Deep Learning for Text Analytics

SAS User Group Malaysia

12th April 2018
Using Deep Learning in Natural Language
SAS in a Chatbot
Natural Language Interaction

- Natural Language Processing (NLP)
- Natural Language Understanding (NLU)
- Natural Language Generation (NLG)
- Natural Language Interaction (NLI)
Natural Language Processing

NLP Layer
(Natural Language Processing)

Knowledge Base
(Source Content)

Data Storage
(Interaction History & Analytics)
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

The 16th American President

number
order
entity
context
Recurrent Neural Network (RNN)

Text Generation

1. Convert text into vectorized input
2. RNN
3. Use weight vector to refine model
4. Calculate vector weight
5. Translating vector back to text

Vector representing a sentence based on the text
Who is the 16th American President

The 16th President who is American
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
Connect to the Server and Load the Action Sets

In [2]:
1. cashost='localhost'
2. casport=5570
3. casauth='~/.authinfo'
4. s = CAS(cashost, casport, 'sasdemol', 'Orion123', protocol='cas')
5. s.sessionprop.setsessopt(caslbin='casuser')
6. s.loadactionset('deepLearn')
7. s.loadactionset('castmine')
8. s.loadactionset('fedsql')

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

Out[2]:

```python
fedsql
```

elapsed 0.00198s · user 0.00194s · mem 0.23MB
Sample RNN Model

The Dataset

**Upload Dataset**

```r
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(sasdemo).
**NOTE:** The table GLOVE has been created in caslib CASUSER(sasdemo) from binary data uploaded to Cloud Analytic Services.

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

**Sheet 1:**

<table>
<thead>
<tr>
<th>title</th>
<th>review</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>An all time classic b &amp; release</td>
<td>As others here have commented on this incredible...</td>
<td>1.0</td>
</tr>
<tr>
<td>Loved</td>
<td>So glad it finally came out!!!!!!!!!!!!!!</td>
<td>1.0</td>
</tr>
<tr>
<td>You won't be disappointed</td>
<td>I have read everything Laura Lippman has writ...</td>
<td>1.0</td>
</tr>
<tr>
<td>Beautiful as always</td>
<td>And he did again I'm a real fan of Gaiman a...</td>
<td>1.0</td>
</tr>
<tr>
<td>Very Nice product and delivery service</td>
<td>I'm almost done reading this great book and ha...</td>
<td>1.0</td>
</tr>
<tr>
<td>Value for money tablet</td>
<td>A value for money piece of tablet made by ASUS...</td>
<td>1.0</td>
</tr>
<tr>
<td>Great book</td>
<td>I think I've already left a review for this bo...</td>
<td>1.0</td>
</tr>
<tr>
<td>Dog Grooming for Dummies</td>
<td>This was a gift to go along with the dog colla...</td>
<td>1.0</td>
</tr>
<tr>
<td>I don't use this to keep coffee/tea warm</td>
<td>I use this to keep my wax warm for in-home wax...</td>
<td>1.0</td>
</tr>
</tbody>
</table>
```

elapsed 0.00222s · user 0.00212s · sys 1.4e-05s · mem 0.070MB
Text Classification Model

