"TextGen: AI-Powered Text Generation System"

Problem Statement:

- Text generation tasks, such as creating short stories, poems, or news articles, often require substantial time and creativity.
- Human-generated content can be subjective, and it may not always meet the desired quality standards.
- There is a need for an automated text generation system that can produce realistic and high-quality content across various genres.

Program:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


# Sample dataset containing short stories, poems, and news articles
dataset = """
Once upon a time, there was a brave knight named Sir Lancelot. He lived in a castle on the edge of the kingdom, guarding it from all evil.


In a faraway land, there was a magical forest where fairies and elves lived in harmony. But one day, darkness crept into the forest, threatening to destroy everything.
```

```
I wandered lonely as a cloud That floats on high o'er vales and hills, When all at
once I saw a crowd, A host, of golden daffodils;


The woods are lovely, dark and deep, But I have promises to keep, And miles to go
before I sleep, And miles to go before I sleep.


Scientists have discovered a new species of deep-sea fish in the Mariana Trench.
The fish, named Mariana snailfish, thrives in the extreme conditions of the trench.


The stock market experienced a sharp decline today, with major indices dropping
by over 5%. Analysts attribute the downturn to concerns over inflation and
geopolitical tensions.
"""


# Tokenize the text
tokenizer = Tokenizer()
tokenizer.fit_on_texts([dataset])
total_words = len(tokenizer.word_index) + 1


# Generate input sequences
input_sequences = []
for line in dataset.split('\n'):
  token_list = tokenizer.texts_to_sequences([line])[0]
  for i in range(1, len(token_list)):
    n_gram_sequence = token_list[:i+1]
    input_sequences.append(n_gram_sequence)


# Pad sequences
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences,
maxlen=max_sequence_len, padding='pre'))


# Create predictors and label
predictors, label = input_sequences[:,:-1],input_sequences[:,-1]


# One-hot encode the label
label = np.eye(total_words)[label.astype(int)]


# Define the model
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
model.add(LSTM(150, return_sequences=True))
model.add(LSTM(150))
model.add(Dense(total_words, activation='softmax'))


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

```
# Fit the model
model.fit(predictors, label, epochs=100, verbose=1, batch_size=32)


# Generate text
def generate_text(seed_text, next_words, model, max_sequence_len):
  for _ in range(next_words):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1,
padding='pre')
    predicted = np.argmax(model.predict(token_list), axis=-1)
    output_word = ""
    for word, index in tokenizer.word_index.items():
      if index == predicted:
        output_word = word
        break
    seed_text += " " + output_word
  return seed_text


# Example usage
seed_text = "once upon a time"
generated_text = generate_text(seed_text, 10, model, max_sequence_len)
print(generated_text)
```