



Veermata Jijabai Technological Institute, Mumbai 400019

Experiment No.: 04

Aim: Implement a sentiment classifier using LSTM Network. Use word embeddings Word2Vec to represent words as input to the model that is sentiment classifier. Display accuracy of your model using 4-fold cross validation on the selected dataset.

Hint - Dataset can be used from Kaggle, Hugging Faces or product reviews given by customers.

Name: Kiran K Patil

Enrolment No.: 211070904

Branch: Computer Engineering

Batch: D

Theory:

1. Sentiment Analysis:

- Sentiment analysis is a natural language processing (NLP) task that involves determining the sentiment expressed in a piece of text. Sentiments are often categorized as positive, negative, or neutral. In the context of product reviews or customer feedback, sentiment analysis helps in understanding the opinions and emotions expressed by users.

2. Long Short-Term Memory (LSTM) Networks:

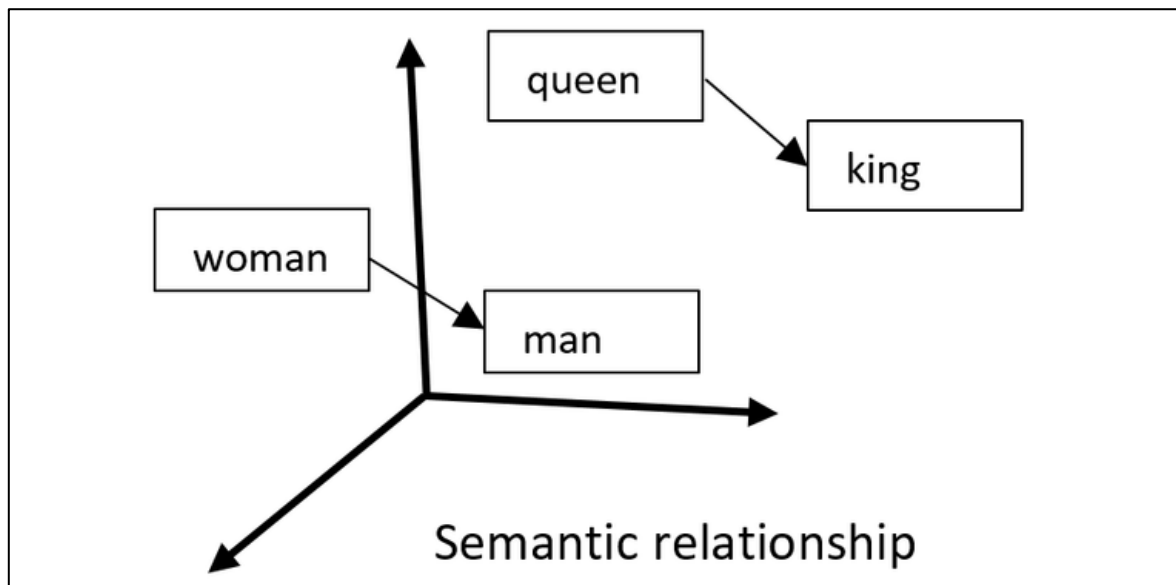
- LSTMs are a type of recurrent neural network (RNN) architecture designed to address the challenges of vanishing gradients in traditional RNNs. The LSTM units are equipped with memory cells and gates (input, output, and forget gates) that allow them to capture and propagate information over long sequences. This makes LSTMs well-suited for tasks involving sequential data, such as sentiment analysis.
- The architecture of an LSTM enables the model to retain information from earlier parts of a sequence, making them effective in capturing dependencies and patterns in language that span across multiple words.

3. LSTM Python for Text Classification

- There are many classic classification algorithms like Decision trees, RFR, SVM, that can fairly do a good job, then why to use LSTM for classification?
- One good reason to use LSTM is that it is effective in memorizing important information.
- If we look and other non-neural network classification techniques they are trained on multiple word as separate inputs that are just word having no actual meaning as a sentence, and while predicting the class it will give the output according to statistics and not according to meaning. That means, every single word is classified into one of the categories.
- This is not the same in LSTM. In LSTM we can use a multiple word string to find out the class to which it belongs. This is very helpful while working with Natural language processing. If we use appropriate layers of embedding and encoding in LSTM, the model will be able to find out the actual meaning in input string and will give the most accurate output class. The following code will elaborate the idea on how text classification is done using LSTM.

