ASSIGNMENT No: 03

Title:-

Design RNN or its variant including LSTM or GRU

a) Select a suitable time series dataset. E.g - Predict sentiments based on product reviews.

b) Apply for prediction

Mapping with Syllabus -

Unit 4

Objective -

Implement a Recurrent Neural Network (RNN) or its variant (LSTM or GRU) on a selected time series dataset, such as predicting sentiments based on product reviews, to develop a predictive model for sentiment analysis.

Outcome -

Solve the language translation problem by Recurrent neural network(RNN)

Software Requirements -

- Python (3.x recommended)
- Jupyter Notebook or any Python IDE or Google Colab

Hardware Requirements -

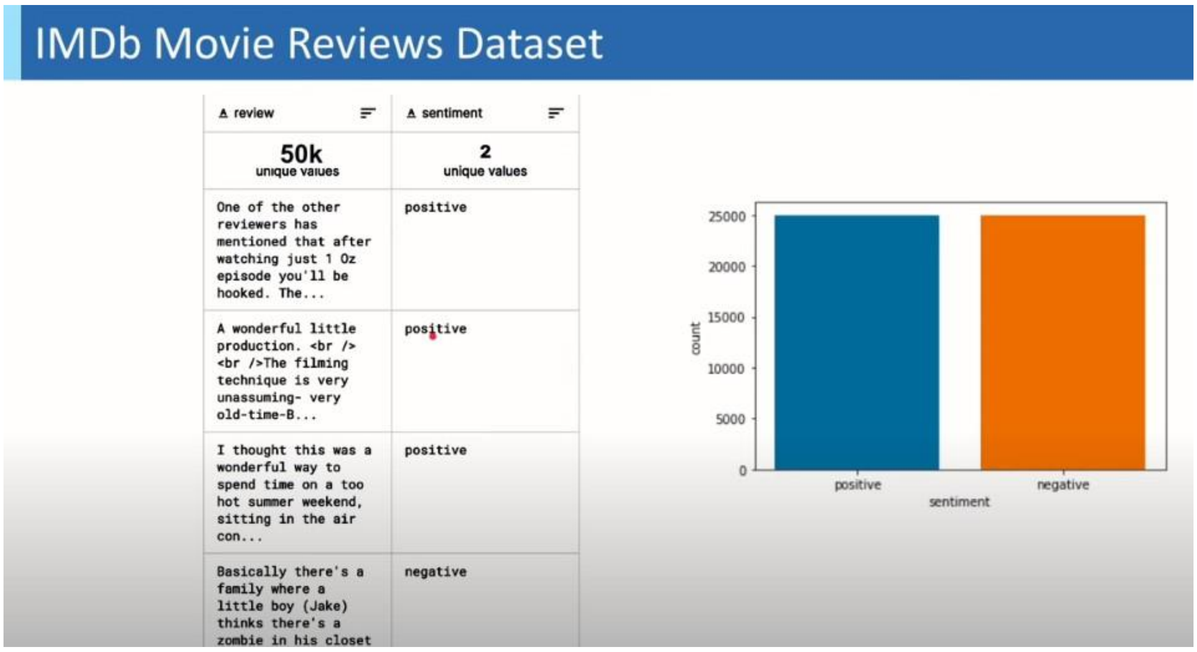
A machine with sufficient RAM and processing power for model training (8GB RAM recommended)

Prerequisites -

- Basic understanding of Python programming
- Familiarity with the concepts of Neural Networks

Dataset -

Inbuilt tensorflow-keras-imdb dataset



IMDb Movie Reviews Dataset

review	sentiment
50k unique values	2 unique values
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet	negative

-

Libraries or Modules Used

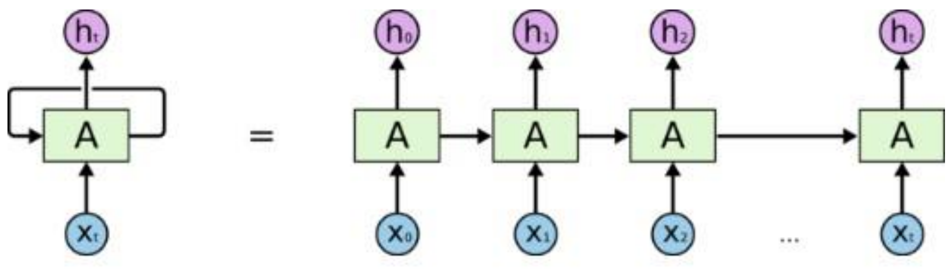
- Keras
- Tensorflow

Theory -

Recurrent Neural Network (RNN)

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.