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

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', srlayers=['data'],
layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='samelen', reversed=True))
s.addlayer(model='sentiment', name='rnn2', srlayers=['rnn1'],
layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='samelen', reversed=False))
s.addlayer(model='sentiment', name='rnn3', srlayers=['rnn1', 'rnn2'],
layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='encoding', act='sigmoid'))
s.addlayer(model='sentiment', name='output', srlayers=['rnn3'],
layer=dict(type='output'))
```

Out[5]:

<table>
<thead>
<tr>
<th>casLib</th>
<th>Name</th>
<th>Rows</th>
<th>Columns</th>
<th>casTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASUSER(sasdemo1)</td>
<td>sentiment</td>
<td>62</td>
<td>5</td>
<td>CASTable('sentiment', caslib='CASUSER(sasdemo1)')</td>
</tr>
</tbody>
</table>

elapsed 0.0382s · user 0.115s · sys 0.0071s · mem 3.78MB
Sample RNN Model

Training the Model

Out[6] =

<table>
<thead>
<tr>
<th>Descr</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Model Name</td>
</tr>
<tr>
<td>1</td>
<td>Model Type</td>
</tr>
<tr>
<td>2</td>
<td>Number of Layers</td>
</tr>
<tr>
<td>3</td>
<td>Number of Input Layers</td>
</tr>
<tr>
<td>4</td>
<td>Number of Output Layers</td>
</tr>
<tr>
<td>5</td>
<td>Number of Convolutional Layers</td>
</tr>
<tr>
<td>6</td>
<td>Number of Pooling Layers</td>
</tr>
<tr>
<td>7</td>
<td>Number of Fully Connected Layers</td>
</tr>
<tr>
<td>8</td>
<td>Number of Recurrent Layers</td>
</tr>
<tr>
<td>9</td>
<td>Number of Weight Parameters</td>
</tr>
<tr>
<td>10</td>
<td>Number of Bias Parameters</td>
</tr>
<tr>
<td>11</td>
<td>Total Number of Model Parameters</td>
</tr>
<tr>
<td>12</td>
<td>Approximate Memory Cost for Training (MB)</td>
</tr>
</tbody>
</table>

OptitersHistory

<table>
<thead>
<tr>
<th>Epoch</th>
<th>LearningRate</th>
<th>Loss</th>
<th>FitError</th>
<th>ValidLoss</th>
<th>ValidError</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.091</td>
<td>0.335889</td>
<td>0.105688</td>
<td>0.329935</td>
</tr>
</tbody>
</table>

OutputCasTables

<table>
<thead>
<tr>
<th>cast.lib</th>
<th>Name</th>
<th>Rows</th>
<th>Columns</th>
<th>casTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASUSER</td>
<td>trainedWeights</td>
<td>68052</td>
<td></td>
<td>CASTable&quot;trainedWeights&quot;, castlib=CASUSER</td>
</tr>
</tbody>
</table>

elapsed 655s · user 6.35e+03s · sys 2.1e+03 · mem 1.62e+04MB
Scoring the Model

```r
sdlScore(table='reviews_test', model='sentiment', initWeights='trainedWeights', copyVars=['review', 'positive'], textFrm=dic(t(initInputEmbeddings='glove'), casout='dic(name='sentiment_out', replace=True))
```

<table>
<thead>
<tr>
<th>Descr</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
<td>947509</td>
</tr>
<tr>
<td>Number of Observations Used</td>
<td>947509</td>
</tr>
<tr>
<td>Misclassification Error (%)</td>
<td>0.56402</td>
</tr>
<tr>
<td>Loss Error</td>
<td>0.32997</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>castLib</th>
<th>Name</th>
<th>Rows</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASTUSER(sasdemo1)</td>
<td>sentiment_out</td>
<td>947509</td>
<td></td>
</tr>
</tbody>
</table>

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

Scoring the Model

Taking Word Order Into Account

```python
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('

')
    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('

')
```

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

```
_DL_PredName_  _DL_PredF_
0    1  0.850873
```

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

```
_DL_PredName_  _DL_PredF_
0    1  0.813228
```
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 otherwise 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', aroclayers=['data'],
              layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='encoding', reversed=True))
   s.addlayer(model='title', name='rnn2', aroclayers=['data'],
              layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='encoding', reversed=False))
   s.addlayer(model='title', name='rnn3', aroclayers=['rnn1', 'rnn2'],
              layer=dict(type='recurrent', n=n, init=init, rnnType='GRU', outputType='arbitrarylength',
                         act='sigmoid', maxoutputlength=32))
   s.addlayer(model='title', name='output', aroclayers=['rnn3'],
              layer=dict(type='output', act='softmax'))

Out[9]:
   § OutputCasTables

<table>
<thead>
<tr>
<th></th>
<th>casLib</th>
<th>Name</th>
<th>Rows</th>
<th>Columns</th>
<th>casTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CASUSER(sasdemo1)</td>
<td>title</td>
<td>62</td>
<td>5</td>
<td>CASTable(title, caslib='CASUSER(sasdemo1)')</td>
</tr>
</tbody>
</table>

eelapsed 0.0221s · user 0.0495s · sys 0.00622s · mem 3.77MB
Sample RNN Model

Training the Model

```python
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,
                                           batchSize=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

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

```
Out[27]:

<table>
<thead>
<tr>
<th>Descr</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
<td>9635</td>
</tr>
<tr>
<td>Number of Observations Used</td>
<td>9635</td>
</tr>
<tr>
<td>Misclassification Error (%)</td>
<td>90.71156</td>
</tr>
<tr>
<td>Loss Error</td>
<td>9.162203</td>
</tr>
</tbody>
</table>
```
Sample RNN Model

Text Generation Output

```
In [97]:
      row = df.loc[df['id'] == 13310].iloc[0]
      print('review:	' + row['review'])
      print('ground truth:	' + row['title'])
      print('prediction:	' + row['_DL_Pred_'])

  review:  i really like this app! very easy to use and the audio is great.
            it is nice to see the spreading of God's word is still free for some people.
  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=n0gv3jm5obouun1uvducbzl8nlf.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