EX.NO.: 19
DATE: 17.03.2025

NEXT WORD PREDICTION USING LSTM AND GRU

To build a text prediction model using LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks for next-word prediction based on the Reuters corpus.

PROCEDURE:

1. Import libraries
2. Dataset loading and handle any hidden spaces
3. Perform text preprocessing
4. Perform tokenization and padding
5. Pad the sequences to maintain uniform input length
6. Split data into training and testing
7. Build LSTM model
8. Perform model compilation and training
9. Prompt user input and predict review
10. Perform model evaluation and print accuracy

CODE AND OUTPUT

```
import re
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import reuters
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, GRU, Dense

# Download NLTK dataset
nltk.download('reuters')
nltk.download('punkt')

# Load Reuters corpus
corpus = [reuters.raw(fileid) for fileid in reuters.fileids()[:1000]]  # Limiting to
1000 articles

# Text cleaning function
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    words = text.split()
    return ' '.join(words)

# Clean the corpus
corpus = [clean_text(doc) for doc in corpus]

# Tokenization
```

```
max_words = 5000
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(corpus)
sequences = tokenizer.texts_to_sequences(corpus)

# Create increasing sequences
sequence_length = 10
sequences = [seq[i:i+sequence_length+1] for seq in sequences for i in range(len(seq) -
sequence_length)]

# Convert to numpy array
sequences = np.array(sequences)
X, y = sequences[:, :-1], sequences[:, -1]

# Padding sequences
X = pad_sequences(X, maxlen=sequence_length, padding='pre')

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# LSTM Model
model_lstm = Sequential([
    Embedding(input_dim=max_words, output_dim=8, input_length=sequence_length),
    LSTM(64),
    Dense(max_words, activation='softmax')
])

model_lstm.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train LSTM model
history_lstm = model_lstm.fit(X_train, y_train, epochs=5, batch_size=32, verbose=1,
validation_data=(X_test, y_test))

# GRU Model
model_gru = Sequential([
    Embedding(input_dim=max_words, output_dim=8, input_length=sequence_length),
    GRU(64),
    Dense(max_words, activation='softmax')
])

model_gru.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train GRU model
history_gru = model_gru.fit(X_train, y_train, epochs=5, batch_size=32, verbose=1,
validation_data=(X_test, y_test))
```

```python
# Predict the next word
def predict_next_word(model, text):
    sequence = tokenizer.texts_to_sequences([text])[0]
    sequence = pad_sequences([sequence], maxlen=sequence_length, padding='pre')
    prediction = model.predict(sequence, verbose=0)
    word_index = np.argmax(prediction)
    for word, index in tokenizer.word_index.items():
        if index == word_index:
            return word

# Get user input
user_input = input("Enter your text prompt: ")

# Predict next word for user input
print("LSTM Prediction:", predict_next_word(model_lstm, user_input))
print("GRU Prediction:", predict_next_word(model_gru, user_input))

# Print model accuracy
print("LSTM Accuracy:", history_lstm.history['accuracy'][-1])
print("GRU Accuracy:", history_gru.history['accuracy'][-1])
```

```
[nltk_data] Downloading package reuters to /root/nltk_data...
[nltk_data]   Package reuters is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Epoch 1/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 37s 14ms/step - accuracy: 0.0589 - loss: 6.8346 - val_accuracy: 0.0712 - val_loss: 6.3716
Epoch 2/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 33s 13ms/step - accuracy: 0.0770 - loss: 6.2230 - val_accuracy: 0.1119 - val_loss: 6.0601
Epoch 3/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 34s 13ms/step - accuracy: 0.1171 - loss: 5.8290 - val_accuracy: 0.1283 - val_loss: 5.8256
Epoch 4/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 41s 13ms/step - accuracy: 0.1367 - loss: 5.5073 - val_accuracy: 0.1427 - val_loss: 5.7049
Epoch 5/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 40s 13ms/step - accuracy: 0.1533 - loss: 5.2858 - val_accuracy: 0.1534 - val_loss: 5.6218
Epoch 1/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 39s 15ms/step - accuracy: 0.0606 - loss: 6.8165 - val_accuracy: 0.0920 - val_loss: 6.1942
Epoch 2/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 40s 14ms/step - accuracy: 0.1021 - loss: 5.9870 - val_accuracy: 0.1300 - val_loss: 5.8850
Epoch 3/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 40s 14ms/step - accuracy: 0.1372 - loss: 5.5970 - val_accuracy: 0.1447 - val_loss: 5.7088
Epoch 4/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 45s 15ms/step - accuracy: 0.1571 - loss: 5.3303 - val_accuracy: 0.1614 - val_loss: 5.5952
Epoch 5/5
2523/2523 ━━━━━━━━━━━━━━━━━━━━ 35s 14ms/step - accuracy: 0.1732 - loss: 5.1009 - val_accuracy: 0.1693 - val_loss: 5.5227
Enter your text prompt: there is a cat chasing the
WARNING:tensorflow:5 out of the last 6 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed a
WARNING:tensorflow:6 out of the last 7 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed a
LSTM Prediction: company
GRU Prediction: states
LSTM Accuracy: 0.15480290353298187
GRU Accuracy: 0.17717598378658295
```