Deep Neural Networks for Text Classification

Aniruddha Shamasundar(ashamas)
Chinmai Kaidabettu Srinivas(ckaidab)
Siddu Madhure Jayanna(smadhur)

Problem Statement

- Develop, compare and analyse performance of 3 different Deep Neural Networks for text classification
    1. Convolutional Neural Network (CNN)
    2. Recurrent Neural Network (RNN)
    3. Hierarchical Attention Network (HAN)

- Use word vectors generated by Google’s GloVe as an underlying data model to get the vector embeddings for our data set

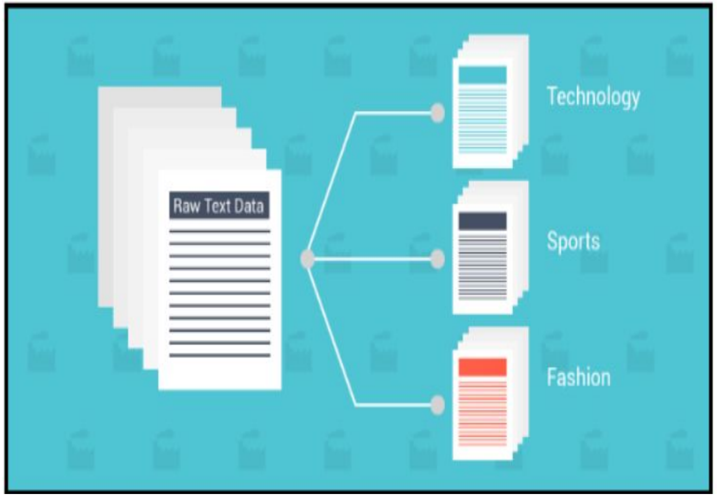
Dataset

News Category Dataset

- Consists of 200k news headlines from the year 2012 to 2018 obtained from HuffPost.
- Consists of 41 different news categories.

Sample Dataset

{
"category": "CRIME",
"headline": "There Were 2 Mass Shootings In Texas Last Week, But Only 1 On TV",
"authors": "Melissa Jeltsen",
"link":"https://www.huffingtonpost.com/entry/texas-amanda-painter-mass-shooting_us_5b081ab4e4b0802d69caad89",
"short_description": "She left her husband. He killed their children. Just another day in America.",
"date": "2018-05-26"
}



Dataset preprocessing- 1

1. Reduced 41 categories to 20 news categories by:
    a. Merging related categories to a single category
2. Combined short description along with news headline to give descriptive news headline
3. Sequence encoding
    - Formatted our text samples into number sequences to feed it to neural network
    - Utilize
        - keras.preprocessing.text.Tokenizer
        - Keras.preprocessing.sequence.pad_sequences

Dataset preprocessing- 2

4) Creation of Embedding Layer using GloVe
    - Compute an index mapping of words to vector embeddings
    - Use the above vector embeddings to create an embedding layer

