interactive.prl

```
#!/usr/local/bin/perl

use Carp;
use FileHandle;

my $DIR = "/home/pan/test/hw2";

my $file_name      = "interactive";
my $raw_file       = "$file_name\.raw";
my $stemmed_file   = "$file_name\.stemmed";
my $tokened_file   = "$file_name\.tokenized";
my $stem_hist_file = "$file_name\.stemmed\.hist";
my $tokn_hist_file = "$file_name\.tokenized\.hist";

print "\n\nINTERACTIVE QUERY:\n\n";
print "Please enter your query.  Press [Enter] on a line\n";
print "by itself to finish entering your query:\n\n";

$query_text ="";
$line_no = 1;

print "Query (1):  ";
while(($curr_line = <STDIN>) ne "\n") {
  $curr_line =~ s/^\s+//;
  $curr_line =~ s/\s+$//;
  $query_text=$query_text.$curr_line;
  $line_no++;
  print "Query ($line_no):  "
}
chomp $query_text;


print "\n\nPlease enter the names of any authors you wish to search\n";
print "for, one per line.  Press [Enter] on a line by itself when\n";
print "you're finished:\n\n";


$author_text ="";
$line_no = 1;

print "Author (1):  ";
while(($curr_line = <STDIN>) ne "\n") {
  $curr_line =~ s/^\s+//;
  $curr_line =~ s/\s+$//;
  $author_text=$author_text.$curr_line;
  $line_no++;
  print "Author ($line_no):  "
}
chomp $author_text;


# Now we need to save the query to the raw file, and run the necessary
# tokenizing tools on this file.

$lst_query_idx = 1;

print "\n\nSaving query to 'interactive.raw'\n";

open(RAWFILE, ">$raw_file");

print RAWFILE "\.I $lst_query_idx\n";

if($query_text ne "") {
  print RAWFILE ".W\n";
  print RAWFILE "$query_text\n";
}
if($author_text ne "") {
  print RAWFILE ".A\n";
  print RAWFILE "$author_text\n";
}

close(RAWFILE);

my $stem_com = "$DIR/stemmer/nstemmer $raw_file > $stemmed_file";
# my $tokn_com = "$DIR/stemmer/nstemmer $raw_file > $tokened_file";

my $stem_hist_com = "cat $stemmed_file | perl $DIR/make_hist.prl > $stem_hist_file";
my $tokn_hist_com = "cat $tokened_file | perl $DIR/make_hist.prl > $tokn_hist_file";
#my $stem_hist_com = "$DIR/make_hist.prl > $stem_hist_file";
#my $tokn_hist_com = "$DIR/make_hist.prl > $tokn_hist_file";

print "Tokenizing and stemming query.\n";

system ("$stem_com") and die "Failed $DIR/stemmer/nstemmer: $!\n";
# system ("$tokn_com") and die "XFailed $DIR/tokenize: $!\n";
&tokenize_lc;

print "Making histogram of the query.\n\n";

```

```
90    system ("$stem_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
91    system ("$tokn_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
92
93    ###################################
94    ## Start generate the bigram files ##
95    ##                                 ##
96    ###################################
97
98
99    # Initialize the stopword list for both stemmed and tokenlized
100   my $stoplist   = "$DIR/common_words";
101   my $stoplist_stemmed   = "$DIR/common_words\.stemmed";
102   my %stoplist_hash = ();
103   my %stoplist_stemmed_hash = ();
104
105   # Initialize the tokenlized stopword list
106   my $stoplist_fh   = new FileHandle $stoplist  , "r"
107           or croak "Failed [stoplist_fh] $stoplist";
108
109   while (defined( $line = <$stoplist_fh> )) {
110           chomp $line;
111           $stoplist_hash{ $line } = 1;
112   }
113
114   # Initialize the stemmed stopword list
115   my $stoplist_stemmed_fh   = new FileHandle $stoplist_stemmed  , "r"
116           or croak "Failed [stoplist_fh] $stoplist_stemmed";
117
118   while (defined( $line = <$stoplist_stemmed_fh> )) {
119           chomp $line;
120           $stoplist_stemmed_hash{ $line } = 1;
121   }
122
123   &gen_bigrams($file_name);
124
125   exit(0);
126
127
128   ########################################################
129   ## TOKENLIZE_LC
130   ## Self implemented tokenizer, because the provided shelled
131   ## one won't work ... (failed on calling token1)
132   ##
133   ## My tokenlize_lc will token the queries as exactly the same
134   ## as the original one
135   ########################################################
136   sub tokenize_lc {
137     my $token_qrys_fh = new FileHandle $raw_file, "r"
138           or croak "Failed $token_intr";
139     # open interactive.tokenized to write
140     open(RAWFILE, ">$tokened_file");
141     my @token_ary = (); # the array to store the token in the query
142
143     while (defined( $word = <$token_qrys_fh> )) {
144
145           chomp $word;
146
147           if ($word =~ /^\.I/) {   # start of query tokens
148             print RAWFILE "$word\n";
149             next;
150           }
151
152           if ($word =~ /^\.W/) {   # start of query tokens
153             print RAWFILE "$word\n";
154             next;
155           }
156
157           if ($word =~ /^\.A/) {   # start of query tokens
158             print RAWFILE "$word\n";
159             next;
160           }
161
162           push (@token_ary, split(/\s+/,$word));
163           # write all the tokens in to file
164           foreach $token (@token_ary) {
165             # uncomment this to have all tokens lowercased
166             # the query.tokenized didn't actually converted to lowercase
167             # so I will not doing so either
168
169             # $token = lc($token);
170
171             # split on each token that contains chars like -_,. etc.
