ID : 4221386

NAME : Moaz Awad Ali

GitHub : Run-For-Next-Word-Prediction

Kaggle : Rnn-For-Next-Word-Prediction

In [2]:
```
print ("https://www.kaggle.com/code/moazawadali/rnn-for-next-word-prediction")
```

```
https://www.kaggle.com/code/moazawadali/rnn-for-next-word-prediction
```

In [3]:
```
print ("https://github.com/AmazingMoaaz/AI323-Computational-Neuroscience/tree/main/Assignment-Task_3-RNN")
```

```
https://github.com/AmazingMoaaz/AI323-Computational-Neuroscience/tree/main/Assignment-Task_3-RNN
```

In [4]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import matplotlib.pyplot as plt
```

```
2025-04-28 23:02:48.311379: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT f
actory: Attempting to register factory for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1745881368.723493      31 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register fact
ory for plugin cuDNN when one has already been registered
E0000 00:00:1745881368.835590      31 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register fa
ctory for plugin cuBLAS when one has already been registered
```

In [5]:
```
text = """
once upon a time there was a little girl who lived in a village near the forest
she liked to wear a red coat with a hood that her grandmother had made for her
everyone in the village called her little red riding hood one morning her mother asked
her to visit grandmother who lived in the forest but warned her not to talk to strangers
on her way little red riding hood met a wolf who asked where she was going
the girl told the wolf about her grandmother who lived alone in the forest
the wolf ran ahead to the grandmother house and pretended to be the little girl
when little red riding hood arrived the wolf was waiting in the bed disguised as grandmother
little red riding hood noticed something strange about her grandmother and said what big eyes you have
the wolf replied all the better to see you with my dear
"""
```

In [6]:
```
text = text.lower().replace('\n', ' ')
words = [word for word in text.split() if word.isalnum()]
```

```
print(f"Total words in the corpus: {len(words)}")
print(f"First 10 words: {words[:10]}")
```

```
Total words in the corpus: 156
First 10 words: ['once', 'upon', 'a', 'time', 'there', 'was', 'a', 'little', 'girl', 'who']
```

In [7]:
```
sequences = []
for i in range(len(words) - 3):
    sequences.append(words[i:i+4])

print(f"Number of sequences: {len(sequences)}")
print(f"First 3 sequences: {sequences[:3]}")
```

```
Number of sequences: 153
First 3 sequences: [['once', 'upon', 'a', 'time'], ['upon', 'a', 'time', 'there'], ['a', 'time', 'there', 'was']]
```

In [8]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([' '.join(words)])
word_index = tokenizer.word_index
vocab_size = len(word_index) + 1  # +1 for the padding token

print(f"Vocabulary size: {vocab_size}")
print(f"Sample from vocabulary: {list(word_index.items())[:5]}")
```

```
Vocabulary size: 79
Sample from vocabulary: [('the', 1), ('her', 2), ('to', 3), ('a', 4), ('little', 5)]
```

In [9]:
```
X = []  # input sequences (first 3 words)
y = []  # target word (4th word)

for seq in sequences:
    X.append(seq[:3])
    y.append(seq[3])

print(f"X shape: {len(X)} sequences of length 3")
print(f"y shape: {len(y)} words")
print(f"Sample X: {X[:3]}")
print(f"Sample y: {y[:3]}")
```

```
X shape: 153 sequences of length 3
y shape: 153 words
Sample X: [['once', 'upon', 'a'], ['upon', 'a', 'time'], ['a', 'time', 'there']]
Sample y: ['time', 'there', 'was']
```

In [10]:
```
X_seq = tokenizer.texts_to_sequences([' '.join(seq) for seq in X])
y_seq = tokenizer.texts_to_sequences([word for word in y])
```

In [11]:
```
y_seq = [item[0] for item in y_seq]

print(f"First 3 X sequences (encoded): {X_seq[:3]}")
print(f"First 3 y values (encoded): {y_seq[:3]}")
```

```
First 3 X sequences (encoded): [[24, 25, 4], [25, 4, 26], [4, 26, 27]]
First 3 y values (encoded): [26, 27, 13]
```

In [12]:
```
X_array = np.array(X_seq)
y_array = np.array(y_seq)

print(f"X shape: {X_array.shape}")
print(f"y shape: {y_array.shape}")
```

```
X shape: (153, 3)
y shape: (153,)
```

