Deep Learning for Text Analytics

SAS User Group Malaysia

12th April 2018

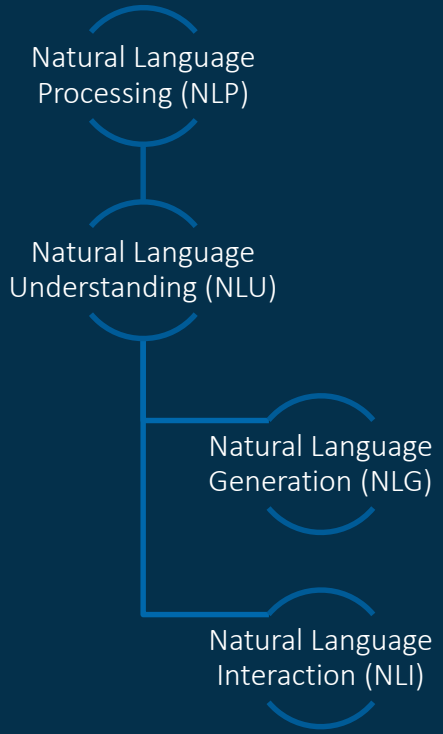
THE POWER TO KNOW.

Using Deep Learning in Natural Language

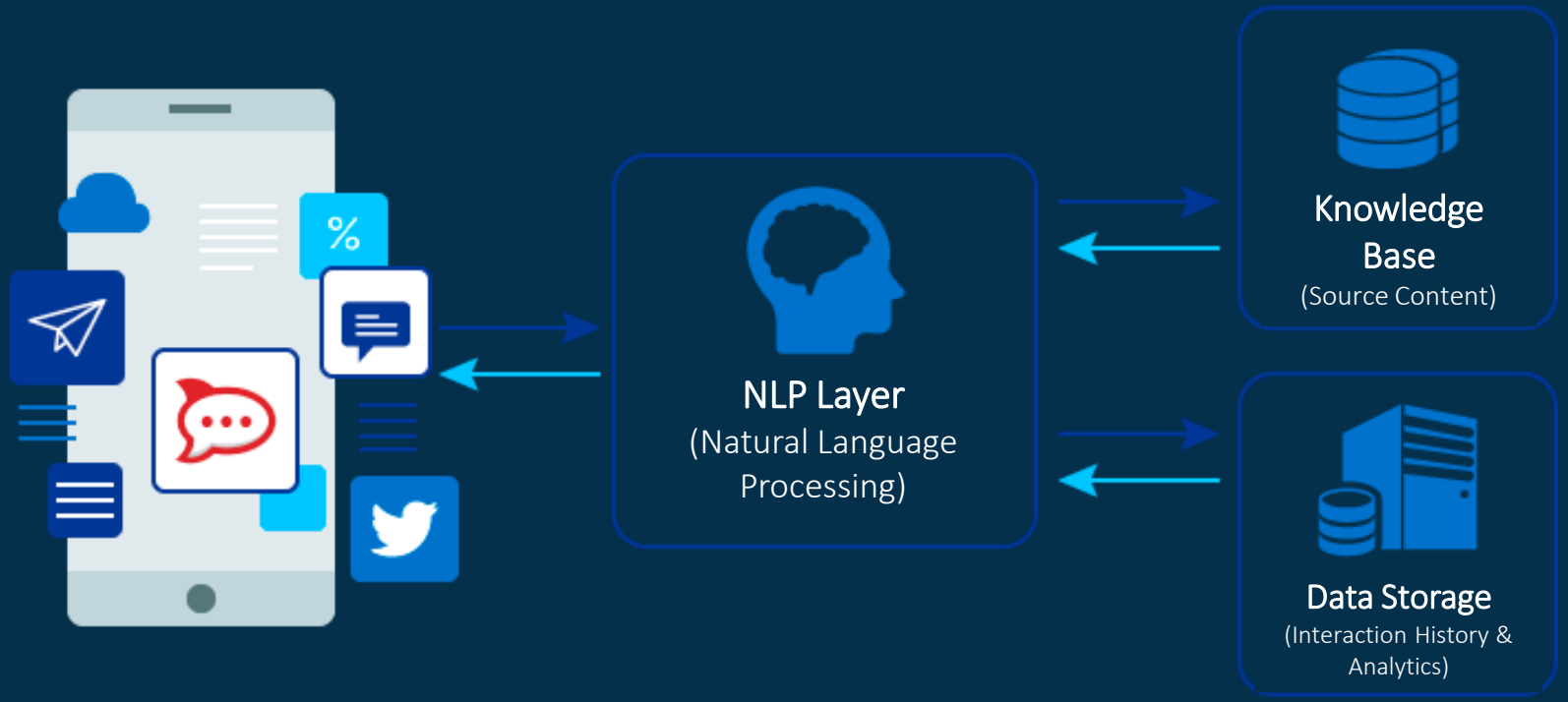
SAS in a Chatbot

Natural Language

Interaction



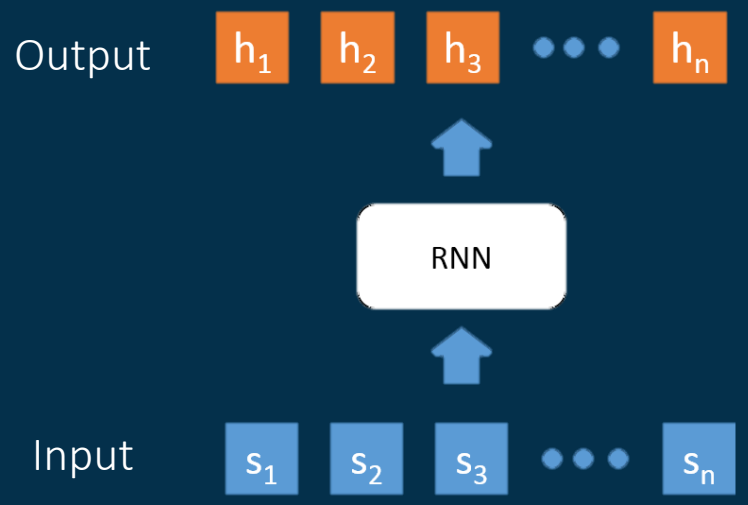
Natural Language Processing



Recurrent Neural Network

Deep Learning

Recurrent Neural Network (RNN)



- Type of Neural Network
  - Recurring over time
- Sequential data
  - Words
  - Time
- Common Methods
  - GRU (Gated Recurrent Unit)
  - LSTM (Long Short Term Memory)

Recurrent Neural Network (RNN)

