深 圳 大 学 实 验 报 告

课程名称：　　　　　随机信号处理

实验项目名称：　　基于马可夫链的文本生成器

学院：　　　　　　电子与信息工程学院

专业：　　　　　　电子信息工程

指导教师：　　　　孙维泽

报告人：　　陈应权　　　学号：　**2022280297**

班级：　　**22** 文华班

实验时间：　**2024** 年 **6** 月 **17** 日——**2024** 年 **6** 月 **29** 日

实验报告提交时间：　**2024** 年 **6** 月 **29** 日

教务处制

Description of format:

- Use Times New Roman, 12 pt, single column, single line spacing.
- When inserting figures and tables, title of the figures and tables must be included.
- Do not change '1、Purposes of the experiment' and '2、Design task and detail requirement'.

1、Purposes of the experiment

1) Use python to count the number of words
2) Use Markov chains to generate random text.
3) Analyze the code and draw reasonable conclusions.

2、Design task and detail requirement

See 'Appendix 1 – Task and requirement for experimental report 4.doc'.

3、The result and Analysis

- **Experiment (100 points):**

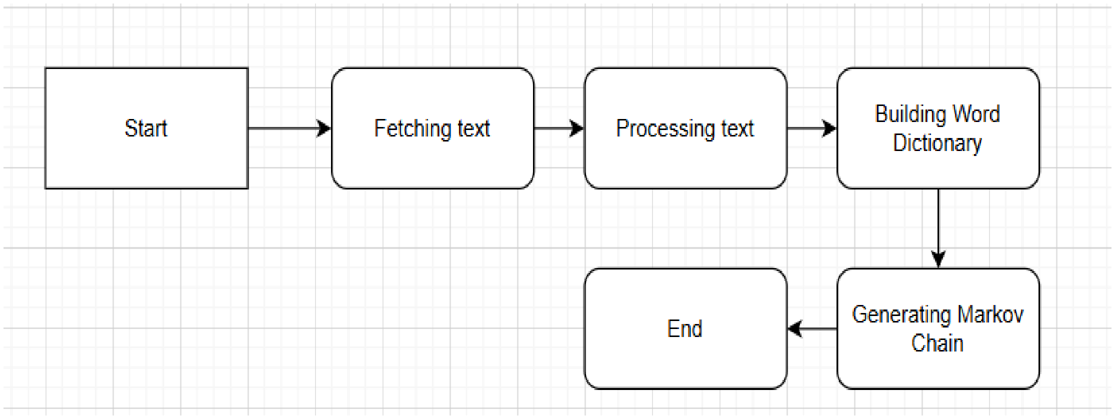
Use a piece of known text to generate a random short text of 100 words using the knowledge of Markov chains. Based on the code provided, write a flowchart and understandings/comments of this code.

Understanding Markov Chains for Text Generation

Basic Concept: A Markov chain is a mathematical model based on probability theory used to describe the random transitions of a system between different states. The key characteristic of a Markov chain is "memorylessness," meaning that the next state of the system depends only on the current state and is independent of the sequence of events that preceded it. This property makes Markov chains very effective for modeling stochastic processes and generating sequences.

Application in Text Generation: When applying Markov chains to text generation, we view text as a sequence of words. By analyzing the structure of the text, we can construct a dictionary that represents the possible following words for each word along with their probabilities. Using this dictionary, we can then generate new text. The specific steps include:

The flow chart:



The flowchart based on code

1) **Start:**
   - Initialize necessary libraries and functions.
2) **Fetching Text:**

   - Fetch text from the URL using urllib.request.urlopen.

3) **Processing Text:**
   - Remove newlines and quotes.
   - Replace punctuation with space-padded versions.
   - Split the text into words.

4) **Building Word Dictionary:**
   - For each word in the text (from the second word to the last word):
     - If the preceding word is not in wordDict, add it as a key with an empty dictionary as its value.
     - If the current word is not in the preceding word's dictionary, initialize its count to 0.
     - If the preceding word is a punctuation mark, increase the current word's count by 3; otherwise, increase it by 1.
5) **Generating Markov Chain:**
   - Initialize the chain with the word "I".
   - For each word in the chain (up to 100 words):
     - Retrieve a new word using the retrieveRandomWord function based on the current word's dictionary.
     - Append the new word to the chain.
6) **Output:**
   - Print the generated Markov chain as a single string.

Code Understanding and Comments:

The provided code is a Python script that generates a random sequence of words using a Markov chain. Here is a detailed understanding and comments of the code:

1) **Importing Libraries:**
    - urllib.request is used to fetch the text from a URL.
    - random.randint is used to generate random numbers.
2) **Function Definitions:**
    - wordListSum(wordList):
        - This function calculates the sum of all values in the given dictionary wordList.
        - sum is initialized to 0 and incremented by each value in the dictionary.
    - retrieveRandomWord(wordList):
        - This function retrieves a random word from the given dictionary wordList based on their frequencies.
        - A random index randIndex is generated between 1 and the sum of all values in the dictionary.
        - The function iterates through the dictionary, subtracting the value from randIndex until randIndex is less than or equal to 0, and returns the corresponding word.
    - buildWordDict(text):
        - This function builds a word dictionary from the given text.
        - Newlines and quotes are removed from the text.
        - Punctuation marks are replaced with space-padded versions.
        - The text is split into words.
        - A dictionary wordDict is created where each word points to another dictionary of words that follow it in the text, with their counts.
        - If the preceding word is a punctuation mark, the count of the following word is increased by 3 (to give punctuation marks higher weight).
3) **Fetching and Processing Text:**
    - The text is fetched from a given URL and converted to a string.
4) **Building Word Dictionary:**
    - The buildWordDict function is called with the fetched text to build the word dictionary wordDict.
5) **Generating Markov Chain:**
    - A Markov chain of 100 words is generated starting with the word "I".
    - For each word, a new word is retrieved using the retrieveRandomWord function based on the current word's dictionary.
    - The new word is appended to the chain.
6) **Printing the Result:**
    - The generated chain of words is printed as a single string.

Comments

- The code uses a Markov chain to generate a random sequence of words based on the input text.
- By giving punctuation marks a higher weight, the code ensures that punctuation plays a key role in sentence breaking, making the generated text more natural and readable.
- The use of random selection ensures that each run of the code generates a different sequence of words, adding an element of variability and creativity to the generated text.
- The wordListSum and retrieveRandomWord functions are crucial for ensuring that the random selection of words is weighted according to their frequencies, making the Markov chain generation more accurate and reflective of the input text.

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。