```
import string
import re
import numpy as np
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt


def find_all(text, word):
    """
    Find all occurrences of a word in a text.

    :param text: (str) The text to search
    :param word: (str) The word to find
    :return: (list(int)) List of word indices
    """

    indices = []
    start_index = 0
    while True:
        index = text.find(word, start_index)

        # Break when no more occurrences can be found
        if index == -1:
            break

        # Append index of string and update new starting index for find
        indices.append(index)
        start_index = index + len(word)

    return indices


class Node:
    def __init__(self, data=None):
        """
        Initialize a Node object for a linked list.

        :param data: (any) The data to store in the node
        """

        self.data = data
        self.next = None


class LinkedList:
    def __init__(self):
        """
        Initialize a LinkedList object.
        """

        self.head = Node()

    def append_node(self, data):
        """
        Append a new node to the linked list.
```

```
        :param data: The data to store in the new node
        """

        new_node = Node(data)
        current_node = self.head

        # Find end of linked list
        while current_node.next is not None:
            current_node = current_node.next
        current_node.next = new_node

    def get_list(self):
        """
        Get the elements of the linked list as a list.

        :return: (list(any)) A list of elements of the linked list
        """
        list_elements = []
        current_node = self.head

        # Find end of linked list
        while current_node.next is not None:
            current_node = current_node.next
            list_elements.append(current_node.data)

        return list_elements


class Article:
    def __init__(self, filename):
        """
        Initialize an Article object.

        :param filename: (str) The filename of the text
        """
        self.filename = filename
        self.file = None
        self.text_start = None
        self.text_end = None
        self.words = {}
        self.read_file()
        self.find_start_end()
        self.lemmatizer = WordNetLemmatizer()
        self.populate()

    def read_file(self):
        """
        Read the contents of the file.
        """
        with open(self.filename, 'r') as f:
            self.file = f.read()

    def find_start_end(self):
        """
        Find the start and end indices of articles in text.
```

```
        """

        self.text_start = find_all(self.file, "<text>")
        self.text_end = find_all(self.file, "</text>")

    def populate(self):
        """
        Populate dictionary of words with linked list of word counts.
        """

        for i in range(len(self.text_end)):
            text = self.file[self.text_start[i] + 5:self.text_end[i] - 1]

            # Remove html tags
            text = re.sub('<[^<]+?>', '', text)

            # Remove punctuation and set convert to lowercase
            text = text.translate(str.maketrans('', '', string.punctuation))
            text = text.lower()

            # Lemmatize (root of word) words in article and count unique words
            words = text.split()
            words = [self.lemmatizer.lemmatize(word, pos='v')
                     for word in words]
            unique_words, counts = np.unique(words, return_counts=True)

            # Add article number to counts
            counts = [[i + 1, count] for count in counts]

            # Zip words to counts
            word_count = dict(zip(unique_words, counts))

            # Add word count as node, initialize linked list if word not found
            for word in unique_words:
                if hash(word) not in self.words.keys():
                    self.words[hash(word)] = LinkedList()
                self.words[hash(word)].append_node(word_count[word])

    def plot(self):
        """
        Plot the distribution of the word counts.
        """
        frequencies = []
        for key in self.words:
            for element in self.words[key].get_list():
                frequencies.append(element[1])

        plt.hist(frequencies, bins=len(self.words))
        plt.xlabel("Frequency of Word in Collection")
        plt.ylabel("Frequency of Count")
        plt.title("Count Distribution of Words in Collection")


def main():
    collection = Article('collection.txt')
    collection.plot()
```

```
    plt.show()


if __name__ == "__main__":
    main()
```