In [13]:
```
y_one_hot = tf.keras.utils.to_categorical(y_array, num_classes=vocab_size)
print(f"y one-hot shape: {y_one_hot.shape}")
```

```
y one-hot shape: (153, 79)
```

In [14]:
```
embedding_dim = 50
lstm_units = 100

model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=3),
    LSTM(units=lstm_units),
    Dense(units=vocab_size, activation='softmax')
])

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

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` i
s deprecated. Just remove it.
  warnings.warn(
I0000 00:00:1745881386.795055      31 gpu_device.cc:2022] Created device /job:localhost/replica:0/task:0/device:GPU:0
with 13942 MB memory:  -> device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5
I0000 00:00:1745881386.795740      31 gpu_device.cc:2022] Created device /job:localhost/replica:0/task:0/device:GPU:1
with 13942 MB memory:  -> device: 1, name: Tesla T4, pci bus id: 0000:00:05.0, compute capability: 7.5
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
lstm (LSTM)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

In [15]:
```
history = model.fit(
    X_array, y_one_hot,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    verbose=1
)
```

```
Epoch 1/100
I0000 00:00:1745881391.191389      96 cuda_dnn.cc:529] Loaded cuDNN version 90300
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 5s 109ms/step - accuracy: 0.0180 - loss: 4.3676 - val_accuracy: 0.0645 - val_loss: 4.3687
Epoch 2/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1295 - loss: 4.3592 - val_accuracy: 0.0645 - val_loss: 4.3678
Epoch 3/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1115 - loss: 4.3509 - val_accuracy: 0.0645 - val_loss: 4.3671
Epoch 4/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0925 - loss: 4.3391 - val_accuracy: 0.0645 - val_loss: 4.3663
Epoch 5/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1071 - loss: 4.3274 - val_accuracy: 0.0645 - val_loss: 4.3657
Epoch 6/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0831 - loss: 4.3137 - val_accuracy: 0.0645 - val_loss: 4.3652
Epoch 7/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0871 - loss: 4.2929 - val_accuracy: 0.0645 - val_loss: 4.3651
Epoch 8/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0746 - loss: 4.2606 - val_accuracy: 0.0645 - val_loss: 4.3660
Epoch 9/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1027 - loss: 4.2209 - val_accuracy: 0.0645 - val_loss: 4.3695
Epoch 10/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0861 - loss: 4.1649 - val_accuracy: 0.0645 - val_loss: 4.3787
Epoch 11/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0736 - loss: 4.1062 - val_accuracy: 0.0645 - val_loss: 4.3998
Epoch 12/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0725 - loss: 4.0007 - val_accuracy: 0.0645 - val_loss: 4.4465
Epoch 13/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1027 - loss: 3.8367 - val_accuracy: 0.0645 - val_loss: 4.5453
Epoch 14/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0819 - loss: 3.7577 - val_accuracy: 0.0645 - val_loss: 4.6943
Epoch 15/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0861 - loss: 3.6754 - val_accuracy: 0.0645 - val_loss: 4.8004
Epoch 16/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.0756 - loss: 3.7102 - val_accuracy: 0.0645 - val_loss: 4.7957
Epoch 17/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0902 - loss: 3.6190 - val_accuracy: 0.0645 - val_loss: 4.7696
Epoch 18/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.0954 - loss: 3.5751 - val_accuracy: 0.0968 - val_loss: 4.7607
Epoch 19/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.1029 - loss: 3.5183 - val_accuracy: 0.0968 - val_loss: 4.7720
Epoch 20/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.0832 - loss: 3.4684 - val_accuracy: 0.1613 - val_loss: 4.7959
Epoch 21/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1051 - loss: 3.4465 - val_accuracy: 0.1290 - val_loss: 4.8253
Epoch 22/100
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1098 - loss: 3.4207 - val_accuracy: 0.1613 - val_loss: 4.8433
Epoch 23/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.1349 - loss: 3.3369 - val_accuracy: 0.1613 - val_loss: 4.8748
Epoch 24/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1675 - loss: 3.2977 - val_accuracy: 0.1613 - val_loss: 4.8734
Epoch 25/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1470 - loss: 3.2930 - val_accuracy: 0.1290 - val_loss: 4.8644
Epoch 26/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.1937 - loss: 3.2056 - val_accuracy: 0.1290 - val_loss: 4.8705
Epoch 27/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.2409 - loss: 3.0623 - val_accuracy: 0.1613 - val_loss: 4.8919