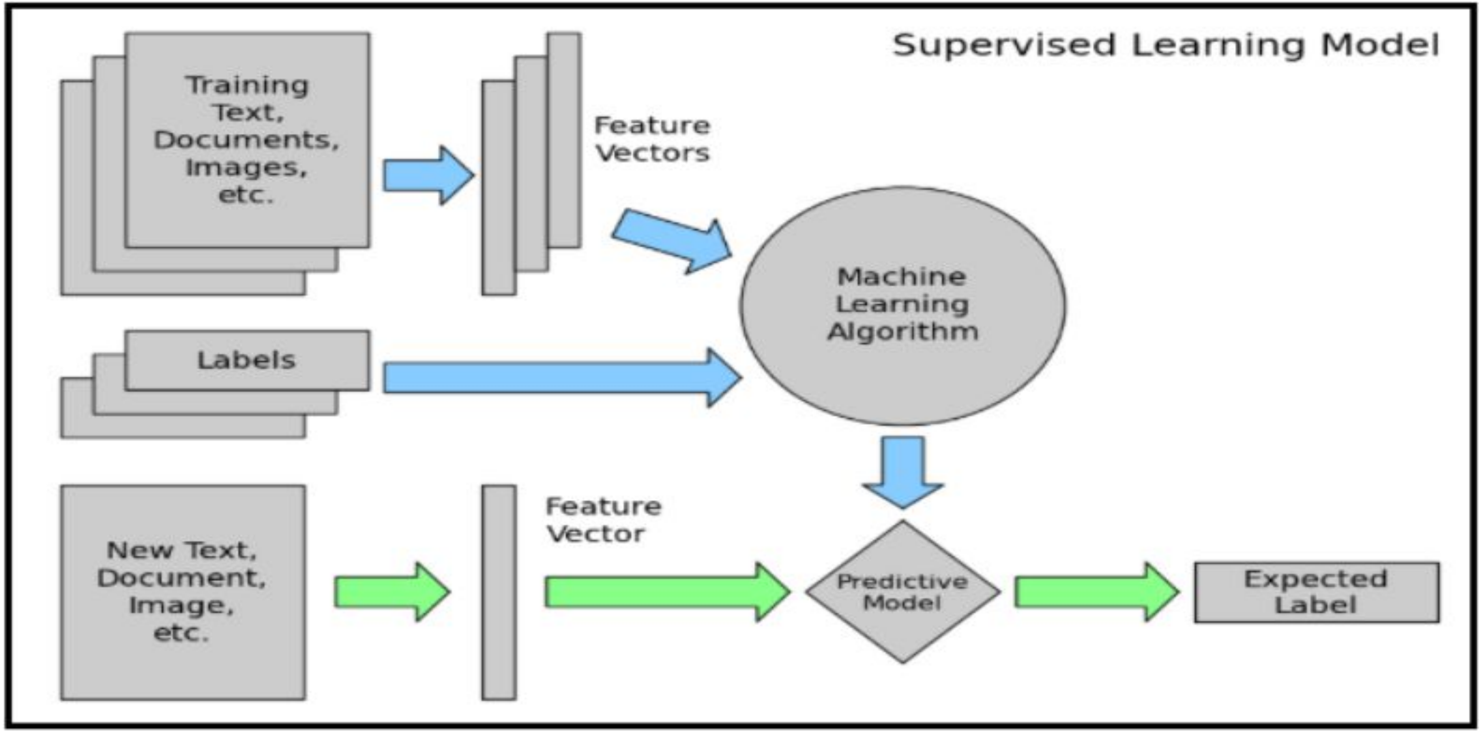
5) Convert each of the 20 categories into a vector of 20-dimensions
    - Use one hot encoding technique.

DNN models

We have built and analyzed the following deep neural networks

1. Convolutional Neural Networks (CNN)
2. Recurrent Neural Networks (RNN)
3. Hierarchical Attention Networks (HAN)

Flow Diagram



Convolutional Neural Networks

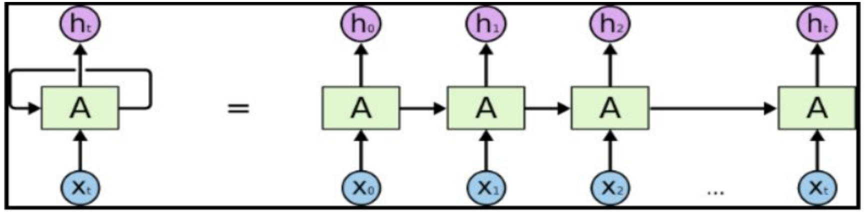
- Convolutional layer is used to identify special pattern in text data
    - Use different sized kernels to identify patterns of different sizes
    - Identifies patterns regardless of the position of words
- Use Max Pooling Layer to reduce the spatial size of the data representation
- Use Dropout layer to avoid overfitting
- Use Dense layer with softmax activation function to classify the news headline

CNN Model Summary

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 29)	0	
embedding_1 (Embedding)	(None, 29, 100)	11440500	input_1[0][0]
conv1d_1 (Conv1D)	(None, 29, 256)	51456	embedding_1[0][0]
conv1d_2 (Conv1D)	(None, 29, 256)	77056	embedding_1[0][0]
conv1d_3 (Conv1D)	(None, 29, 256)	102656	embedding_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 9, 256)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 9, 256)	0	conv1d_2[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 9, 256)	0	conv1d_3[0][0]
dropout_1 (Dropout)	(None, 9, 256)	0	max_pooling1d_1[0][0]
dropout_2 (Dropout)	(None, 9, 256)	0	max_pooling1d_2[0][0]
dropout_3 (Dropout)	(None, 9, 256)	0	max_pooling1d_3[0][0]
concatenate_1 (Concatenate)	(None, 9, 768)	0	dropout_1[0][0] dropout_2[0][0] dropout_3[0][0]
flatten_1 (Flatten)	(None, 6912)	0	concatenate_1[0][0]
dropout_4 (Dropout)	(None, 6912)	0	flatten_1[0][0]
dense_1 (Dense)	(None, 20)	138260	dropout_4[0][0]

