and so on.

There are many approaches to automatic text classification, which can be grouped into three different types of systems:

- Rule-based systems
- Machine Learning based systems
- Hybrid systems

Classification using Deep Learning

One way to classify Text Data is to use Deep Learning networks. Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. We have used Keras neural network packages to achieve our goals. We have selected Sentiment Labelled Sentences Data Set from the UCI Text Datasets for this Assignment. We have worked on the Amazon review dataset which has 1000 reviews with 50% positive reviews and 50% negative reviews which are labelled using 1 and 0 respectively.

Goals

1. Preprocessing the dataset :
    - Removing stopwords, punctuations, numbers, single characters and whitespaces from the individual reviews.
    - Converting the words to list of integer indices and vectorizing the input data so that it can be fed to the neural network.
2. Splitting the dataset into training and testing data.
3. Create different deep network models and test their accuracy using GCloud.

Libraries Required

library(keras)

library(tm)

library(stringr)

require(caTools)

Models and Performance

No.	Code	Performance
1.	#model1 library(keras) model <- keras_model_sequential() %>% layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>% layer_dense(units = 16, activation = "relu") %>% layer_dense(units = 1, activation = "sigmoid") model %>% compile(optimizer = "rmsprop",loss = "binary_crossentropy", metrics = c("accuracy")) model %>% fit(x_train, y_train, epochs = 6,batch_size = 512) #plot of history based on validation data history <- model %>% fit(partial_x_train, partial_y_train, epochs = 20, batch_size = 512, validation_data = list(x_val, y_val)) results <- model %>% evaluate(x_test, y_test)	Loss : 0.6367 Accuracy : 0.6833 Link to Google Cloud Output

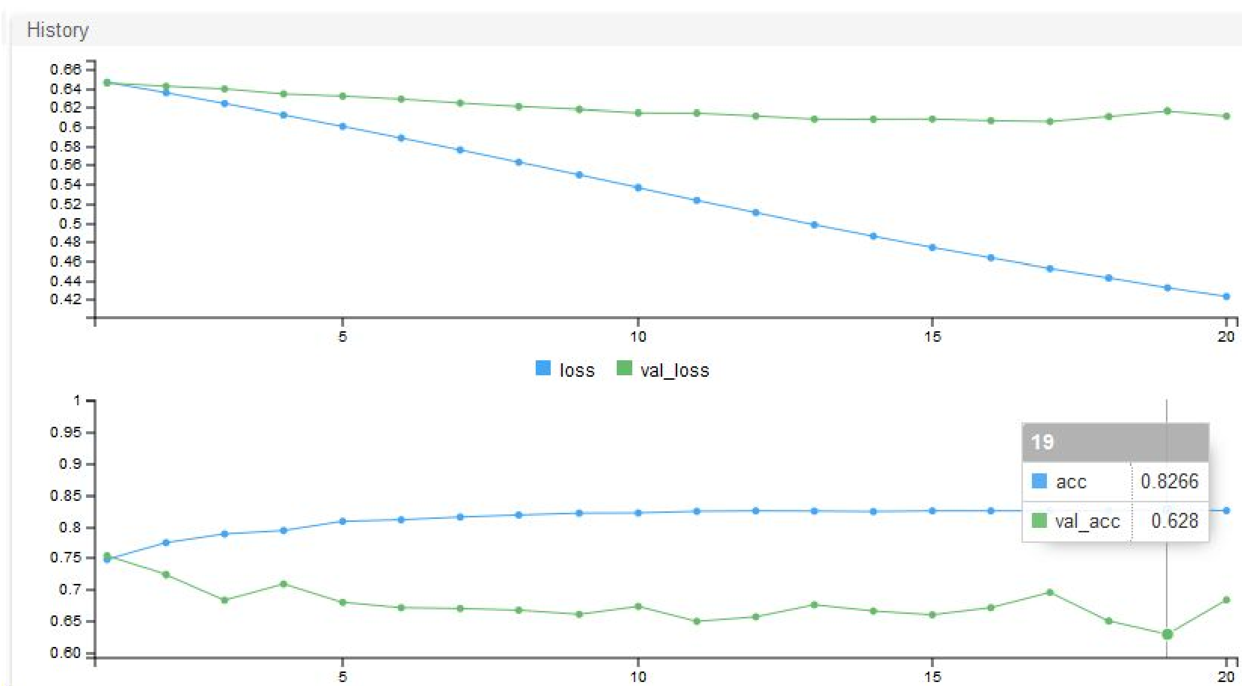
2.	#model 2 library(keras) model2 <- keras_model_sequential() %>% layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>% layer_dense(units = 16, activation = "relu") %>% layer_dense(units = 1, activation = "sigmoid") model2 %>% compile(optimizer = "rmsprop",loss = "mse", metrics = c("accuracy")) model2 %>% fit(x_train, y_train, epochs = 10,batch_size = 512) #plot of history based on validation data history <- model2 %>% fit(partial_x_train, partial_y_train, epochs = 20, batch_size = 512, validation_data = list(x_val, y_val)) results2 <- model2 %>% evaluate(x_test, y_test)	Loss : 0.221 Accuracy : 0.6547 Link to Google Cloud Output
3.	#model 3 library(keras) model3 <- keras_model_sequential() %>% layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 1, activation = "sigmoid") model3 %>% compile(optimizer = "rmsprop",loss = "poisson", metrics = c("accuracy")) model3 %>% fit(x_train, y_train, epochs = 10,batch_size = 512) #plot of history based on validation data history <- model3 %>% fit(partial_x_train, partial_y_train, epochs = 20, batch_size = 512,	Loss : 0.9546 Accuracy : 0.6589 Link to Google Cloud Output

