```
require 'nokogiri'
require 'open-uri'
require 'lemmatizer'
require 'histogram/array'
require 'chartkick'
include Chartkick::Helper

# ruby doesn't have a native implementation of linked lists
class Node
  attr_accessor :val, :next

  def initialize(val, next_node)
      @val = val
      @next = next_node
  end
end

class LinkedList

  def initialize(val)
    @head = Node.new(val, nil)
  end

  def add(val)
    current = @head
    while current.next != nil
      current = current.next
    end
    current.next = Node.new(val, nil)
  end

  def return_list
    elements = []
    current = @head
    while current.next != nil
      elements << current
      current = current.next
    end
    elements << current
  end

  def return_values
    return_list.map{|n| n.val}
  end
end

class WordCounter
  attr_accessor :source_url, :text_source, :lem, :preprocess_storage, :wor
d_count, :articles_with_counts, :word_count_table, :graph_data

  def initialize
    @source_url =
"http://gss.uva.nl/binaries/content/assets/programmas/information-
studies/txt-for-assignment-data-science.txt?3015083536432"
```

```
    @text_source = "./txt-for-assignment-data-science.txt"
    @lem = Lemmatizer.new
    @preprocess_storage = {}
    @articles_with_counts = {}
    @word_count_table = {}
    retrieve_tokenize_and_lemmatize
    @graph_data = format_for_plotting(histogram_data_for_collection)
  end

  def retrieve_tokenize_and_lemmatize
    doc = Nokogiri::HTML(open(@text_source))
    doc.search('text').each_with_index do |link, index|
      # simple white space tokenizer with ruby regex sufficient
      tokenized_text = link.content.scan(/\w+/)
      @preprocess_storage[index] = tokenized_text.map{ |token|
@lem.lemma(token.downcase) }
    end
  end

  def count(article_tokens)
    article_tokens.each_with_object({}) do |token, article_word_count|
      article_word_count[token] ||= 0
      article_word_count[token] += 1
    end
  end

  def count_tokens_by_article
    @preprocess_storage.each do |article_id, tokenized_article|
      @articles_with_counts[article_id + 1] = count(tokenized_article)
    end
  end

  def count_tokens_by_collection
    count(@preprocess_storage.values.flatten)
  end

  def all_frequencies_descending
    count_tokens_by_collection.values.sort.reverse
  end

  def article_counts_by_word
    @articles_with_counts.each do |article_id, counts_by_word|
      counts_by_word.each do |word, count|
        if @word_count_table[word] == nil
          @word_count_table[word] = LinkedList.new([article_id, count])
        else
          @word_count_table[word].add([article_id, count])
        end
      end
    end
  end

  def histogram_data_for_collection
    collection_counts = count_tokens_by_collection.values
```

```
    (bins, freq) =
collection_counts.histogram(collection_counts.uniq.sort)
  end

  def format_for_plotting(histogram_arrays)
    frequency_count_hash = {}
    count_array = histogram_arrays[0]
    frequency_array = histogram_arrays[1]
    count_array.each_with_index do |count, index|
      frequency_count_hash[count.to_i] = frequency_array[index].to_i
    end
    frequency_count_hash
  end

  def plot_histogram
    graph_html = column_chart @graph_data, xtitle: 'Word Count', ytitle:
'Count Frequency'
    open('frequency_count_plot.html', 'w+') do |f|
      f.puts '<script src="https://www.google.com/jsapi"></script>'
      f.puts '<script src="chartkick.js"></script>'
      f.puts graph_html
    end
  end

  def count_source
    count_tokens_by_article
    article_counts_by_word
    @word_count_table
  end
end
```