Recurrent Neural Networks

- Overcomes shortcoming of traditional NN in dealing with sequence data
  - Integrates lexical and semantic information
- Layers Involved:
  - Use Spatial Dropout layer to perform variational dropout in text models
  - Use LSTM layer to retain last output in RNN



  - Use Dense layer with softmax activation function for multi-class text classification

RNN Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 29, 100)           11440500
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 29, 100)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 200)               240800
_________________________________________________________________
dense_2 (Dense)              (None, 20)                4020
=================================================================
```

Hierarchical Attention Networks

- Main idea - words make sentences and sentences make documents
- Preprocess data and construct 3D Matrix in order to cater the needs of HAN architecture
    - First Dimension represents total number of documents.
    - Second Dimension represents number of sentences in a document
    - Third dimension represents number of words in a sentence
- Layers Involved:
    - Use bidirectional LSTM layer to incorporate contextual information
    - Use Time Distributed layer to apply a layer to every temporal slice of input
    - Use Attention layer to:
        - Apply attention mechanism at word level and sentence level
        - Enables to attend more and less important content during document representation
- Use Dense layer with softmax activation function for multi-class text classification.

HAN Model Summary

```
____________________________________________________________________
Layer (type)                 Output Shape              Param #
====================================================================
input_7 (InputLayer)         (None, 3, 29)             0
____________________________________________________________________
time_distributed_5 (TimeDist (None, 3, 200)            11842300