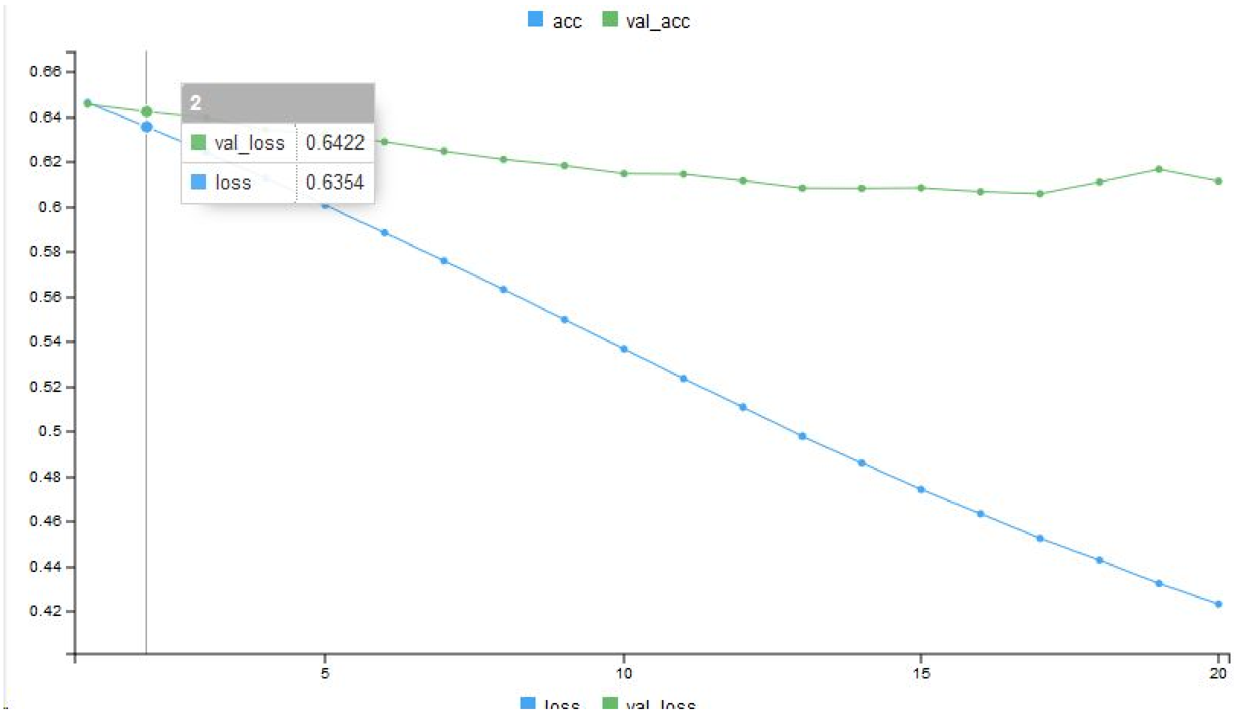
	`` validation_data = list(x_val, y_val)) results3 <- model3 %>% evaluate(x_test, y_test) ```	
4.	``` #model 4 library(keras) model4 <- keras_model_sequential() %>% layer_dense(units = 16, activation = "tanh", input_shape = c(10000)) %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 1, activation = "sigmoid") model4 %>% compile(optimizer = "rmsprop",loss = "poisson", metrics = c("accuracy")) model4 %>% fit(x_train, y_train, epochs = 10,batch_size = 512) #plot of history based on validation data history <- model4 %>% fit(partial_x_train, partial_y_train, epochs = 20, batch_size = 512, validation_data = list(x_val, y_val)) results4 <- model4 %>% evaluate(x_test, y_test) ```	Loss : 0.9333 Accuracy : 0.6505 Link to Google Cloud Output
5.	``` #model 5 model5 <- keras_model_sequential() %>% layer_dense(units = 16, activation = "relu", input_shape = c(10000)) %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 16, activation = "tanh") %>% layer_dense(units = 1, activation = "sigmoid") model5 %>% compile(optimizer = "rmsprop", loss = "poisson", metrics = c("binary_accuracy")) model5 %>% fit(x_train, y_train, epochs = 10,batch_size = 512) ```	Loss : 0.9417 Accuracy : 0.6536 Link to Google Cloud Output