4. Word Embeddings (Word2Vec):

- Word embeddings are vector representations of words that capture semantic relationships. Word2Vec, developed by Mikolov et al., is a popular word embedding technique that learns continuous vector representations of words based on their context in a given corpus.
- The Word2Vec model is trained by predicting the context words around a target word in a given window. The resulting embeddings place similar words close to each other in the vector space. This helps the model to capture the meaning and relationships between words in a more meaningful way compared to traditional one-hot encoding.
- The learned word embeddings carry semantic information, allowing the model to understand the contextual meaning of words in the dataset.



4. Cross-Validation:

- Cross-validation is a resampling technique used to assess the performance of a machine learning model. In k-fold cross-validation:
 - The dataset is divided into k subsets (folds).
 - The model is trained k times, each time using k-1 folds for training and the remaining fold for validation.
 - This process ensures that each data point is part of the validation set exactly once.

- Stratified k-fold ensures that the distribution of classes in each fold is representative of the overall distribution in the dataset. This is particularly important in sentiment analysis, where you want to ensure that each fold contains a balanced representation of positive and negative sentiments.
- Cross-validation helps in obtaining a more reliable estimate of the model's performance, reducing the risk of overfitting or underfitting to a specific subset of the data.



These foundational concepts lay the groundwork for building a sentiment classifier using LSTM networks with Word2Vec embeddings and performing a robust evaluation through cross-validation. The subsequent steps involve implementing these theories into code, training the model, and assessing its performance on real-world data.