Word Vector

Unlabeled Corpus

The 15th American President

The 16th American President

The 17th American President

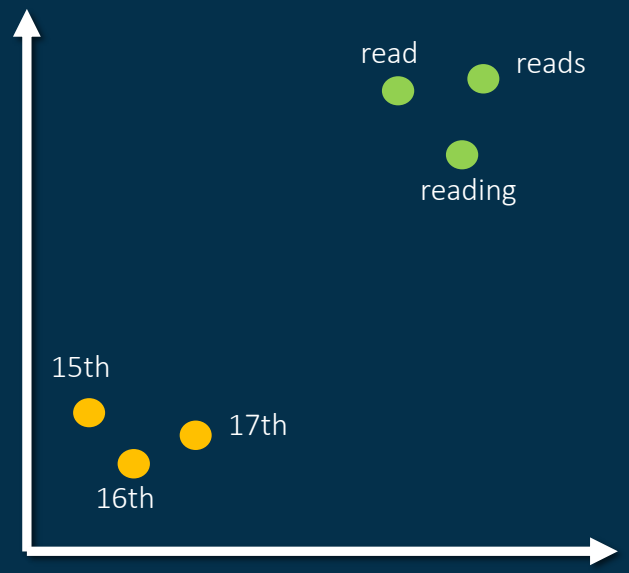
Alex reads this sentence

Alex read this sentence

Alex is reading this sentence

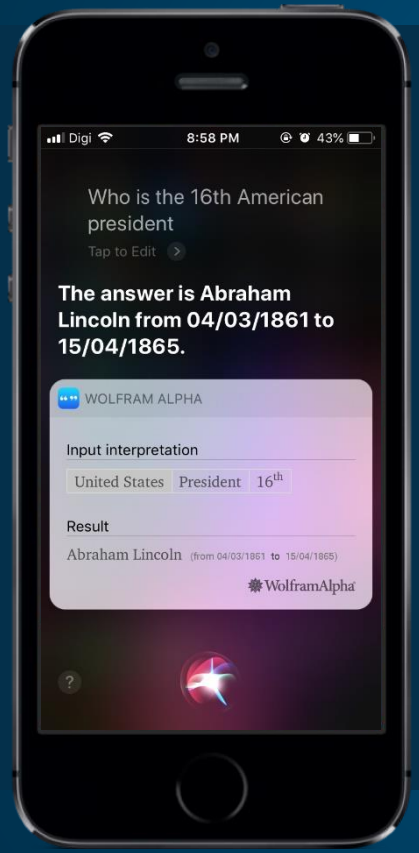
Word Vector Algorithm

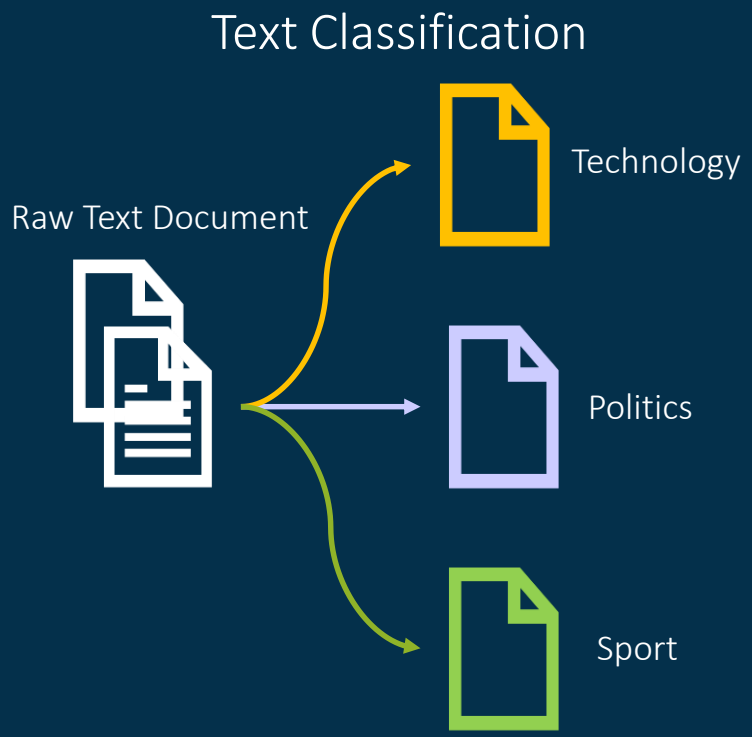
Words with similar context should have similar vectors



Recurrent Neural Network (RNN)

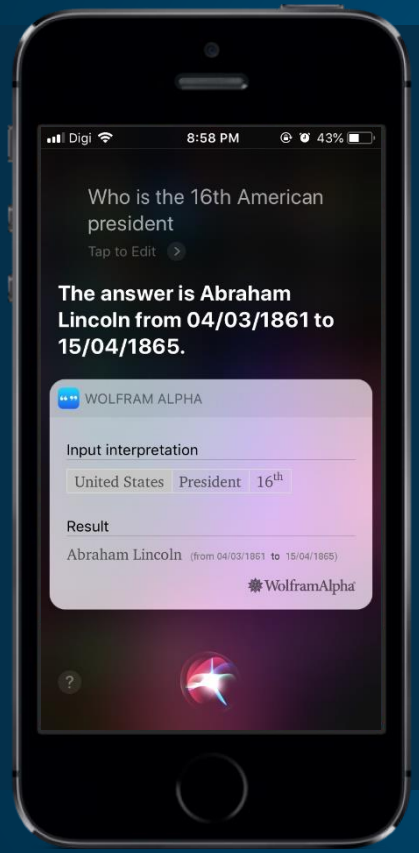
Text Classification





Recurrent Neural Network (RNN)