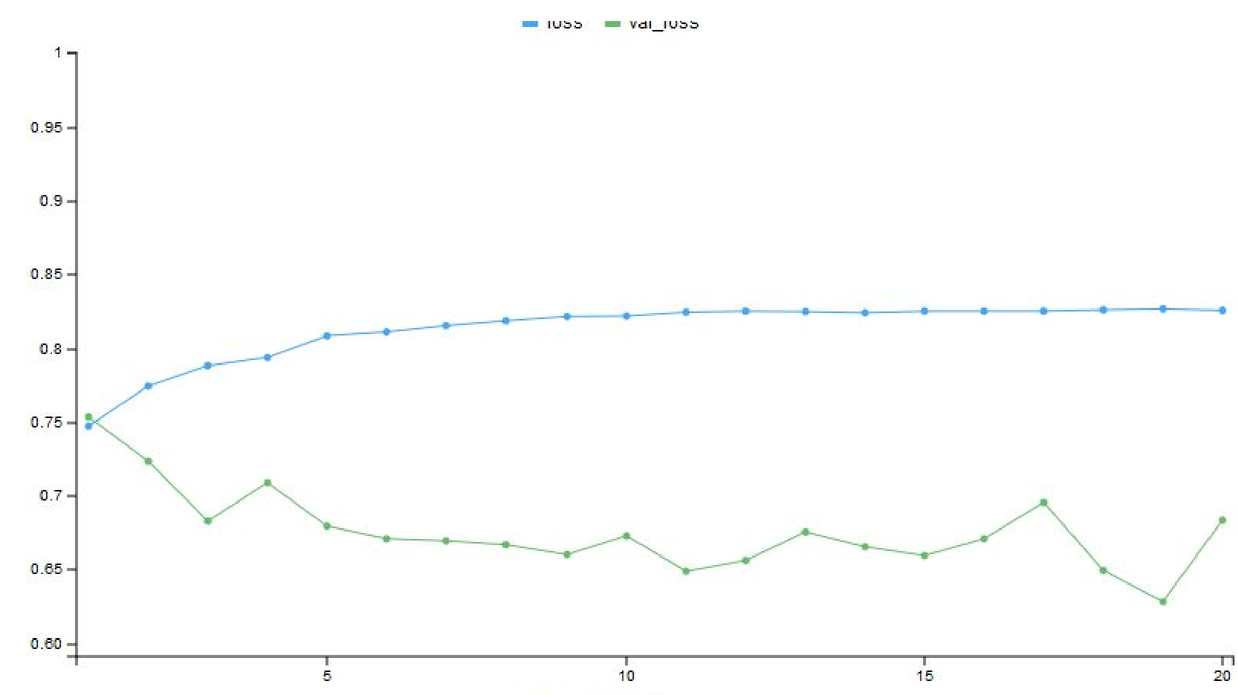
```
#plot of history based on validation data
history <- model5 %>% fit(
  partial_x_train,
  partial_y_train,
  epochs = 20,
  batch_size = 512,
  validation_data = list(x_val, y_val)
)

results5 <- model5 %>% evaluate(x_test, y_test)
```

Performance Plots

We have just shown the plots for a single model. The rest of the plots and detailed outputs can be seen by going to the links given above.




acc
val_acc
2
val_loss 0.6422
loss 0.6354
0.66
0.64
0.62
0.6
0.58
0.56
0.54
0.52
0.5
0.48
0.46
0.44
0.42
5
10
15
20


1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.60
5
10
15
20

CloudML	
job	cloudml_2019_03_10_012259742
logs	View logs
status	SUCCEEDED
created	2019-03-10 01:23:36 GMT
time	00:08:57
ml_units	0.09

Run	
context	cloudml
script	TextClassification.R
started	2019-03-10 01:27:33 GMT
time	00:00:23

Metrics	
loss	0.4233
acc	0.8255
val_loss	0.6113
val_acc	0.6833

Evaluation	
eval_loss	0.6367
eval_acc	0.6568

Optimization	
loss	binary_crossentrop
optimizer	<tensorflow.python.keras.optimizers.RMSprop
lr	0.00

Training	
samples	2,728
validation_samples	1,500
epochs	20
batch_size	512

Conclusion

We have performed the training, testing and validation on 5 different deep Learning models and have observed the accuracy and loss for each these models. The different models were created by changing different parameters like the activation function, the loss function, the optimizer and by varying the number of hidden layers and the number of units in each of them.