Epoch 28/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.2414 - loss: 3.0203 - val_accuracy: 0.1613 - val_loss: 4.8960
Epoch 29/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.2901 - loss: 2.9800 - val_accuracy: 0.1613 - val_loss: 4.9033
Epoch 30/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.2923 - loss: 2.8472 - val_accuracy: 0.1613 - val_loss: 4.9075
Epoch 31/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3480 - loss: 2.7907 - val_accuracy: 0.1613 - val_loss: 4.9242
Epoch 32/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.3606 - loss: 2.6701 - val_accuracy: 0.1613 - val_loss: 4.9446
Epoch 33/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.3356 - loss: 2.6748 - val_accuracy: 0.1613 - val_loss: 4.9564
Epoch 34/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3391 - loss: 2.6555 - val_accuracy: 0.1290 - val_loss: 4.9615
Epoch 35/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4027 - loss: 2.4952 - val_accuracy: 0.1290 - val_loss: 4.9836
Epoch 36/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3865 - loss: 2.4524 - val_accuracy: 0.1290 - val_loss: 5.0155
Epoch 37/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4389 - loss: 2.3374 - val_accuracy: 0.1290 - val_loss: 5.0433
Epoch 38/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4130 - loss: 2.3124 - val_accuracy: 0.1290 - val_loss: 5.0571
Epoch 39/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3933 - loss: 2.3259 - val_accuracy: 0.1290 - val_loss: 5.0870
Epoch 40/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4349 - loss: 2.1494 - val_accuracy: 0.1290 - val_loss: 5.1041
Epoch 41/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4293 - loss: 2.1884 - val_accuracy: 0.1290 - val_loss: 5.1357
Epoch 42/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4464 - loss: 2.1110 - val_accuracy: 0.1290 - val_loss: 5.1550
Epoch 43/100
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4540 - loss: 2.0151 - val_accuracy: 0.1613 - val_loss: 5.1880
Epoch 44/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5606 - loss: 1.8255 - val_accuracy: 0.1613 - val_loss: 5.2315
Epoch 45/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.4942 - loss: 1.9277 - val_accuracy: 0.1290 - val_loss: 5.2685
Epoch 46/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5457 - loss: 1.8016 - val_accuracy: 0.1290 - val_loss: 5.3076
Epoch 47/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5627 - loss: 1.6984 - val_accuracy: 0.1613 - val_loss: 5.3310
Epoch 48/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6066 - loss: 1.5969 - val_accuracy: 0.1613 - val_loss: 5.3591
Epoch 49/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5733 - loss: 1.5681 - val_accuracy: 0.1290 - val_loss: 5.3823
Epoch 50/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.5520 - loss: 1.6218 - val_accuracy: 0.1290 - val_loss: 5.4112
Epoch 51/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6412 - loss: 1.4570 - val_accuracy: 0.1290 - val_loss: 5.4539
Epoch 52/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6540 - loss: 1.4031 - val_accuracy: 0.1290 - val_loss: 5.4968
Epoch 53/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6921 - loss: 1.3379 - val_accuracy: 0.1290 - val_loss: 5.5333
Epoch 54/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7805 - loss: 1.1970 - val_accuracy: 0.1290 - val_loss: 5.5739
Epoch 55/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7277 - loss: 1.2762 - val_accuracy: 0.1290 - val_loss: 5.5986
Epoch 56/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7747 - loss: 1.1911 - val_accuracy: 0.1290 - val_loss: 5.6346
Epoch 57/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7936 - loss: 1.1253 - val_accuracy: 0.1290 - val_loss: 5.6637
Epoch 58/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7793 - loss: 1.1422 - val_accuracy: 0.1290 - val_loss: 5.7015
Epoch 59/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7929 - loss: 1.0494 - val_accuracy: 0.1290 - val_loss: 5.7255
Epoch 60/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8119 - loss: 1.0409 - val_accuracy: 0.1290 - val_loss: 5.7534
Epoch 61/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8152 - loss: 0.9283 - val_accuracy: 0.1290 - val_loss: 5.7838
Epoch 62/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8145 - loss: 1.0083 - val_accuracy: 0.1290 - val_loss: 5.8198
Epoch 63/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8743 - loss: 0.8701 - val_accuracy: 0.1290 - val_loss: 5.8437
Epoch 64/100
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8693 - loss: 0.8435 - val_accuracy: 0.1290 - val_loss: 5.8824
