LAB-5 Text corpus creation and binary classification using DNN

Name:P.Asha Belcilda

Rollno:225229104

Steps

In [3]:
```
import numpy as np
def load_data():
# Load motivational quotes and demotivational quotes from files
    with open('tamil quotes.txt', 'r', encoding='utf-8') as f:
        motivational = f.readlines()
    with open('tamil quotes.txt', 'r', encoding='utf-8') as f:
        demotivational = f.readlines()
# Combine both classes of quotes and create labels (1 for motivational, 0 for
    quotes = motivational + demotivational
    labels = np.concatenate([np.ones(len(motivational)), np.zeros(len(demotiva
    return quotes, labels
quotes, labels = load_data()
```

In [6]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
def preprocess_data(quotes):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(quotes)
    sequences = tokenizer.texts_to_sequences(quotes)
    vocab_size = len(tokenizer.word_index) + 1
# Pad the sequences to have the same length
    max_sequence_length = max(len(seq) for seq in sequences)
    padded_sequences = pad_sequences(sequences, maxlen=max_sequence_length, pa
    return padded_sequences, vocab_size
X, vocab_size = preprocess_data(quotes)
```

```
C:\Users\ashac\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarnin
g: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [7]:
```
load_data()
```

Out[7]:
```
(['Statement|Label\n',
  'ஒவ்வொரு சிறிய மாற்றமும் பெரிய வெற்றியின் ஒரு பகுதியாகும்|Motiva
tion\n',
  'மனம் துயரத்தின் பின் நிலைக்கும், வாழ்க்கை மாறிவிடுகின்றது.|Demoti
vation\n',
  'வெற்றி இலக்கை அடைய தோல்விகள் படிகட்டுகள்.|Motivation\n',
  'காலம் வருகின்றது போல் இருக்கும், உலகம் புரிகின்றது போல் இருக்க
முடியாது.|Demotivation\n',
  'உங்களால் பறக்க முடியாவிட்டால் ஓடுங்கள்.|Motivation\n',
  'பொய்யான வாழ்க்கைக்கு உண்மையான குடியிடுவோம்|Demotivation\n',
  'நம்மீது நம்பிக்கை நமக்கிருக்கும் வரை வாழ்க்கை நம்வசம்.|Motivation
\n',
  'என் உயிர் துளைக்கும் வழியில் அழிவுகள் போகின்றன.|Demotivation\n',
  'நீங்களே கட்டியெழுப்பும் சுவர்களால் மட்டுமே நீங்கள் அடைக்கப்ப
ட்டுள்ளீர்கள்.|DeMotivation\n',
  'உன் திறமையை வெளி காட்டு, உலகம் உன்னை கண்டறியும்.|Motivatio
n\n',
  'நினைவுகள் மறைந்து விட்டு நினைக்க முடியாத பிரிவுகள் போன்றன.|D
emotivation\n',
  'நீங்கள் செய்யாவிட்டால் கனவுகள் செயல்படாது.|Demotivation\n',
  'உலகில் நீங்கள் காண விரும்பும் மாற்றமாக இருங்கள்.|Motivation\n',
  'மனிதர் வாழ்க்கை, அவர் நினைவின் அழிவு.|Demotivation\n',
  'செயல் அனைத்து வெற்றிகளுக்கும் அடித்தளமாகும்.|Motivation\n',
  'வெற்றி என்பது கொடுத்து பெறுவது அல்ல முயன்று அடைவது.|Motivati
on\n',
  'காலம் கெட்டதும் மனம் கெட்டது மிகுந்த துன்பம்.|Demotivation\n',
  'சவால்கள் மேல் சவாரி செய்வதே வெற்றிக்கு வழி.|Motivation\n',
  'பலனை எதிர்ப்பார்க்காதே, நன்மையைச் செய்.|Motivation\n',
  'வாழ்க்கைக்கு எந்த திறனும் பேசியது இல்லை.|Demotivation\n',
  'Statement|Label\n',
  'ஒவ்வொரு சிறிய மாற்றமும் பெரிய வெற்றியின் ஒரு பகுதியாகும்|Motiva
tion\n',
  'மனம் துயரத்தின் பின் நிலைக்கும், வாழ்க்கை மாறிவிடுகின்றது.|Demoti
vation\n',
  'வெற்றி இலக்கை அடைய தோல்விகள் படிகட்டுகள்.|Motivation\n',
  'காலம் வருகின்றது போல் இருக்கும், உலகம் புரிகின்றது போல் இருக்க
முடியாது.|Demotivation\n',
  'உங்களால் பறக்க முடியாவிட்டால் ஓடுங்கள்.|Motivation\n',
  'பொய்யான வாழ்க்கைக்கு உண்மையான குடியிடுவோம்|Demotivation\n',
  'நம்மீது நம்பிக்கை நமக்கிருக்கும் வரை வாழ்க்கை நம்வசம்.|Motivation
\n',
  'என் உயிர் துளைக்கும் வழியில் அழிவுகள் போகின்றன.|Demotivation\n',
  'நீங்களே கட்டியெழுப்பும் சுவர்களால் மட்டுமே நீங்கள் அடைக்கப்ப
ட்டுள்ளீர்கள்.|DeMotivation\n',
  'உன் திறமையை வெளி காட்டு, உலகம் உன்னை கண்டறியும்.|Motivatio
n\n',
  'நினைவுகள் மறைந்து விட்டு நினைக்க முடியாத பிரிவுகள் போன்றன.|D
emotivation\n',
  'நீங்கள் செய்யாவிட்டால் கனவுகள் செயல்படாது.|Demotivation\n',
  'உலகில் நீங்கள் காண விரும்பும் மாற்றமாக இருங்கள்.|Motivation\n',
  'மனிதர் வாழ்க்கை, அவர் நினைவின் அழிவு.|Demotivation\n',
  'செயல் அனைத்து வெற்றிகளுக்கும் அடித்தளமாகும்.|Motivation\n',
  'வெற்றி என்பது கொடுத்து பெறுவது அல்ல முயன்று அடைவது.|Motivati
on\n',
  'காலம் கெட்டதும் மனம் கெட்டது மிகுந்த துன்பம்.|Demotivation\n',
  'சவால்கள் மேல் சவாரி செய்வதே வெற்றிக்கு வழி.|Motivation\n',
  'பலனை எதிர்ப்பார்க்காதே, நன்மையைச் செய்.|Motivation\n',
```

```
 'வாழ்க்கைக்கு எந்த திறனும் பேசியது இல்லை.|Demotivation\n'],
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0.]))
```