Implementation:



```
[ ] !pip install tensorflow-gpu

collecting tensorflow-gpu
Downloading tensorflow-gpu-2.12.0.tar.gz (2.6 kB)
error: subprocess-exited-with-error

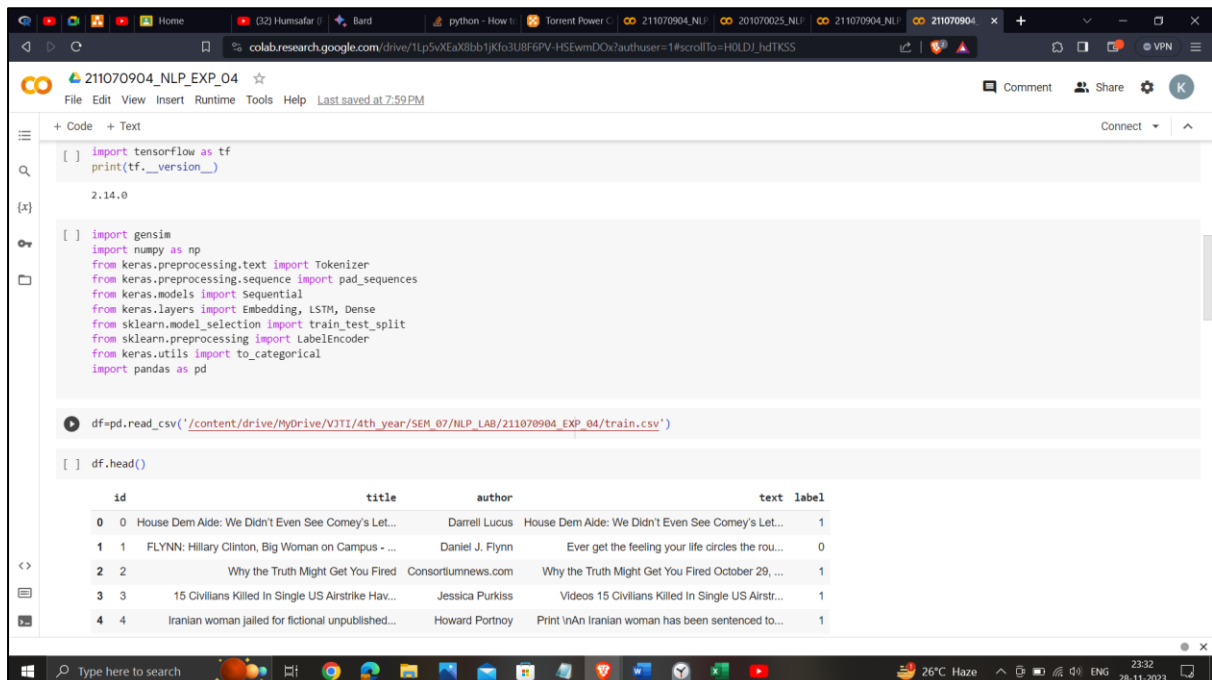
  python setup.py egg_info did not run successfully.
  exit code: 1
  See above for output.

note: This error originates from a subprocess, and is likely not a problem with pip.
Preparing metadata (setup.py) ... error
error: metadata-generation-failed

  Encountered error while generating package metadata.
  See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.

[ ] from google.colab import drive
drive.mount('/content/drive')
```



```
[ ] import tensorflow as tf
print(tf.__version__)

2.14.0

[ ] import gensim
import numpy as np
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.utils import to_categorical
import pandas as pd

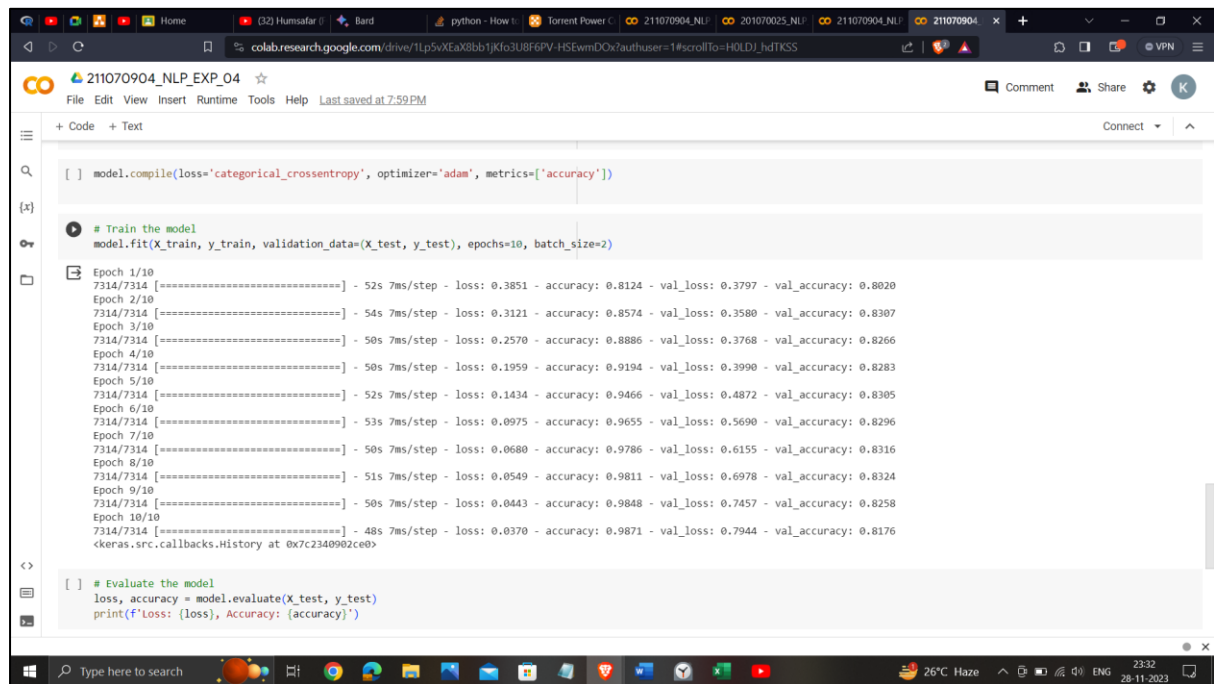
df=pd.read_csv('/content/drive/MyDrive/V3TI/4th_year/SEM_07/NLP_LAB/211070904_EXP_04/train.csv')

[ ] df.head()
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