172             my @sub_token = split(/([^A-Za-z0-9])/,$token);
173             if($#sub_token == 0) {
174               print RAWFILE "$token\n";
175             }
176             else {
177               foreach $sub_t (@sub_token) {
178                 print RAWFILE "$sub_t\n";
179               }
180             }
```

```
181             }
182             @token_ary = ();
183
184       }
185
186       close(RAWFILE);
187     }
188
189     ########################################################
190     ## GEN_BIGRAMS
191     ##
192     ## This function will generate the bigrams term sets
193     ########################################################
194     sub gen_bigrams {
195
196             my $file_name = shift;
197
198             # array to store all the bigrams as value
199             my @bigrams = ();
200             my $last_word = undef;  # used to store the last word so we could have pairs
201
202             # my $file_name      = "interactive";
203             my $raw_file       = "$file_name\.raw";
204             my $stemmed_file   = "$file_name\.stemmed";
205             my $tokened_file   = "$file_name\.tokenized";
206             my $stem_hist_file = "$file_name\.stemmed\.hist";
207             my $tokn_hist_file = "$file_name\.tokenized\.hist";
208             my $stem_hist_com = "cat $stemmed_file\.bigrams | perl $DIR/make_hist.prl > $stem_hist_file\.bigrams";
209             my $tokn_hist_com = "cat $tokened_file\.bigrams | perl $DIR/make_hist.prl > $tokn_hist_file\.bigrams";
210
211             my $tokened_file_fh = new FileHandle $tokened_file, "r"
212                     or croak "Failed $tokened_file";
213
214             my $stemmed_file_fh = new FileHandle $stemmed_file, "r"
215                     or croak "Failed $stemmed_file";
216
217
218             #########
219             # make the stemmed bigrams
220             #########
221             open(RAWFILE, ">$stemmed_file\.bigrams");
222
223             while (defined( $word = <$stemmed_file_fh> )) {
224                     chomp $word;
225
226                     # if we encounter a normal word, we check if it eligible for bigram
227                     if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {   # start of query tokens
228                       # if both words are not in the stoplist, we added to the bigrams array
229                       if (! exists $stoplist_stemmed_hash{ $last_word } and ! exists $stoplist_stemmed_hash{ $word }) {
230                             push (@bigrams, "$last_word+$word");
231                       }
232                     }
233
234                     # if next doc/query, pop all the elements in the array into file
235                     if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
236                             # after we scanned each doc/qry, all the bigrams are in the array now
237                             # we append all the bigrams to the end of doc/qry
238                             foreach $bigram (@bigrams) {
239                                     print RAWFILE "$bigram\n";
240                             }
241                             # clean up the array
242                             @bigrams = ();
243                     }
244
245                     # we shift last_word by 1 and write the current word into the output file
246                     $last_word = $word;
247                     print RAWFILE "$word\n";
248             }
249
250             # do it once again to make sure we have everything write into the file
251             if ($#bigrams != -1) {
252                     foreach $bigram (@bigrams) {
253                             print RAWFILE "$bigram\n";
254                     }
255             }
256
257             # close the bigram file after we created it
258             close(RAWFILE);
259
260             # generate hist file
261             system ("$stem_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
262
263             # clean up the array
264             @bigrams = ();
265
266             #########
267             # make the tokenlized bigrams
268             #########
269             open(RAWFILE, ">$tokened_file\.bigrams");
270
271             while (defined( $word = <$tokened_file_fh> )) {
```

```
272                     chomp $word;
273
274                     # if we encounter a normal word, we check if it eligible for bigram
275                     if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {    # start of query tokens
276                       # if both words are not in the stoplist, we added to the bigrams array
277                       if (! exists $stoplist_hash{ $last_word } and ! exists $stoplist_hash{ $word }) {
278                             push (@bigrams, "$last_word+$word");
279                       }
280                     }
281
282                     # if next doc/query, pop all the elements in the array into file
283                     if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
284                             # after we scanned each doc/qry, all the bigrams are in the array now
285                             # we append all the bigrams to the end of doc/qry
286                             foreach $bigram (@bigrams) {
287                                     print RAWFILE "$bigram\n";
288                             }
289                             # clean up the array
290                             @bigrams = ();
291                     }
292
293                     # we shift last_word by 1 and write the current word into the output file
294                     $last_word = $word;
295                     print RAWFILE "$word\n";
296             }
297
298             # do it once again to make sure we have everything write into the file
299             if ($#bigrams != -1) {
300                     foreach $bigram (@bigrams) {
301                             print RAWFILE "$bigram\n";
302                     }
303             }
304
305             # close the bigram file after we created it
306             close(RAWFILE);
307
308             # generate hist file
309             system ("$tokn_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
310
311             # clean up the array
312             @bigrams = ();
313     }
```