深 圳 大 学 实 验 报 告

课程名称：Stochastic Signal Processing

实验项目名称：Markov chain experiment

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：廖祖颐 学号：2022110131

班级：文华班

实验时间：2024.6.18

实验报告提交时间：2024.6.20

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.

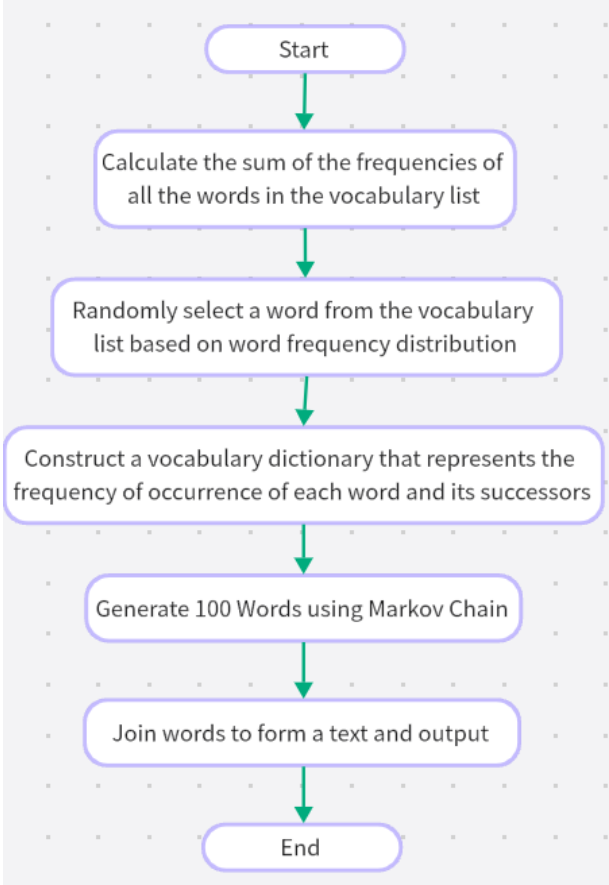


Fig 1 Flowchart

```
# Function to calculate the total sum of the frequencies
def wordListSum(wordList):
    sum = 0
    for word, value in wordList.items():
        sum += value
    return sum
```

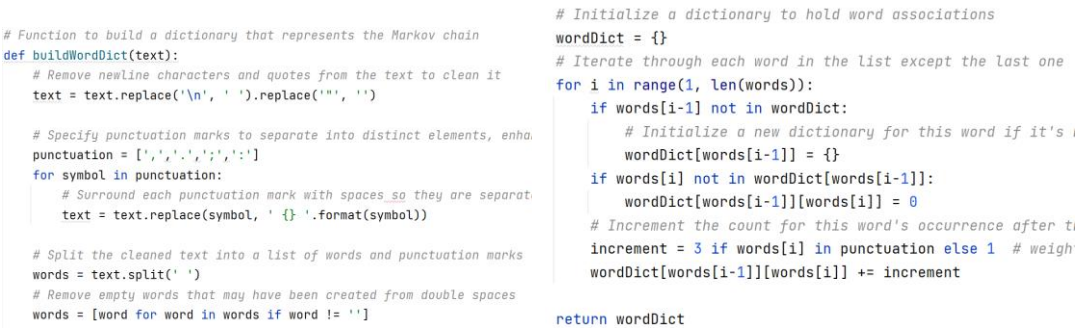
Fig 2 WordListSum function

This function calculates the total sum of frequencies of all words in a given list (a dictionary where keys are words and values are their frequencies). This sum represents the total number of possible transitions from the current word.

```
# Function to select a random word based on the frequency distribution of possibl
def retrieveRandomWord(wordList):
    # Generate a random index up to the sum of all frequencies in the word list
    randIndex = randint(1, wordListSum(wordList))
    for word, value in wordList.items():
        # Decrease randIndex by the frequency count until it is zero or negative
        randIndex -= value
        if randIndex <= 0:
            # Return the word that brings randIndex to zero or below, ensuring se
            return word
```

