```ruby
#Zach Danial
# This document contains the source code for a simple learning algorithm for sentiment analysis
# of the Reuters Business RSS feed that improves as it is used.

# The algorithm initially runs on a data set comprised of amazon reviews, imdb reviews, and yelp reviews
# that are labelled with their rating.  The reviews are broken down by the 'sentiment1' file and word
# data is placed inside of a text file where it is accesed and updated as the algorithm encounters old
# and new words.

# The algorithm removes the top 100 most used English words from the analysis to more accurately
# reflect the sentiment of a sentence.

# Word data is stored in the text file 'learning flow'

# import requirements
require 'rss'
require 'open-uri'


$new_words = {}

# Book class is a database of each word, the number of times the algorithm has seen the word,
# and the total score of the word
class Book
  @@page = 0
  @@number_of_files = 0
  def self.increment_num
    @@number_of_files += 1
  end
  def self.num
    return @@number_of_files
  end
  def self.next_page
    @@page += 1
  end
  def self.prev_page
    @@page -=1
  end
  def self.wrap
    @@page = 0
  end
  def self.backwrap
    @@page = $keys.length - 1
```

```
44    end
45    def self.page
46      return @@page
47    end
48    def self.all
49      ObjectSpace.each_object(self).to_a
50    end
51
52    def initialize(title)
53      @title = title
54      @story = $data[title][0]
55      @time = $data[title][1]
56
57    end
58
59    def time
60      return @time
61    end
62    def title
63      return @title
64    end
65    def story
66      return @story
67    end
68
69    # Analyzes the words of the title of an article and its
   story, as well as cumulative score
70    def title_wordsanalyzed
71      @title_wordsanalyzed = {}
72      @title_total = 0
73      broken(@title).each do |word|
74        word = word.downcase
75        if (word.in $words) and word.alpha?
76          begin
77            @title_total += $words[word][2]
78            @title_wordsanalyzed[word] = $words[word][2]
79          rescue
80              next
81          end
82        end
83      end
84      return @title_wordsanalyzed
85    end
86    def title_score
87      if @title_wordsanalyzed.keys.length > 0
88        @title_score = (@title_total/@title_wordsanalyzed.
   keys.length).to_f
89        return @title_score
90      else
91        return 'Incomplete Analysis'
```

```
 92      end
 93    end
 94
 95    def story_wordsanalyzed
 96      @story_wordsanalyzed = {}
 97      @story_total = 0
 98      broken(@story).each do |word|
 99        word = word.downcase
100        if word.alpha? and (word.in $words)
101          begin
102          @story_total += $words[word][2]
103          @story_wordsanalyzed[word] = $words[word][2]
104          rescue
105            next
106          end
107
108        end
109      end
110      return @story_wordsanalyzed
111    end
112    def story_score
113      if @story_wordsanalyzed.keys.length > 0
114        @story_score = (@story_total/@story_wordsanalyzed.
    keys.length).to_f
115        return @story_score
116      else
117        return 'Incomplete Analysis'
118      end
119    end
120
121    # Method to update 'learning flow'
122    def edit
123      title_score = @title_score
124      title_words = @title_wordsanalyzed
125      story_score = @story_score
126      story_words = @story_wordsanalyzed
127      title_words.each_key do |key|
128        word = key.downcase
129        if word.alpha? and word.in $words
130          $words[word][0]+=title_score
131          $words[word][1]+=1
132        end
133      end
134      story_words.each_key do |key|
135        word = key.downcase
136        if word.alpha? and word.in $words
137          $words[word][0]+=story_score
138          $words[word][1]+=1
139        end
140      end
```

```
141     $anti_words.each do |word|
142       $words.delete(word.chomp)
143     end
144     $words.each_key do |key|
145       if not key.alpha?
146         $words.delete(key)
147       end
148     end
149   end
150   def new_words
151     title_score = @title_score
152     title_words = broken(@title)
153     story_score = @story_score
154     story_words = broken(@story)
155     title_words.each do |key|
156       word = key.downcase
157       if ((not word.in $words) and word.alpha?)
158         if word.in $new_words
159           $new_words[word][0]+=title_score
160           $new_words[word][1]+=1
161         elsif not word.in $new_words
162           $new_words[word] = [title_score,1]
163         end
164       end
165     end
166
167     story_words.each do |key|
168       word = key.downcase
169       if ((not word.in $words) and word.alpha?)
170         if word.in $new_words
171           begin
172             $new_words[word][0]+=story_score
173             $new_words[word][1]+=1
174           rescue
175             puts word
176             puts story_score
177             $new_words[word][0]
178           end
179
180         elsif not word.in $new_words
181           $new_words[word] = [story_score.to_f,1]
182         end
183       end
184     end
185     $new_words.each do |key, array|
186       $words[key] = array
187     end
188   end
189
190
```

```
191   # Method to streamline analysis and updating
192   def full_analysis
193     self.title_wordsanalyzed
194     self.title_score
195     self.story_wordsanalyzed
196     self.story_score
197     self.edit
198     self.new_words
199   end
200   def print_out
201     title_out = "\n"+(@title_score.round(3)*10).to_s+"\n"+
    line_break(@title)+"\n"
202     story_out = (10*@story_score).round(3).to_s + " - "+
    line_break(@story)
203     stop = "\n\n-------------------------------------\n"+(
    "Words Analyzed: \n")
204     out = title_out + "\n"+ story_out + stop
205     @title_wordsanalyzed.each_pair do |key, val|
206       out += key.upcase + "\t\t\t"+(10*val).round(2).to_s+
    "\n"
207     end
208     out += "\n\n"
209     @story_wordsanalyzed.each_pair do |key, val|
210       out += key.upcase + "\t\t\t"+(10*val).round(2).to_s+
    "\n"
211     end
212     puts out
213   end
214
215 end
216
217 # Some convinient methods
218 class Object
219   def in(ary)
220     ary.include?(self)
221   end
222 end
223 class String
224   def alpha?