An unrolled recurrent neural network.

First, it takes the X(0) from the sequence of input and then it outputs h(0) which together with X(1) is the input for the next step. So, the h(0) and X(1) is the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training
The formula for the current state is

.

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1}+ W_{xh}x_t)$$

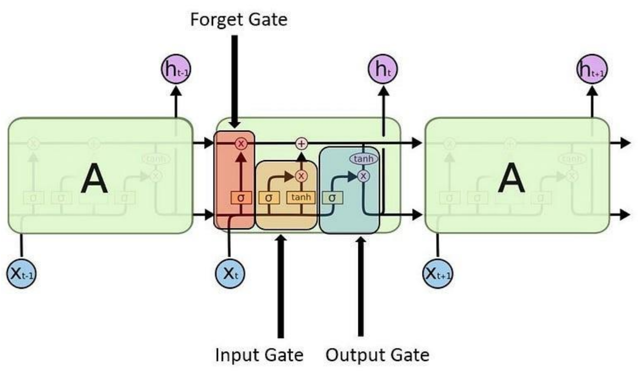
W is weight, h is the single hidden vector, Whh is the weight at previous hidden state, Whx is the weight at current input state, tanh is the Activation funtion, that implements a Non-linearity that squashes the activations to the range[-1.1]

$$y_t = W_{hy}h_t$$

Yt is the output state. Why is the weight at the output state.

Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:



1) <u>Input gate</u> - discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

2) <u>Forget gate</u> - discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input(Xt) and outputs a number between 0(omit this)and 1(keep this)for each number in the cell state Ct−1.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$

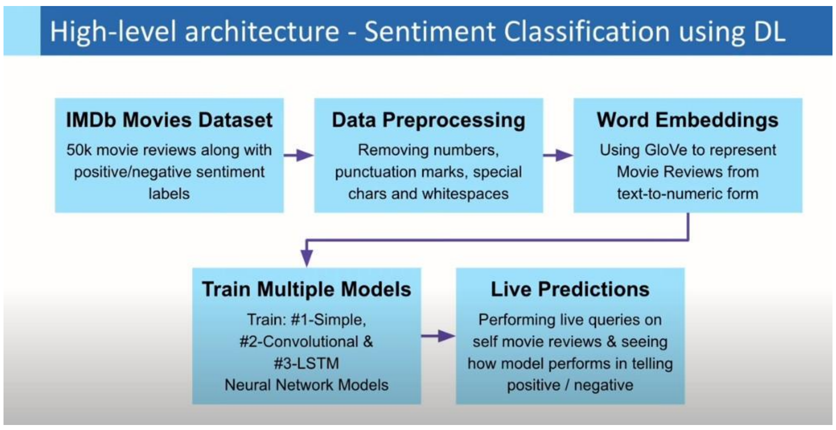
3) <u>Output gate</u> — the input and the memory of the block is used to decide the output. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid.

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \;+\; b_o\right)$$

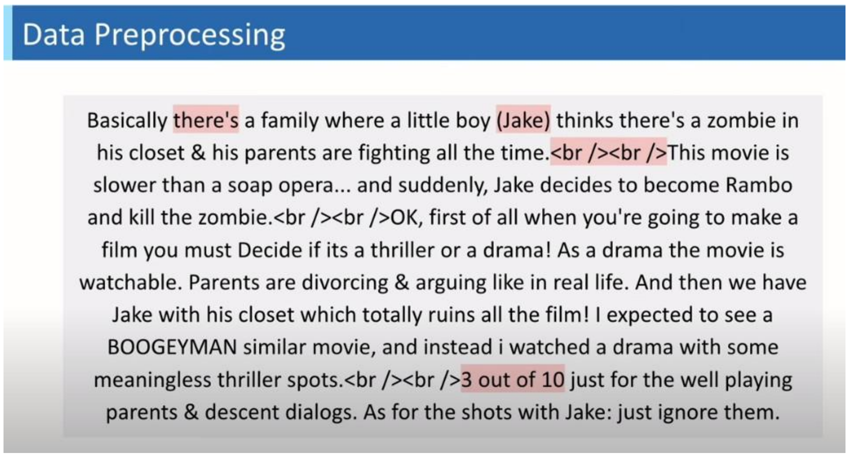
$$h_t = o_t * \tanh\left(C_t\right)$$

Algorithm -

1) Load IMDb Movie Reviews dataset (50,000 reviews)