In [11]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
def create_model(nodes=32, layers=1):
    model = Sequential()
    model.add(Embedding(input_dim=vocab_size, output_dim=nodes, input_length=X
    for _ in range(layers):
        model.add(LSTM(nodes, return_sequences=True))
    model.add(LSTM(nodes))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accu
    return model
```

In [15]:

```
from sklearn.model_selection import train_test_split

import time
import matplotlib.pyplot as plt
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
# Function to train and evaluate the model
def train_and_evaluate_model(nodes, layers):
    model = create_model(nodes=nodes, layers=layers)
    start_time = time.time()
    model.fit(X_train, y_train, epochs=5, batch_size=128, verbose=1)
    end_time = time.time()
    _, train_accuracy = model.evaluate(X_train, y_train)
    _, test_accuracy = model.evaluate(X_test, y_test)
    return train_accuracy, test_accuracy, end_time - start_time
# Define the configurations to evaluate
nodes_list = [6, 32, 64, 128, 256, 512, 1024]
layers_list = [1, 2, 3, 4, 5]
# Create a dictionary to store the results for each configuration
results = {}
# Find the best configuration
best_accuracy = 0
best_config = None
# Evaluate each configuration and store the components
for nodes in nodes_list:
    if nodes == 32: # Evaluate all layers for node 32
        for layers in layers_list:
            train_accuracy, test_accuracy, running_time = train_and_evaluate_m
            results[(nodes, layers)] = {
                "Train Accuracy": train_accuracy,
                "Test Accuracy": test_accuracy,
                "Running Time": running_time
        }
            if test_accuracy > best_accuracy:
                best_accuracy = test_accuracy
                best_config = (nodes, layers)
    else: # Only evaluate the first layer for other nodes
        layers = 1
        train_accuracy, test_accuracy, running_time = train_and_evaluate_model
        results[(nodes, layers)] = {
                "Train Accuracy": train_accuracy,
                "Test Accuracy": test_accuracy,
                "Running Time": running_time
        }
        if test_accuracy > best_accuracy:
            best_accuracy = test_accuracy
            best_config = (nodes, layers)