Text Classification



Text Classification

The 16th American President

number

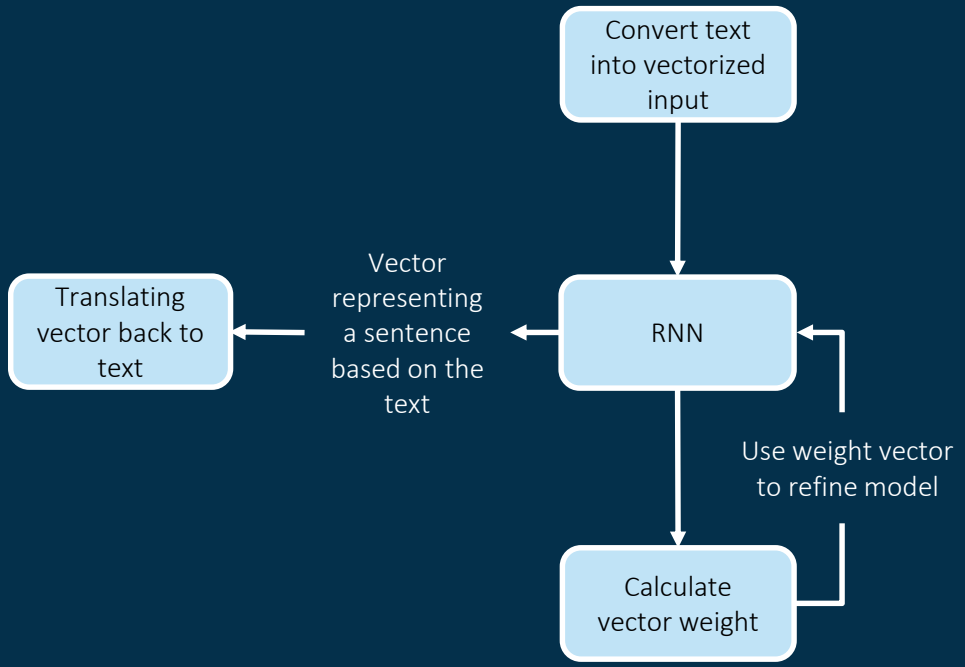
order

entity

context

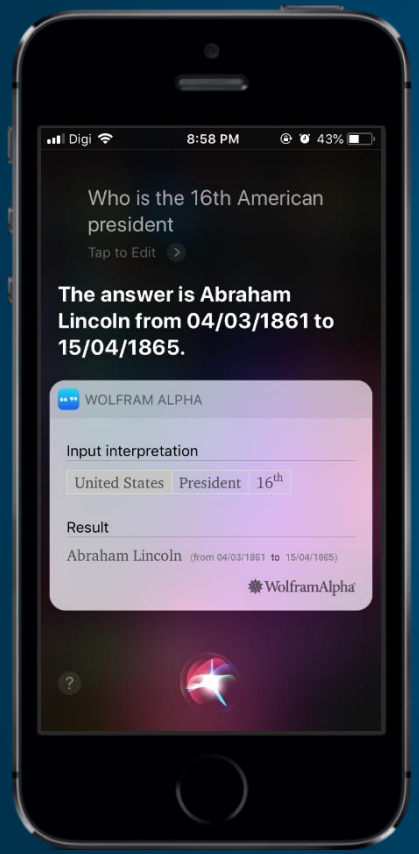
Recurrent Neural Network (RNN)

Text Generation



Text Generation

Text Structure – Word Order



Who is the 16th American President

≠

The 16th President who is American

Text Structure

Word Order

Two random corpus

- I don't like this director but I like this movie *Positive sentiment*
- I like this director but I don't like this movie *Negative sentiment*

- Specific words can be strong indicators
  - Sentiment – boring, exciting
  - Topic – deep learning, Siri