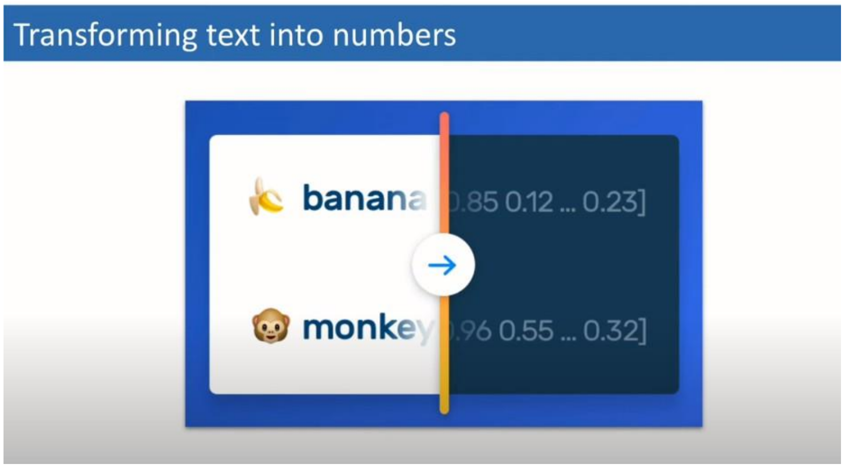

High-level architecture - Sentiment Classification using DL
IMDb Movies Dataset
50k movie reviews along with positive/negative sentiment labels
Data Preprocessing
Removing numbers, punctuation marks, special chars and whitespaces
Word Embeddings
Using GloVe to represent Movie Reviews from text-to-numeric form
Train Multiple Models
Train: #1-Simple, #2-Convolutional & #3-LSTM Neural Network Models
Live Predictions
Performing live queries on self movie reviews & seeing how model performs in telling positive / negative

2) Pre-process dataset by removing special characters, numbers, etc. from user reviews + convert sentiment labels positive & negative to numbers 1 & 0, respectively



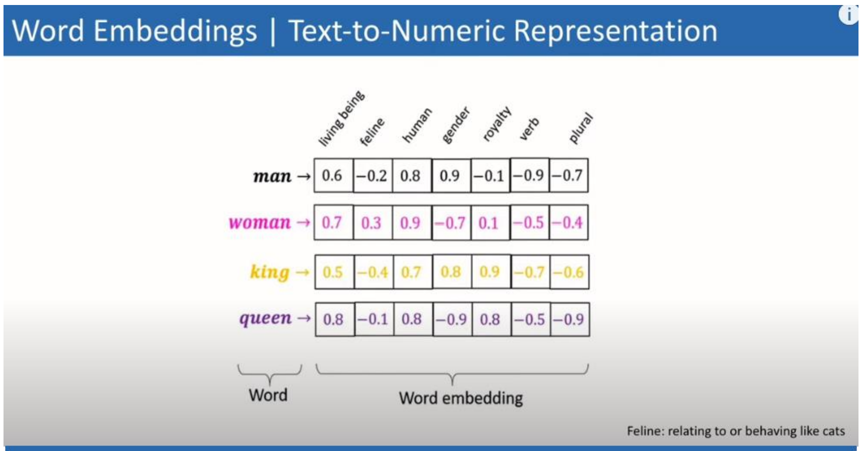
Data Preprocessing

Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time.<br /><br />This movie is slower than a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.<br /><br />OK, first of all when you're going to make a film you must Decide if its a thriller or a drama! As a drama the movie is watchable. Parents are divorcing & arguing like in real life. And then we have Jake with his closet which totally ruins all the film! I expected to see a BOOGEYMAN similar movie, and instead i watched a drama with some meaningless thriller spots.<br /><br />3 out of 10 just for the well playing parents & descent dialogs. As for the shots with Jake: just ignore them.



Transforming text into numbers

3) Import GloVe Word Embedding to build Embedding Dictionary + Use this to build Embedding Matrix for our Corpus