225     !!match(/^[[:alpha:]]+$/)
226   end
227
228   def alnum?
229     !!match(/^[[:alnum:]]+$/)
230   end
231 end
232
233 def find(string,item)
234   i = 0
235   string.each_char do |letter|
```

```
236     if letter == item
237       return i
238     end
239     i+=1
240   end
241 end
242
243 def spaced_out(string)
244   string = string.to_s
245   out = ''
246   string.each_char do |letter|
247     if letter.alpha? or letter == " "
248       out += letter
249     else
250       out += ' '+letter+' '
251     end
252   end
253   return out
254 end
255 def broken(string)
256   return spaced_out(string).split(' ')
257 end
258 def line_break(string)
259   out = ''
260   count = 0
261   string=string.to_s
262   string.each_char do |letter|
263     out += letter
264     if letter == ' '
265       count+=1
266     end
267     if count == 9
268       out += "\n"
269       count =0
270     end
271   end
272   return out
273 end
274
275 def load_step_3(file, dict)
276   cache = open(file,'r').readlines
277   for i in (0..(cache.length)).step(3)
278     word = cache[i].to_s.chomp
279     score = cache[i+1].to_f
280     count = cache[i+2].to_f
281     if count != 0
282       dict[word]= [score.to_f, count.to_i, (score/count).
    to_f]
283     end
284   end
```

```
285
286 end
287 $words = {}
288 load_step_3('sentiment4.txt',$words)
289 puts $words['on']
290 def delete_anti_words(from, with)
291   $anti_words = []
292   antis = open(with, 'r').readlines
293   antis.each do |word|
294     $anti_words+=[word]
295   end
296
297   $anti_words.each do |word|
298     from.delete(word)
299   end
300   $words.each_key do |word|
301     if word.length<=1
302       $words.delete(word)
303     end
304   end
305 end
306 delete_anti_words($words,'100words.txt')
307
308
309 # Grabs Reuters RSS Feed
310 def grab_feed
311   $data = {}
312   url = 'http://feeds.reuters.com/reuters/businessNews?
    format=xml'
313   open(url) do |rss|
314     feed = RSS::Parser.parse(rss)
315     feed.items.each do |item|
316       stop = find(item.description,'<')
317       $data[item.title] = [item.description[0..stop-1],
    item.pubDate]
318     end
319   end
320 end
321 grab_feed
322 $keys = $data.keys
323
324 def for_python
325   current = open('current.txt','w')
326   storage = open('for_python.txt','a')
327   Book.all.each do |book|
328     current.write(book.time.to_s+','+book.title_score.to_s
    +','+book.story_score.to_s+"\n")
329     storage.write(book.time.to_s+','+book.title_score.to_s
    +','+book.story_score.to_s+"\n")
330   end
```

```
331   current.close
332   storage.close
333 end
334
335 def full_start
336   $keys.each do |key|
337     old_key = open('keys.txt','r').readlines
338     $old_keys = []
339     old_key.each do |key|
340       $old_keys += [key.chomp]
341     end
342     if key.in $old_keys
343       Book.increment_num
344       instance_variable_set("@page#{Book.num}",Book.new(
    key))
345       puts (100*Book.num/$keys.length).to_s + '%'
346     elsif not key.in $old_keys
347       Book.increment_num
348       instance_variable_set("@page#{Book.num}",Book.new(
    key))
349       puts (100*Book.num/$keys.length).to_s + '%'
350     end
351   end
352 end
353 def ending
354   Book.all.each do |book|
355     book.full_analysis
356   end
357   $anti_words.each do |word|
358     $words.delete(word.chomp)
359   end
360   $words.each_key do |key|
361     if not key.alpha?
362       $words.delete(key)
363     end
364   end
365   out = open('learning flow', 'w')
366   $words.each do |key, array|
367     out.write(key.downcase+"\n")
368     out.write(array[0].to_s+"\n")
369     out.write(array[1].to_s+"\n")
370   end
371   out.close
372   down_keys = open('keys.txt','w')
373   $keys.each do |title|
374     down_keys.write(title+"\n")
375   end
376   down_keys.close
377   $keys = $keys.reverse
378   puts 'done'
```

```
379   for_python
380 end
381 full_start
382 ending
383
384
385
386 # Output for reading
387 puts "\nWelcome to Reuters RSS Feed Sentiment Analysis!\n\
    n"
388 while true
389   puts 'Page ' + (Book.page+1).to_s + ' / ' + Book.num.
    to_s
390   Book.all[Book.page].print_out
391   puts '( < , > ) or page number:'
392   opt = gets.chomp
393   if opt == '>'
394     Book.next_page
395     if Book.page == $keys.length
396       Book.wrap
397     end #wrap
398   elsif opt == '<'
399     Book.prev_page
400     if Book.page<0
401       Book.backwrap
402     end #backwrap
403   end #Page changing
404   puts "\n
    ----------------------------------------------\n\n"
405 end
406
```