Creating, Training, Scoring an RNN

Using Deep Learning

Sample RNN Model

Loading the Action Sets

Connect to the Server and Load the Action Sets

In [2]:

```
cashost='localhost'
casport=5570
casauth='~/.authinfo'

s = CAS(cashost, casport, 'sasdemo1', 'Orion123', protocol='cas')
s.sessionprop.setsessopt(caslib='casuser')

s.loadactionset('deepLearn')
s.loadactionset('castmine')
s.loadactionset('fedsql')
```

```
NOTE: 'CASUSER(sasdemo1)' is now the active caslib.
NOTE: Added action set 'deepLearn'.
NOTE: Added action set 'castmine'.
NOTE: Added action set 'fedsql'.
```

Out[2]:

§ actionset

fedsql

elapsed 0.00196s · user 0.00194s · mem 0.23MB

Sample RNN Model

The Dataset

Upload Dataset

In [3]:
```
s.upload('/opt/sasinside/DemoData/glove_100d_tab_clean.txt', casout=dict(name='glove', replace=False),
         importoptions=dict(fileType='delimited', delimiter='\t'))

s.upload('/opt/sasinside/DemoData/reviews_sample.csv', casout=dict(name='reviews_train', replace=True),
          importoptions=dict(fileType='csv', varChars=True, getNames=True))
s.upload('/opt/sasinside/DemoData/reviews_sample.csv', casout=dict(name='reviews_test', replace=True),
          importoptions=dict(fileType='csv', varChars=True, getNames=True))
s.upload('/opt/sasinside/DemoData/reviews_sample.csv', casout=dict(name='reviews_dev', replace=True),
          importoptions=dict(fileType='csv', varChars=True, getNames=True))
```

```
NOTE: Cloud Analytic Services made the uploaded file available as table GLOVE in caslib CASUSER(sasdemo1).
NOTE: The table GLOVE has been created in caslib CASUSER(sasdemo1) from binary data uploaded to Cloud Analytic Servic
es.
```

In [4]:
```
s.fetch(table='reviews_train', to=10, fetchVars=['title', 'review', 'positive'])
```

Out[4]: **§ Fetch**

Selected Rows from Table REVIEWS_TRAIN

	title	review	positive
0	An all time classic band & release	As others here have commented on,this incredib...	1.0
1	Loved It!!!!!!!!!!!!!!!!!!!!	So glad it finally came out!!!!!!!!!!!!!!!!!!...	1.0
2	You won't be disappointed	I have read everything Laura Lippman has writt...	1.0
3	Beautiful as always	And he did it again!I'm a real fan of Gaiman a...	1.0
4	So Funny and yet I learned a lot	I'm almost done reading this great book and ha...	1.0
5	Very nice product and delivery service!	Awsome! I just concerned about the high temper...	1.0
6	Value for money tablet	A value for money piece of tablet made by ASUS...	1.0
7	Great book.	I think I've already left a review for this bo...	1.0
8	Dog Grooming for Dummies	This was a gift to go along with the dog cutte...	1.0
9	I don't use this to keep coffee/tea warm	I use this to keep my wax warm for in-home wax...	1.0

elapsed 0.00222s · user 0.00212s · sys 1.4e-05s · mem 0.878MB

Sample RNN Model

Text Classification Model

Text Classification Model

In [5]:

```
# text classification where it predicts for the target:

n=50
init='normal'

s.buildmodel(model=dict(name='sentiment', replace=True), type='RNN')
s.addlayer(model='sentiment', name='data', layer=dict(type='input'))
s.addlayer(model='sentiment', name='rnn1', srclayers=['data'],
           layer=dict(type='recurrent',n=n,init=init, rnnType='GRU',outputType='samelength', reversed=True))
s.addlayer(model='sentiment', name='rnn2', srclayers=['data'],
           layer=dict(type='recurrent',n=n,init=init,rnnType='GRU',outputType='samelength', reversed=False))
s.addlayer(model='sentiment', name='rnn3', srclayers=['rnn1', 'rnn2'],
           layer=dict(type='recurrent',n=n,init=init,rnnType='GRU',outputType='encoding', act='sigmoid'))
s.addlayer(model='sentiment', name='outlayer', srclayers=['rnn3'],
           layer=dict(type='output'))
```