Fig 3 retrieveRandomWord function

This function selects a random word from the word list, weighted by their frequencies. It generates a random number up to the total frequency sum and iterates through the words, decrementing the random number by each word's frequency until it reaches zero or below. The word that causes this is returned, ensuring that words with higher frequencies are more likely to be chosen.



```
# Function to build a dictionary that represents the Markov chain
def buildWordDict(text):
    # Remove newline characters and quotes from the text to clean it
    text = text.replace('\n', ' ').replace('"', '')

    # Specify punctuation marks to separate into distinct elements, enha
    punctuation = [',','.',';',':']
    for symbol in punctuation:
        # Surround each punctuation mark with spaces so they are separat
        text = text.replace(symbol, ' {} '.format(symbol))

    # Split the cleaned text into a list of words and punctuation marks
    words = text.split(' ')
    # Remove empty words that may have been created from double spaces
    words = [word for word in words if word != '']

    # Initialize a dictionary to hold word associations
    wordDict = {}
    # Iterate through each word in the list except the last one
    for i in range(1, len(words)):
        if words[i-1] not in wordDict:
            # Initialize a new dictionary for this word if it's
            wordDict[words[i-1]] = {}
        if words[i] not in wordDict[words[i-1]]:
            wordDict[words[i-1]][words[i]] = 0
        # Increment the count for this word's occurrence after t
        increment = 3 if words[i] in punctuation else 1  # weigh
        wordDict[words[i-1]][words[i]] += increment

    return wordDict
```

Fig 4 buildWordDict function

This function processes the input text to build a Markov chain dictionary. It performs the following steps:

1. Removes newline characters and quotes for cleaner processing.
2. Surrounds punctuation marks with spaces so that they are treated as separate elements in the word list.
3. Splits the cleaned text into a list of words and punctuation marks.

4. Iterates through the list to build a dictionary (wordDict), where each word is associated with another dictionary of words that can follow it, along with their frequencies. Punctuation marks are given a higher weight (frequency increment of 3) to influence their likelihood in the generated text.

```
# Generate a Markov chain text of 100 words starting with 'I'
length = 100
chain = ['I']
for i in range(0, length):
    # Add a new word by randomly selecting from the dictionary based
    newWord = retrieveRandomWord(wordDict[chain[-1]])
    chain.append(newWord)

# Join the words in the chain to form a coherent string and print it
print(' '.join(chain))
```

Fig 5 Generation text

The code fetches the text from the specified URL. The code initializes the text generation with the starting word 'I'. It then iterates to generate a sequence of words, each chosen based on the previous word using the retrieveRandomWord function. The generated words are appended to the chain list. Finally, the words in the chain are joined into a single string and printed as the output. Here comes the results:

1. I shall find a subject of the principles and maintain their agents , like the great dread of the language of the Constitution the residue of the accumulation in the executive power . If we may regret anything in the United States , if it has gone , or was the legislative branch . Fellow-citizens , to recommend measures to , which has actually granted by the production of party , and a rigid adherence to any of that of the land a misconstruction of power also . This is unable to monarchy , living with these sagacious people . It
2. I conceive , the means which those feelings . It was intended , to realize the people to be worthy representatives , is , that wise legislation of keeping the General Government , it any open warfare may be true--and it is an Administration not , but that the rich are much the Union . In reference to all the State legislatures not only as a duty of every tendency of government which the mutual interests and , for a proper plan for immutable history , and , and that the officer at one presiding over its correction is suffered to
3. You will has never be adopted , they did not insensible of the danger to the Constitution of an express grant , it becomes my opinion may have been always observable that I sincerely believe that the object for talent and common with the people , it behooves the freedom , it . As , could not but that I refer to propitiate an express grant , there was the rights . However deeply we want of many years since , to do not its authors it is and strengthens with the ordinary legislation . There is for one time .

Fig 6 Results of three experiments

Analysis of the text output:

Each generated text sequence retains a semblance of grammatical structure. This is because the Markov chain uses word pairs from the input text, ensuring that transitions between words often make grammatical sense. The use of punctuation is consistent and frequent, reflecting the increased weighting applied to punctuation marks in the word list. This adds a level of authenticity to the generated text, making it appear more like natural language. While each sentence fragment appears grammatically correct, the overall flow of ideas and the logical coherence between sentences are lacking

Conclusion:

The experiment demonstrates the effectiveness of a Markov chain model in generating text that resembles the style of a given input. However, it also highlights the inherent limitations of this approach, particularly in creating globally coherent and contextually rich narratives. For more advanced text generation, higher-order Markov models or more sophisticated approaches like neural networks (e.g., GPT models) would be required to capture the deeper contextual relationships within the 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。