# Print the results for all configurations
for config, values in results.items():
    node, layers = config
    print("Node: {}, Layers: {}".format(node, layers))
    print("Training Accuracy:", values["Train Accuracy"])
    print("Testing Accuracy:", values["Test Accuracy"])
    print("Running Time:", values["Running Time"])
```

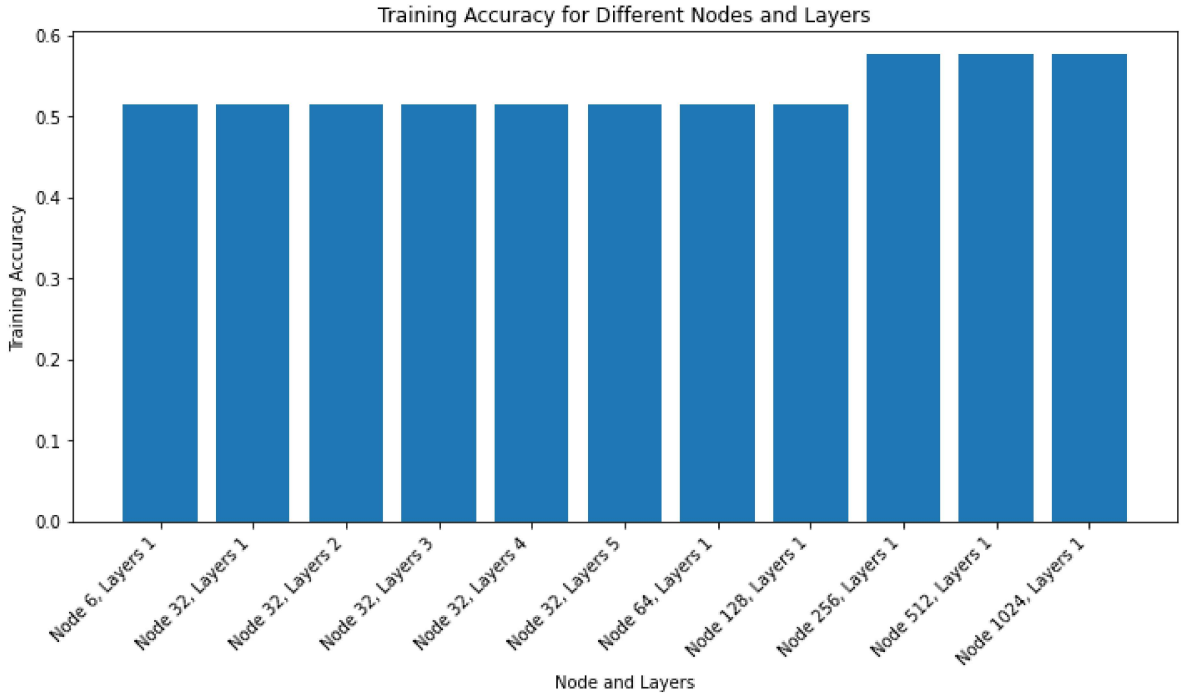
```
    print("-" * 30)
```

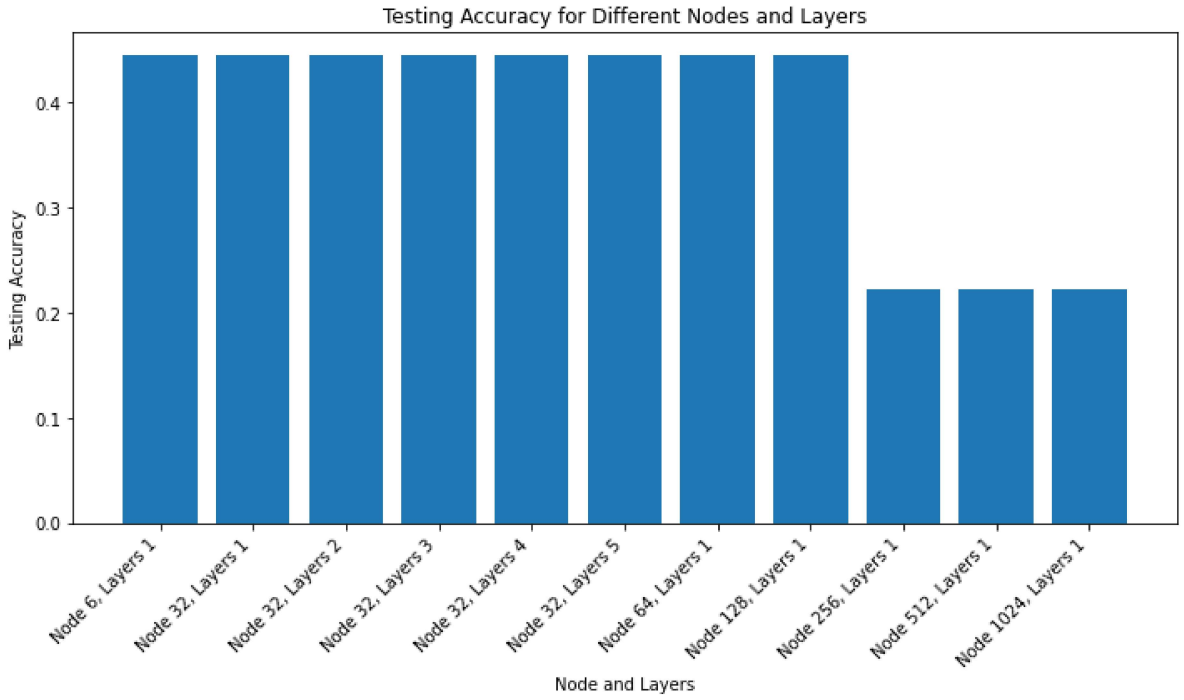
```
Epoch 1/5
1/1 [==============================] - 13s 13s/step - loss: 0.6932 - accur
acy: 0.4848
Epoch 2/5
1/1 [==============================] - 0s 16ms/step - loss: 0.6931 - accur
acy: 0.5152
Epoch 3/5
1/1 [==============================] - 0s 16ms/step - loss: 0.6930 - accur
acy: 0.5152
Epoch 4/5
1/1 [==============================] - 0s 16ms/step - loss: 0.6929 - accur
acy: 0.5152
Epoch 5/5
1/1 [==============================] - 0s 19ms/step - loss: 0.6929 - accur
acy: 0.5152
2/2 [==============================] - 3s 16ms/step - loss: 0.6928 - accur
acy: 0.5152
1/1 [==============================] - 0s 96ms/step - loss: 0.6946 - accur
acy: 0.4444
```

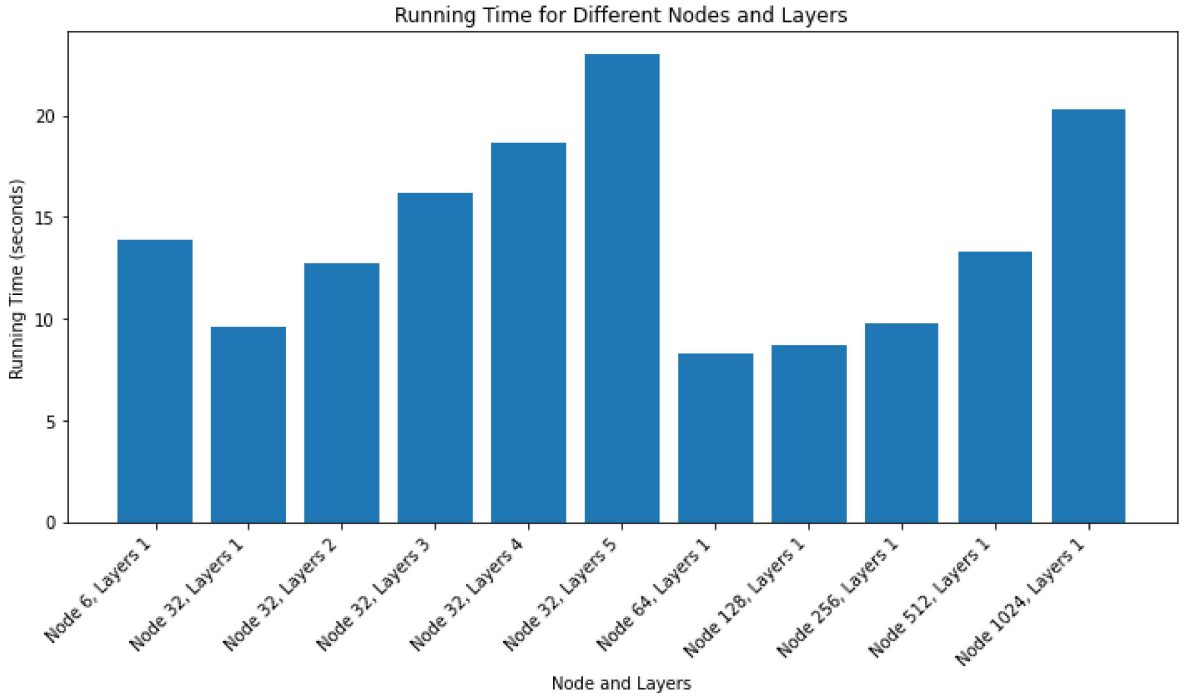
In [16]:

```
# Plot the graphs for Training Accuracy, Testing Accuracy, and Running Time fo

x_labels = ["Node {}, Layers {}".format(node, layers) for node, layers in resu
train_accuracies = [values["Train Accuracy"] for values in results.values()]
test_accuracies = [values["Test Accuracy"] for values in results.values()]
running_times = [values["Running Time"] for values in results.values()]
# Plot the bar graph for Training Accuracy
plt.figure(figsize=(10, 6))
plt.bar(x_labels, train_accuracies)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Node and Layers")
plt.ylabel("Training Accuracy")
plt.title("Training Accuracy for Different Nodes and Layers")
plt.tight_layout()
plt.show()
# Plot the bar graph for Testing Accuracy
plt.figure(figsize=(10, 6))
plt.bar(x_labels, test_accuracies)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Node and Layers")
plt.ylabel("Testing Accuracy")
plt.title("Testing Accuracy for Different Nodes and Layers")
plt.tight_layout()
plt.show()
# Plot the bar graph for Running Time
plt.figure(figsize=(10, 6))
plt.bar(x_labels, running_times)
plt.xticks(rotation=45, ha='right')
plt.xlabel("Node and Layers")
plt.ylabel("Running Time (seconds)")
plt.title("Running Time for Different Nodes and Layers")
plt.tight_layout()
plt.show()
# Print the results for the best configuration
print("BEST CONFIGURATION:", best_config)
print("Components of the Best Configuration:")
print("Training Accuracy:", results[best_config]["Train Accuracy"])
print("Testing Accuracy:", results[best_config]["Test Accuracy"])
print("Running Time:", results[best_config]["Running Time"])
```


Training Accuracy for Different Nodes and Layers
Training Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Node 6, Layers 1
Node 32, Layers 1
Node 32, Layers 2
Node 32, Layers 3
Node 32, Layers 4
Node 32, Layers 5
Node 64, Layers 1
Node 128, Layers 1
Node 256, Layers 1
Node 512, Layers 1
Node 1024, Layers 1
Node and Layers


Testing Accuracy for Different Nodes and Layers
Testing Accuracy
0.0
0.1
0.2
0.3
0.4
Node 6, Layers 1
Node 32, Layers 1
Node 32, Layers 2
Node 32, Layers 3
Node 32, Layers 4
Node 32, Layers 5
Node 64, Layers 1
Node 128, Layers 1
Node 256, Layers 1
Node 512, Layers 1
Node 1024, Layers 1
Node and Layers



BEST CONFIGURATION: (6, 1)
Components of the Best Configuration:
Training Accuracy: 0.5151515007019043
Testing Accuracy: 0.4444444477558136
Running Time: 13.849305391311646

In [37]:

```python
# Function to read text from a file and return the words

def read_text_from_file(file_path):
    with open(file_path, "r", encoding="utf-8") as file:
        text = file.read()
    return text
# Function to clean and tokenize the text into words
def preprocess_text(text):
    # Remove unwanted characters and split the text into words
    words = text.strip().replace("\n", " ").split(" ")
    # Remove any empty strings from the list of words
    words = [word.strip() for word in words if word.strip()]
    return words
# Function to count the word frequency in a list of words
def count_word_frequency(words):
    word_frequency = {}
    for word in words:
        if word in word_frequency:
            word_frequency[word] += 1
        else:
            word_frequency[word] = 1
    return word_frequency
# Read and preprocess
motivational_text = read_text_from_file("tamil quotes.txt")
motivational_words = preprocess_text(motivational_text)
motivational_word_frequency = count_word_frequency(motivational_words)
# Read and preprocess
demotivational_text = read_text_from_file("tamil quotes.txt")
demotivational_words = preprocess_text(demotivational_text)
demotivational_word_frequency = count_word_frequency(demotivational_words)
# Display the word frequency
print("Word Frequency for motivational:")
for word, frequency in motivational_word_frequency.items():
    print(f"{word}: {frequency}")
# Display the word frequency
print("\nWord Frequency for demotivational:")
for word, frequency in demotivational_word_frequency.items():
    print(f"{word}: {frequency}")
```

```
Word Frequency for motivational:
Statement|Label: 1
ஒவ்வொரு: 1
சிறிய: 1
மாற்றமும்: 1
பெரிய: 1
வெற்றியின்: 1
ஒரு: 1
பகுதியாகும்|Motivation: 1
மனம்: 2
துயரத்தின்: 1
பின்: 1
நிலைக்கும்,: 1
வாழ்க்கை: 2
மாறிவிடுகின்றது.|Demotion: 1
வெற்றி: 2
இலக்கை: 1
அடைய: 1
தோல்விகள்: 1
```

In [26]:
```
!pip install prettytable
```

```
Collecting prettytable
  Downloading prettytable-3.8.0-py3-none-any.whl (27 kB)
Requirement already satisfied: wcwidth in c:\users\ashac\anaconda3\lib\site-p
ackages (from prettytable) (0.2.5)
Installing collected packages: prettytable
Successfully installed prettytable-3.8.0
```

In [27]:
```
!pip install tabulate
```

```
Requirement already satisfied: tabulate in c:\users\ashac\anaconda3\lib\site-
packages (0.8.9)
```

In [43]:

```
from tabulate import tabulate

# Function to read text from a file and return the words
def read_text_from_file(file_path):
    with open(file_path, "r", encoding="utf-8") as file:
        text= file.read()
    return text
# Function to clean and tokenize the text into words
def preprocess_text(text):

    # Remove unwanted characters and split the text into words
    words= text.strip().replace("\n", " ").split(" ")
    # Remove any empty strings from the list of words
    words= [word.strip() for word in words if word.strip()]
    return words
# Function to count the word frequency in a list of words
def count_word_frequency(words):
    word_frequency= {}
    for word in words:
        if word in word_frequency:
            word_frequency[word] +=1
        else:
            word_frequency[word] = 1
    return word_frequency
# Function to create a neat table from word frequency dictionary
def create_table(word_frequency):
    table= []
    for word, frequency in word_frequency.items():
        table.append([word, frequency])
    return table
# Read and preprocess
motivational_text= read_text_from_file("tamil quotes.txt")
motivational_quotes_words= preprocess_text(motivational_text)
motivational_quotes_word_frequency= count_word_frequency(motivational_quotes_w

# Read and preprocess
demotivational_quotes_text= read_text_from_file("tamil quotes.txt")
demotivational_quotes_words= preprocess_text(demotivational_quotes_text)
demotivational_quotes_word_frequency= count_word_frequency(demotivational_quot

# Display the word frequency
print
("Word Frequency for motivational Quotes:")
motivational_quotes_table= create_table(motivational_quotes_word_frequency)
print(tabulate(motivational_quotes_table, headers=["Word", "Frequency"], table
# Display the word frequency
print("\nWord Frequency for demotivational Quotes:")
sad_quotes_table = create_table(sad_quotes_word_frequency)
print(tabulate(demotivational_quotes_table, headers=["Word", "Frequency"], tab
```

Word	Frequency
Statement|Label	1
ஒவ்வொரு	1
சிறிய	1
மாற்றமும்	1
பெரிய	1
வெற்றியின்	1
ஒரு	1
பகுதியாகும்|Motivation	1
மனம்	2
துயரத்தின்	1
பின்	1
நிலைக்கும்,	1
வாழ்க்கை	2
மாறிவிடுகின்றது.|Demotivation	1
வெற்றி	2
இலக்கை	1