Epoch 65/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8600 - loss: 0.8629 - val_accuracy: 0.1290 - val_loss: 5.9238
Epoch 66/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8904 - loss: 0.7795 - val_accuracy: 0.1290 - val_loss: 5.9501
Epoch 67/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8760 - loss: 0.7881 - val_accuracy: 0.1290 - val_loss: 5.9761
Epoch 68/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8926 - loss: 0.7247 - val_accuracy: 0.1290 - val_loss: 5.9906
Epoch 69/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8728 - loss: 0.6866 - val_accuracy: 0.1290 - val_loss: 6.0115
Epoch 70/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9013 - loss: 0.6863 - val_accuracy: 0.1290 - val_loss: 6.0365
Epoch 71/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9108 - loss: 0.6266 - val_accuracy: 0.1290 - val_loss: 6.0652
Epoch 72/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9424 - loss: 0.5667 - val_accuracy: 0.1290 - val_loss: 6.0971
Epoch 73/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9276 - loss: 0.5688 - val_accuracy: 0.1290 - val_loss: 6.1113
Epoch 74/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9270 - loss: 0.5438 - val_accuracy: 0.1290 - val_loss: 6.1280
Epoch 75/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9607 - loss: 0.4966 - val_accuracy: 0.1290 - val_loss: 6.1568
Epoch 76/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9742 - loss: 0.4616 - val_accuracy: 0.1290 - val_loss: 6.1716
Epoch 77/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9302 - loss: 0.5016 - val_accuracy: 0.1290 - val_loss: 6.1911
Epoch 78/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9605 - loss: 0.4564 - val_accuracy: 0.1290 - val_loss: 6.2123
Epoch 79/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9564 - loss: 0.4644 - val_accuracy: 0.1290 - val_loss: 6.2349
Epoch 80/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9574 - loss: 0.4229 - val_accuracy: 0.1290 - val_loss: 6.2543
Epoch 81/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9574 - loss: 0.4096 - val_accuracy: 0.1290 - val_loss: 6.2755
Epoch 82/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9628 - loss: 0.4058 - val_accuracy: 0.1290 - val_loss: 6.2843
Epoch 83/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9721 - loss: 0.3865 - val_accuracy: 0.1290 - val_loss: 6.2940
Epoch 84/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9595 - loss: 0.3699 - val_accuracy: 0.1290 - val_loss: 6.3056
Epoch 85/100
```

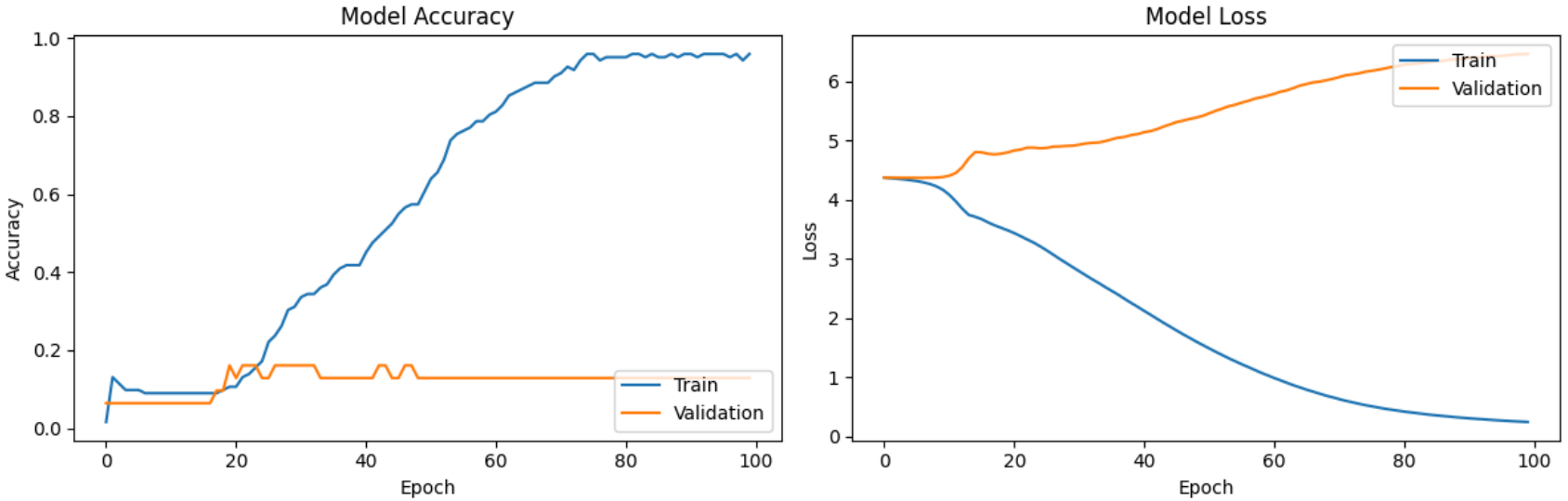
```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9482 - loss: 0.3928 - val_accuracy: 0.1290 - val_loss: 6.3219
Epoch 86/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9480 - loss: 0.3561 - val_accuracy: 0.1290 - val_loss: 6.3348
Epoch 87/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9626 - loss: 0.3304 - val_accuracy: 0.1290 - val_loss: 6.3404
Epoch 88/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9440 - loss: 0.3571 - val_accuracy: 0.1290 - val_loss: 6.3604
Epoch 89/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9532 - loss: 0.3076 - val_accuracy: 0.1290 - val_loss: 6.3660
Epoch 90/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9565 - loss: 0.3455 - val_accuracy: 0.1290 - val_loss: 6.3763
Epoch 91/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9482 - loss: 0.3112 - val_accuracy: 0.1290 - val_loss: 6.3908
Epoch 92/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9626 - loss: 0.2756 - val_accuracy: 0.1290 - val_loss: 6.3948
Epoch 93/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9659 - loss: 0.2846 - val_accuracy: 0.1290 - val_loss: 6.4033
Epoch 94/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.9607 - loss: 0.2809 - val_accuracy: 0.1290 - val_loss: 6.4176
Epoch 95/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9617 - loss: 0.2615 - val_accuracy: 0.1290 - val_loss: 6.4195
Epoch 96/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9638 - loss: 0.2599 - val_accuracy: 0.1290 - val_loss: 6.4246
Epoch 97/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9678 - loss: 0.2428 - val_accuracy: 0.1290 - val_loss: 6.4351
Epoch 98/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9482 - loss: 0.2671 - val_accuracy: 0.1290 - val_loss: 6.4481
Epoch 99/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9333 - loss: 0.2615 - val_accuracy: 0.1290 - val_loss: 6.4529
Epoch 100/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9649 - loss: 0.2513 - val_accuracy: 0.1290 - val_loss: 6.4572
```

In [16]:
```
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='lower right')
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.tight_layout()
plt.show()
```



In [17]:

```
def predict_next_word(input_text):
    # Clean and tokenize the input text
    words = input_text.lower().split()

    # Check if we have exactly 3 words
    if len(words) != 3:
        return "Please provide exactly 3 words as input."

    # Check if all words are in the vocabulary
    for word in words:
        if word not in word_index:
            return f"Word '{word}' is not in the vocabulary. Please try another word."

    # Convert to sequence
```

```
    seq = tokenizer.texts_to_sequences([input_text])[0]

    # Make prediction
    prediction = model.predict(np.array([seq]))
    predicted_index = np.argmax(prediction)

    # Get the word from the index
    for word, index in word_index.items():
        if index == predicted_index:
            return word

    return "Could not find the predicted word in the vocabulary."
```

In [18]:

```
test_inputs = [
    "once upon a",
    "little red riding",
    "in the forest",
    "she liked to"
]

for input_text in test_inputs:
    predicted_word = predict_next_word(input_text)
    print(f"Input: '{input_text}'")
    print(f"Predicted next word: '{predicted_word}'")
    print(f"Complete sequence: '{input_text} {predicted_word}'")
    print("-" * 50)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 223ms/step
Input: 'once upon a'
Predicted next word: 'time'
Complete sequence: 'once upon a time'
--------------------------------------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
Input: 'little red riding'
Predicted next word: 'hood'
Complete sequence: 'little red riding hood'
--------------------------------------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Input: 'in the forest'
Predicted next word: 'but'
Complete sequence: 'in the forest but'
--------------------------------------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Input: 'she liked to'
Predicted next word: 'wear'
Complete sequence: 'she liked to wear'
--------------------------------------------------
```

In [19]:
```
user_input = "girl told the"
predicted_word = predict_next_word(user_input)
print(f"Predicted next word: '{predicted_word}'")
print(f"Complete sequence: '{user_input} {predicted_word}'")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
Predicted next word: 'wolf'
Complete sequence: 'girl told the wolf'
```