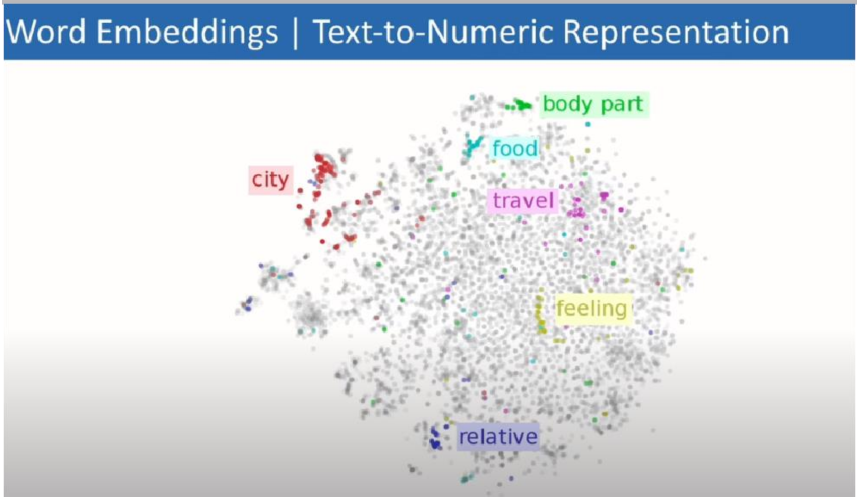


Word Embeddings | Text-to-Numeric Representation

Word	living being	feline	human	gender	royalty	verb	plural
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

Word
Word embedding

Feline: relating to or behaving like cats



Word Embeddings | Text-to-Numeric Representation

4) Model Training using Deep Learning in Keras for separate: Simple Neural Net, CNN and LSTM Models and analyse model performance and resultsPerform Predictions
5) Perform predictions on real IMDb movie reviews

Application -

1) Product Review Sentiment Analysis:
Predict sentiment (positive, negative, neutral) from user reviews for product improvement insights.
2) Customer Feedback Analysis:
Analyze sentiments in customer feedback to understand overall satisfaction and identify areas for improvement.
3) Brand Monitoring:
Monitor social media for product mentions and analyze sentiments to assess brand perception.
4) Market Research:
Analyze sentiments in market surveys to gauge consumer opinions about specific products or features.
5) Quality Assurance in E-commerce:
Automatically categorize and flag reviews with negative sentiments to improve product quality.

Inference -

The process involves data preparation, embedding, model design, training, and evaluation.
The use of different architectures such as Simple Neural Net, CNN, and LSTM allows for comparison and analysis of their performance on sentiment prediction for IMDb movie reviews. The GloVe Word Embedding enhances the models' understanding of the textual data.
Finally, predictions are made on real IMDb movie reviews to assess the models' applicability and accuracy.

Code -

```
import tensorflow as tf
from tensorflow.keras.datasets import imdb from
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Set the parameters
max_features = 10000  # Number of words to consider as features
maxlen = 100  # Cut texts after this number of words (among top max_features most common words)
batch_size = 32

# Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features) # Pad

sequences to have a consistent length for the input to the RNN
```

```
x_train = pad_sequences(x_train, maxlen=maxlen) x_test =
pad_sequences(x_test, maxlen=maxlen)

# Build the RNN model with LSTM
model = Sequential()
model.add(Embedding(max_features, 128)) model.add(LSTM(64,
dropout=0.2, recurrent_dropout=0.2)) model.add(Dense(1,
activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train,
batch_size=batch_size, epochs=5,
validation_data=(x_test, y_test))

# Evaluate the model
score, acc = model.evaluate(x_test, y_test, batch_size=batch_size) print(f'Test score:
{score}')
print(f'Test accuracy: {acc}')
```

Output -

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras- datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
Epoch 1/5
782/782 [==============================] - 319s 401ms/step - loss: 0.4161 - accuracy:
0.8074 - val_loss: 0.3585 - val_accuracy: 0.8412 Epoch
2/5
782/782 [==============================] - 288s 368ms/step - loss: 0.2625 - accuracy:
0.8950 - val_loss: 0.3482 - val_accuracy: 0.8454 Epoch
3/5
782/782 [==============================] - 284s 363ms/step - loss: 0.1931 - accuracy:
0.9244 - val_loss: 0.4158 - val_accuracy: 0.8375 Epoch
4/5
782/782 [==============================] - 285s 365ms/step - loss: 0.1431 - accuracy:
0.9472 - val_loss: 0.4504 - val_accuracy: 0.8412 Epoch
5/5
782/782 [==============================] - 287s 367ms/step - loss: 0.1093 - accuracy:
0.9606 - val_loss: 0.4790 - val_accuracy: 0.8413
782/782 [==============================] - 25s 32ms/step - loss: 0.4790 - accuracy:
```

```
0.8413
Test score: 0.47897636890411377
Test accuracy: 0.8413199782371521
```

References:

https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e
https://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://youtu.be/oWo9SNcyxlI?si=0OzO6SUYZ_FxbTgY

Conclusion: Thus Designed RNN or its variant including LSTM or GRU