Out[5]: **§ OutputCasTables**

	casLib	Name	Rows	Columns	casTable
0	CASUSER(sasdemo1)	sentiment	62	5	CASTable('sentiment', caslib='CASUSER(sasdemo1)')

elapsed 0.0382s · user 0.116s · sys 0.0071s · mem 3.78MB

Sample RNN Model

Training the Model

Out[6]:

§ ModelInfo

	Descr	Value
0	Model Name	sentiment
1	Model Type	Recurrent Neural Network
2	Number of Layers	5
3	Number of Input Layers	1
4	Number of Output Layers	1
5	Number of Convolutional Layers	0
6	Number of Pooling Layers	0
7	Number of Fully Connected Layers	0
8	Number of Recurrent Layers	3
9	Number of Weight Parameters	67600
10	Number of Bias Parameters	452
11	Total Number of Model Parameters	68052
12	Approximate Memory Cost for Training (MB)	1127

§ OptIterHistory

	Epoch	LearningRate	Loss	FitError	ValidLoss	ValidError
0	0.0	0.001	0.335889	0.105688	0.329935	0.10564

§ OutputCasTables

	casLib	Name	Rows	Columns	casTable
0	CASUSER(sasdemo1)	trainedWeights	68052	3	CASTable('trainedWeights', caslib='CASUSER(sas...

elapsed 655s · user 6.38e+03s · sys 2.1e+03s · mem 1.02e+04MB

Sample RNN Model

Scoring the Model

Scoring the Model

In [8]:
```
s.dlScore(table='reviews_test', model='sentiment', initWeights='trainedWeights', copyVars=['review', 'positive'],
                textParms=dict(initInputEmbeddings='glove'), casout=dict(name='sentiment_out', replace=True))
```

Out[8]:

§ ScoreInfo

	Descr	Value
0	Number of Observations Read	947509
1	Number of Observations Used	947509
2	Misclassification Error (%)	10.56402
3	Loss Error	0.32997

§ OutputCasTables

	casLib	Name	Rows	Columns	casTable
0	CASUSER(sasdemo1)	sentiment_out	947509	7	CASTable('sentiment_out', caslib='CASUSER(sasd...

elapsed 199s · user 2.26e+03s · sys 910s · mem 5.17e+03MB

Sample RNN Model

Scoring the Model

Taking Word Order Into Account

In [10]:

```
review1 = "I don't like the author of this book , however I like this book" # positive review
print(review1)
print(score_one_doc(review1))
print('\n\n')
review2 = "I like the author of this book , however I don't like this book" # negative review
print(review2)
print(score_one_doc(review2))
print('\n\n\n\n\n')
```

```
I don't like the author of this book , however I like this book
Selected Rows from Table SENTIMENT_ONE_DOC

   _DL_PredName_  _DL_PredP_
0              1    0.950873



I like the author of this book , however I don't like this book
Selected Rows from Table SENTIMENT_ONE_DOC

   _DL_PredName_  _DL_PredP_
0              1    0.813228
```

Sample RNN Model

Text Generation Model

Building the Text Generation Model

In [9]:

```
#target is var so that is for text generation;
#var2 is a text variable (explicitly state to treat it as text otherwies it'll be text variable)

n=50
init='normal'

s.buildmodel(model=dict(name='title', replace=True), type='RNN')
s.addlayer(model='title', name='data', layer=dict(type='input'))
s.addlayer(model='title', name='rnn1', srclayers=['data'],
           layer=dict(type='recurrent',n=n, init=init, rnnType='GRU',outputType='encoding', reversed=True))
s.addlayer(model='title', name='rnn2', srclayers=['data'],
           layer=dict(type='recurrent',n=n, init=init, rnnType='GRU',outputType='encoding', reversed=False))
s.addlayer(model='title', name='rnn3', srclayers=['rnn1', 'rnn2'],
           layer=dict(type='recurrent',n=n*2, init=init, rnnType='GRU',outputType='arbitrarylength',
                      act='sigmoid', maxoutputlength=32))
s.addlayer(model='title', name='outlayer', srclayers=['rnn3'],
           layer=dict(type='output', act='softmax'))
```