```
211070904_NLP_EXP_04
File Edit View Insert Runtime Tools Help Last saved at 7:59 PM
+ Code + Text
[ ] df.shape
(20800, 5)
[ ] df.isnull().sum()
id      0
title   558
author  1957
text     39
label    0
dtype: int64
[ ] ##Drop Nan Values
df=df.dropna()
[ ] df.head()
   id  title author text label
0  0  House Dem Aide: We Didn't Even See Comey's Let...  Darrell Lucas  House Dem Aide: We Didn't Even See Comey's Let...  1
1  1  FLYNN: Hillary Clinton, Big Woman on Campus - ...  Daniel J. Flynn  Ever get the feeling your life circles the rou...  0
2  2  Why the Truth Might Get You Fired  Consortiumnews.com  Why the Truth Might Get You Fired October 29, ...  1
3  3  15 Civilians Killed In Single US Airstrike Hav...  Jessica Purkiss  Videos 15 Civilians Killed In Single US Aistr...  1
4  4  Iranian woman jailed for fictional unpublished...  Howard Portnoy  Print \nAn Iranian woman has been sentenced to...  1
[ ] tokenizer = Tokenizer()
tokenizer.fit_on_texts(df.text)
```

```
211070904_NLP_EXP_04
File Edit View Insert Runtime Tools Help Last saved at 7:59 PM
+ Code + Text
[ ] tokenizer = Tokenizer()
tokenizer.fit_on_texts(df.text)
sequences = tokenizer.texts_to_sequences(df.text)
x = pad_sequences(sequences, maxlen=6)
[ ] label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df.label)
y = to_categorical(y)
[ ] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
[ ] # Train a Word2Vec model on your corpus
sentences = [text.split() for text in df.text]
word2vec_model = gensim.models.Word2Vec(sentences, vector_size=100, window=5, min_count=1, sg=0)
[ ] # Create an embedding matrix
vocab_size = len(tokenizer.word_index) + 1
embedding_dim = 100
embedding_matrix = np.zeros((vocab_size, embedding_dim))
for word, i in tokenizer.word_index.items():
    if word in word2vec_model.wv:
        embedding_matrix[i] = word2vec_model.wv[word]
[ ] # Build the LSTM model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, weights=[embedding_matrix], input_length=6, trainable=False))
model.add(LSTM(units=128))
model.add(Dense(units=2, activation='softmax'))
```



```
[ ] model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=2)

Epoch 1/10
7314/7314 [=====] - 52s 7ms/step - loss: 0.3851 - accuracy: 0.8124 - val_loss: 0.3797 - val_accuracy: 0.8020
Epoch 2/10
7314/7314 [=====] - 54s 7ms/step - loss: 0.3121 - accuracy: 0.8574 - val_loss: 0.3580 - val_accuracy: 0.8307
Epoch 3/10
7314/7314 [=====] - 50s 7ms/step - loss: 0.2570 - accuracy: 0.8886 - val_loss: 0.3768 - val_accuracy: 0.8266
Epoch 4/10
7314/7314 [=====] - 50s 7ms/step - loss: 0.1959 - accuracy: 0.9194 - val_loss: 0.3990 - val_accuracy: 0.8283
Epoch 5/10
7314/7314 [=====] - 52s 7ms/step - loss: 0.1434 - accuracy: 0.9466 - val_loss: 0.4872 - val_accuracy: 0.8305
Epoch 6/10
7314/7314 [=====] - 53s 7ms/step - loss: 0.0975 - accuracy: 0.9655 - val_loss: 0.5690 - val_accuracy: 0.8296
Epoch 7/10
7314/7314 [=====] - 50s 7ms/step - loss: 0.0680 - accuracy: 0.9786 - val_loss: 0.6155 - val_accuracy: 0.8316
Epoch 8/10
7314/7314 [=====] - 51s 7ms/step - loss: 0.0549 - accuracy: 0.9811 - val_loss: 0.6978 - val_accuracy: 0.8324
Epoch 9/10
7314/7314 [=====] - 50s 7ms/step - loss: 0.0443 - accuracy: 0.9848 - val_loss: 0.7457 - val_accuracy: 0.8258
Epoch 10/10
7314/7314 [=====] - 48s 7ms/step - loss: 0.0370 - accuracy: 0.9871 - val_loss: 0.7944 - val_accuracy: 0.8176
<keras.src.callbacks.History at 0x7c2340902ce0>

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

<https://colab.research.google.com/drive/1Lp5vXEaX8bb1jKfo3U8F6PV-HSEwmDOx?usp=sharing>

Conclusion:

In conclusion, this experiment provided a foundational understanding of text processing, covering character encoding, Unicode handling, regular expressions, and text normalization. These skills are crucial for effective data preprocessing, paving the way for more advanced natural language processing applications.