____________________________________________________________________
bidirectional_4 (Bidirection (None, 3, 300)            421200
____________________________________________________________________
time_distributed_6 (TimeDist (None, 3, 200)            60200
____________________________________________________________________
attention_with_context_4 (At (None, 200)               40400
____________________________________________________________________
dropout_6 (Dropout)          (None, 200)               0
____________________________________________________________________
dense_8 (Dense)              (None, 20)                4020
====================================================================
```

Tuning Hyperparameters

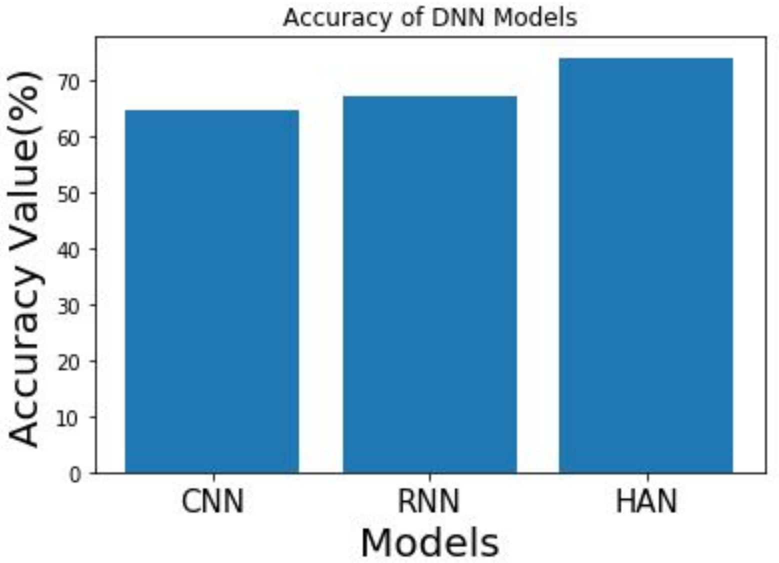
- Learning rate
- Batch size
- GloVe embedding dimensions
- Number of epochs
- Maximum number of words in a sentence for each model
- Number of layers in each model
- Number of filters for CNN
- Kernel size for max pooling in CNN
- Number of cells for LSTM layer in RNN
- Maximum number of sentences considered in HAN
- Dropout percentage

Results

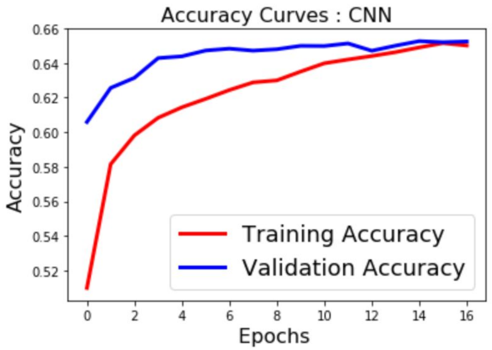
The accuracy rate of CNN: **64.75%**

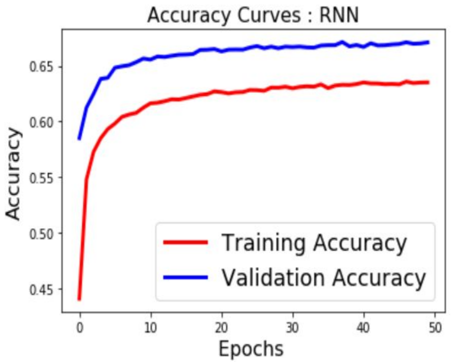
The accuracy rate of RNN: **67.01%**

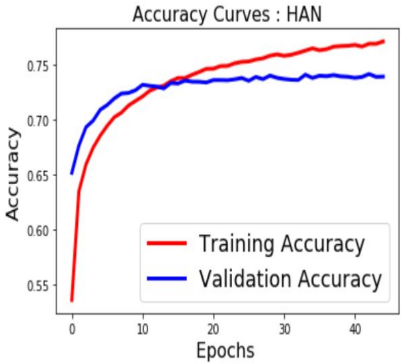
The accuracy rate of HAN: **74.05%**



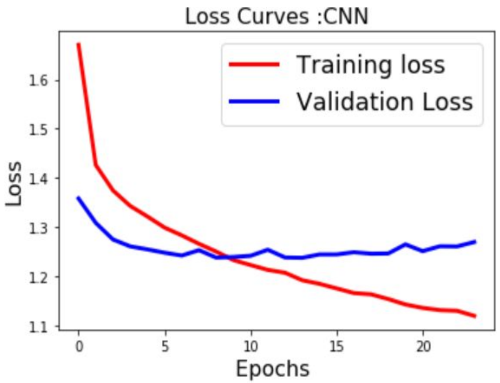
Analysis and Conclusion

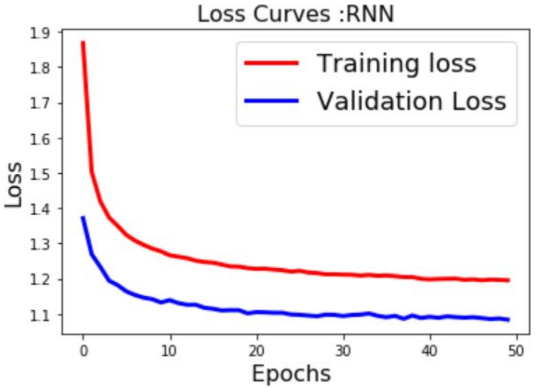


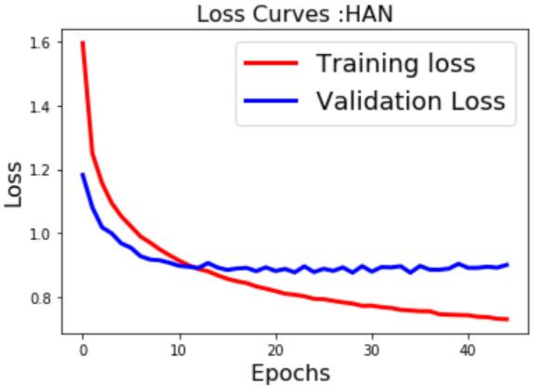




Contd..







Performance: HAN > RNN > CNN

However, CNN model has outperformed the other two models (RNN & HAN) in terms of training time.

References

https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf

https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

https://www.aclweb.org/anthology/W18-5408

https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17

https://medium.com/analytics-vidhya/hierarchical-attention-networks-d220318cf87e

Thank You