Out[9]: **§ OutputCasTables**

	casLib	Name	Rows	Columns	casTable
0	CASUSER(sasdemo1)	title	62	5	CASTable('title', caslib='CASUSER(sasdemo1)')

elapsed 0.0221s · user 0.0495s · sys 0.00622s · mem 3.77MB

Sample RNN Model

Training the Model

```
In [8]: s.dlTrain(table='reviews_dev', model='title',
                  modelWeights=dict(name='trainedWeights', replace=True),
                  textParms=dict(initEmbeddings='glove', hasInputTermIds=False,
                                 modelOutputEmbeddings=dict(name='trainedGlove', replace=True)),
                  target='var2', inputs=['var3'], texts=['var2', 'var3'],
                  optimizer=dict(miniBatchSize=16, maxEpochs=1, loglevel=3,
                                 algorithm=dict(method='adam', beta1=0.9, beta2=0.999, gamma=0.5, learningRate=0.01
                                                , stepSize=20, lrPolicy='step',clipGradMin=-10, clipGradMax=10,
                                               )
                                ),
                  seed=12345
                )
```

```
NOTE:  Synchronous SGD is starting.
NOTE:  The total number of parameters is 3822440.
NOTE:  The approximate memory cost is 19739.00 MB.
NOTE:  Loading weights cost       0.00 (s).
NOTE:  Initializing each layer cost     147.37 (s).
NOTE:  The total number of threads on each worker is 8.
NOTE:  The total number of minibatch size per thread on each worker is 16.
NOTE:  The maximum number of minibatch size across all workers for the synchronous mode is 128.
NOTE:  Target variable: Var2
NOTE:  Number of input variables:     1
NOTE:  Number of numeric input variables:      2
NOTE:  Batch              nUsed   Learning Rate     Loss    Fit Error      Time (s) (Training)
NOTE:        0              128    0.01          11.679          1          0.79
NOTE:        1              128    0.01          11.414          1          0.77
NOTE:        2              128    0.01            11.2          1          1.39
NOTE:        3              128    0.01          11.093          1          0.72
NOTE:        4              128    0.01          10.858     0.9983          0.85
NOTE:        5              128    0.01           10.64     0.9835          1.18
NOTE:        6              128    0.01           10.46     0.9835          0.91
```

Sample RNN Model

Scoring the Model

Scoring

In [27]:
```
# misclassification error for text generation can be misleading since it requires exact matc
s.dlscore(table='reviews_test', model='title', initWeights='title_trainedWeights',
          textParms=dict(initInputEmbeddings='glove', initOutputEmbeddings='glove_ids', hasO
          copyvars=['id', 'title', 'review', 'positive'], casout=dict(name='title_out', repl
df = s.fetch(table=dict(name='title_out'), to=100)['Fetch']
```

Out[27]:

§ ScoreInfo

	Descr	Value
0	Number of Observations Read	9635
1	Number of Observations Used	9635
2	Misclassification Error (%)	90.71156
3	Loss Error	9.162203

§ OutputCasTables

Sample RNN Model

Text Generation Output

Bad Generated Titles

In [97]:
```
row = df.loc[df['id']==13310].iloc[0]
print('review:\t\t' + row['review'])
print('ground truth:\t' + row['title'])
print('prediction:\t' + row['_DL_Pred_'])
```

```
review:         i really like this app! very easy to use and the audio is grea
t. it is nice to see the spreading of God's word is still Free for some people.
a+++
ground truth:   awesome!
prediction:     very app
```

In [4]:
```
s.endsession()
```

Out[4]:

elapsed 0.00826s · user 0.001s · mem 0.229MB

Useful Links

- What's New In SAS Deep Learning (Documentation)

http://go.documentation.sas.com/?docsetId=casdlpg&docsetTarget=n0gv3jjm5obouun1uvducbzl8nlf.htm&docsetVersion=8.2&locale=en

- Understanding Recurrent Neural Networks

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- RNN Simplified

https://www.youtube.com/watch?v=_aCuOwF1ZjU

Thank You

sas.com

THE POWER TO KNOW.