```
import collections
import random

class SmallLanguageModel:
    def __init__(self):
        self.markov_chain = {}
        self.start_characters = []

    def train(self, text):
        """Train the language model on the given text data."""
        text = text.strip().lower()
        self.start_characters = [sent[0] for sent in text.split('.') if sent.strip()]

        for i in range(len(text) - 1):
            current_char = text[i]
            next_char = text[i + 1]

            if current_char not in self.markov_chain:
                self.markov_chain[current_char] = collections.defaultdict(int)

            self.markov_chain[current_char][next_char] += 1

        for char, next_chars in self.markov_chain.items():
            total_count = sum(next_chars.values())
            for next_char, count in next_chars.items():
                next_chars[next_char] = count / total_count

    def predict_next_character(self, current_char):
        """Predict the next character based on the current character."""
        if current_char not in self.markov_chain:
            return random.choice(self.start_characters)

        next_chars = self.markov_chain[current_char]
        return random.choices(list(next_chars.keys()), weights=list(next_chars.values()), k=1)[0]

    def generate_text(self, length=100):
        """Generate text of the specified length."""
        if not self.markov_chain:
            return "Error: Model not trained yet."

        current_char = random.choice(self.start_characters)
        generated_text = current_char

        for _ in range(length - 1):
            next_char = self.predict_next_character(current_char)
            generated_text += next_char
            current_char = next_char

        return generated_text

# Test the model
if __name__ == "__main__":
    # Sample text for training
    sample_text = """
    The quick brown fox jumps over the lazy dog.
    Sphinx of black quartz, judge my vow.
    Pack my box with five dozen liquor jugs.
    How vexingly quick daft zebras jump!
    The five boxing wizards jump quickly.
    """

    # Create and train the model
    model = SmallLanguageModel()
    model.train(sample_text)

    # Generate and print some text
    print("Generated text:")
    print(model.generate_text(length=200))

    # Print some statistics
    print("\nMarkov Chain Statistics:")
    print(f"Number of unique characters: {len(model.markov_chain)}")
    print("Transition probabilities for 'e':")
    if 'e' in model.markov_chain:
        for next_char, prob in model.markov_chain['e'].items():
            print(f"  e -> {next_char}: {prob:.2f}")
    else:
        print("  'e' not found in the Markov chain.")

    # Test prediction
    print("\nNext character predictions:")
    for char in "thequickbrownfox":
        next_char = model.predict_next_character(char)
        print(f"  After '{char}': '{next_char}'")
```

```
Generated text:
th
  juowive jumy dg. juafthoge  das. brar bly juove br   my. p!
```

```
 jump  ox  ve  he he  owiqump zaz,   juithex qug fown ove lararazebox jumphe juicklardg.   ve w
doxings.   fick five quize my  livero

Markov Chain Statistics:
Number of unique characters: 31
Transition probabilities for 'e':
  e ->  : 0.60
  e -> r: 0.10
  e -> n: 0.10
  e -> x: 0.10
  e -> b: 0.10

Next character predictions:
  After 't': ' '
  After 'h': 'o'
  After 'e': 'r'
  After 'q': 'u'
  After 'u': 'm'
  After 'i': 'z'
  After 'c': 'k'
  After 'k': 'l'
  After 'b': 'o'
  After 'r': 'a'
  After 'o': 'z'
  After 'w': 'i'
  After 'n': 'x'
  After 'f': 't'
  After 'o': 'w'
  After 'x': 'i'
```

```
import collections
import random

class SmallLanguageModel:
    def __init__(self):
        """
        Initialize the SmallLanguageModel.

        This method sets up the necessary variables for the language model.
        """
        self.markov_chain = {}
        self.start_characters = []
```

```
def train(self, text):
    """
    Train the language model on the given text data.

    Args:
        text (str): The input text to train the model on.
    """
    # Store the start characters (first character of each sentence)
    self.start_characters = [sent[0] for sent in text.split('.') if sent]

    # Create a dictionary to store the Markov chain
    for i in range(len(text) - 1):
        current_char = text[i]
        next_char = text[i + 1]

        # If the current character is not in the Markov chain yet, add it
        if current_char not in self.markov_chain:
            self.markov_chain[current_char] = collections.defaultdict(int)

        # Increment the count for the next character
        self.markov_chain[current_char][next_char] += 1

    # Normalize the probabilities for each character
    for char, next_chars in self.markov_chain.items():
        total_count = sum(next_chars.values())
        for next_char, count in next_chars.items():
            next_chars[next_char] = count / total_count

def generate_text(self, length=100):
    """
    Generate text based on the trained Markov chain.

    Args:
        length (int): The desired length of the generated text.

    Returns:
        str: The generated text.
    """
    if not self.markov_chain:
        return "Error: Model not trained yet."

    current_char = random.choice(self.start_characters)
    generated_text = current_char
```

```
        for _ in range(length - 1):
            if current_char not in self.markov_chain:
                # If we reach a character with no followers, start a new sentence
                current_char = random.choice(self.start_characters)
                generated_text += '. ' + current_char
            else:
                next_char = random.choices(
                    list(self.markov_chain[current_char].keys()),
                    weights=list(self.markov_chain[current_char].values())
                )[0]
                generated_text += next_char
                current_char = next_char

        return generated_text

model = SmallLanguageModel()
model.train("Your training text goes here.")
generated_text = model.generate_text(length=200)
print(generated_text)
```

```
Youres terere.. Yoextres heraingourer t tes hes tr hexte.. Yoe.. Yoerainininng tes g t he.. Youres
goer trainingoe.. Your text goe.. Yourainining hext terainingoes hext t tr hext he.. Youre.. Yoes g
text g t g trera
```