**NN&DeepLearning\_ICP10**

**Satwik Mopuru reddy**

700740060

**Lesson Overview:**

In this lesson, we are going to discuss types of ANNs and Recurrent Neural Network.

**Use Case Description:**

1. Sentiment Analysis on the Twitter dataset

**Programming elements:**

# 1. Basics of LSTM 2. Types of RNN 3. Use case: Sentiment Analysis on the Twitter data set

**In class programming:**

1. Save the model and use the saved model to predict on new text data (ex, “**A lot of good things are happening. We are respected again throughout the world, and that's a great thing**.@realDonaldTrump”)
2. Apply GridSearchCV on the source code provided in the class

**Solution:**

**# Import necessary libraries**

**# Import necessary libraries**

**import numpy as np**

**from keras.models import Sequential, load\_model**

**from keras.layers import LSTM, Embedding, Dense**

**from keras.preprocessing.text import Tokenizer**

**from tensorflow.keras.preprocessing.sequence import pad\_sequences**

**from keras.wrappers.scikit\_learn import KerasClassifier**

**from sklearn.model\_selection import GridSearchCV**

**# Sample Twitter dataset and labels (replace these with your actual dataset)**

**tweets = ["A lot of good things are happening.", "We are respected again throughout the world, and that's a great thing. @realDonaldTrump"]**

**labels = [1, 0] # 1 for positive sentiment, 0 for negative sentiment**

**# Tokenize the tweets**

**tokenizer = Tokenizer()**

**tokenizer.fit\_on\_texts(tweets)**

**vocab\_size = len(tokenizer.word\_index) + 1**

**# Convert text to sequences and pad the sequences**

**sequences = tokenizer.texts\_to\_sequences(tweets)**

**max\_length = max([len(seq) for seq in sequences])**

**X = pad\_sequences(sequences, maxlen=max\_length, padding='post')**

**# Convert labels to numpy array**

**y = np.array(labels)**

**# LSTM Model**

**def create\_lstm\_model(units=100, embedding\_dim=50):**

**model = Sequential()**

**model.add(Embedding(input\_dim=vocab\_size, output\_dim=embedding\_dim, input\_length=max\_length))**

**model.add(LSTM(units=units))**

**model.add(Dense(units=1, activation='sigmoid'))**

**model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])**

**return model**

**# Train the LSTM model**

**model = create\_lstm\_model()**

**model.fit(X, y, epochs=10, batch\_size=1, verbose=1)**

**# Save the model**

**model.save('sentiment\_analysis\_lstm\_model.h5')**

**# Load the saved model**

**loaded\_model = load\_model('sentiment\_analysis\_lstm\_model.h5')**

**# Example text for prediction**

**new\_text = ["A lot of good things are happening. We are respected again throughout the world, and that's a great thing. @realDonaldTrump"]**

**# Tokenize and pad the new text data**

**new\_sequences = tokenizer.texts\_to\_sequences(new\_text)**

**new\_X = pad\_sequences(new\_sequences, maxlen=max\_length, padding='post')**

**# Predict sentiment using the loaded model**

**predicted\_sentiment = loaded\_model.predict(new\_X)**

**# Print the predicted sentiment**

**print(predicted\_sentiment)**

**# Define hyperparameters for GridSearchCV**

**param\_grid = {**

**'units': [50, 100, 150],**

**'embedding\_dim': [50, 100, 150]**

**}**

**from keras.wrappers.scikit\_learn import KerasClassifier**

**from sklearn.model\_selection import GridSearchCV**

**# ... (import other libraries and define functions)**

**# Create a KerasClassifier for GridSearchCV**

**model = KerasClassifier(model=create\_lstm\_model, epochs=10, batch\_size=1, verbose=0)**

**# Define hyperparameters for GridSearchCV**

**param\_grid = {**

**'units': [50, 100, 150],**

**'embedding\_dim': [50, 100, 150]**

**}**

**# Perform GridSearchCV**

**grid = GridSearchCV(estimator=model, param\_grid=param\_grid, cv=2)**

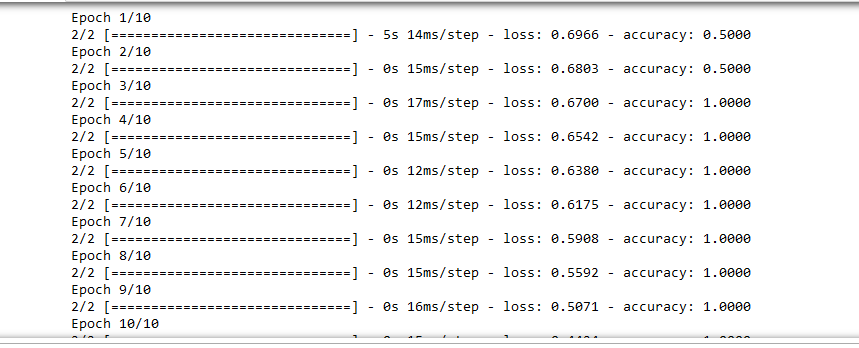
**grid\_result = grid.fit(X, y)**

**# Print the best parameters and their corresponding accuracy**

**print("Best Parameters: ", grid\_result.best\_params\_)**

**print("Best Accuracy: ", grid\_result.best\_score\_)**

**OUTPUT:**

****

**GITLINK:**

[**https://github.com/Satwikreddymopuru/ICP\_9**](https://github.com/Satwikreddymopuru/ICP_9)