interactive.prl

```perl
#!/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.

$1st_query_idx = 1;

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

open(RAWFILE, ">$raw_file");

print RAWFILE "\.I $1st_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";

```

```perl
system ("$stem_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
system ("$tokn_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";

####################################
## Start generate the bigram files ##
##                                ##
####################################


# Initialize the stopword list for both stemmed and tokenlized
my $stoplist   = "$DIR/common_words";
my $stoplist_stemmed   = "$DIR/common_words\.stemmed";
my %stoplist_hash = ();
my %stoplist_stemmed_hash = ();

# Initialize the tokenlized stopword list
my $stoplist_fh   = new FileHandle $stoplist  , "r"
        or croak "Failed [stoplist_fh] $stoplist";

while (defined( $line = <$stoplist_fh> )) {
        chomp $line;
        $stoplist_hash{ $line } = 1;
}

# Initialize the stemmed stopword list
my $stoplist_stemmed_fh   = new FileHandle $stoplist_stemmed  , "r"
        or croak "Failed [stoplist_fh] $stoplist_stemmed";

while (defined( $line = <$stoplist_stemmed_fh> )) {
        chomp $line;
        $stoplist_stemmed_hash{ $line } = 1;
}

&gen_bigrams($file_name);

exit(0);


######################################################
## TOKENLIZE_LC
## Self implemented tokenizer, because the provided shelled
## one won't work ... (failed on calling token1)
##
## My tokenlize_lc will token the queries as exactly the same
## as the original one
######################################################
sub tokenize_lc {
  my $token_qrys_fh = new FileHandle $raw_file, "r"
        or croak "Failed $token_intr";
  # open interactive.tokenized to write
  open(RAWFILE, ">$tokened_file");
  my @token_ary = (); # the array to store the token in the query

  while (defined( $word = <$token_qrys_fh> )) {

        chomp $word;

        if ($word =~ /^\.I/) {   # start of query tokens
          print RAWFILE "$word\n";
          next;
        }

        if ($word =~ /^\.W/) {   # start of query tokens
          print RAWFILE "$word\n";
          next;
        }

        if ($word =~ /^\.A/) {   # start of query tokens
          print RAWFILE "$word\n";
          next;
        }

        push (@token_ary, split(/\s+/,$word));
        # write all the tokens in to file
        foreach $token (@token_ary) {
          # uncomment this to have all tokens lowercased
          # the query.tokenized didn't actually converted to lowercase
          # so I will not doing so either

          # $token = lc($token);

          # split on each token that contains chars like -_,. etc.
          my @sub_token = split(/([^A-Za-z0-9])/,$token);
          if($#sub_token == 0) {
            print RAWFILE "$token\n";
          }
          else {
            foreach $sub_t (@sub_token) {
              print RAWFILE "$sub_t\n";
            }
          }
        }
        @token_ary = ();

  }

  close(RAWFILE);
}

########################################################
## GEN_BIGRAMS
##
## This function will generate the bigrams term sets
#######################################################
sub gen_bigrams {

        my $file_name = shift;

        # array to store all the bigrams as value
        my @bigrams = ();
        my $last_word = undef;  # used to store the last word so we could have pairs

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

        my $tokened_file_fh = new FileHandle $tokened_file, "r"
                or croak "Failed $tokened_file";

        my $stemmed_file_fh = new FileHandle $stemmed_file, "r"
                or croak "Failed $stemmed_file";


        #########
        # make the stemmed bigrams
        #########
        open(RAWFILE, ">$stemmed_file\.bigrams");

        while (defined( $word = <$stemmed_file_fh> )) {
                chomp $word;

                # if we encounter a normal word, we check if it eligible for bigram
                if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {   # start of query tokens
                  # if both words are not in the stoplist, we added to the bigrams array
                  if (! exists $stoplist_stemmed_hash{ $last_word } and ! exists $stoplist_stemmed_hash{ $word }) {
                        push (@bigrams, "$last_word+$word");
                  }
                }

                # if next doc/query, pop all the elements in the array into file
                if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
                        # after we scanned each doc/qry, all the bigrams are in the array now
                        # we append all the bigrams to the end of doc/qry
                        foreach $bigram (@bigrams) {
                                print RAWFILE "$bigram\n";
                        }
                        # clean up the array
                        @bigrams = ();
                }

                # we shift last_word by 1 and write the current word into the output file
                $last_word = $word;
                print RAWFILE "$word\n";
        }

        # do it once again to make sure we have everything write into the file
        if ($#bigrams != -1) {
                foreach $bigram (@bigrams) {
                        print RAWFILE "$bigram\n";
                }
        }

        # close the bigram file after we created it
        close(RAWFILE);

        # generate hist file
        system ("$stem_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";

        # clean up the array
        @bigrams = ();

        #########
        # make the tokenlized bigrams
        #########
        open(RAWFILE, ">$tokened_file\.bigrams");

        while (defined( $word = <$tokened_file_fh> )) {
                chomp $word;

                # if we encounter a normal word, we check if it eligible for bigram
                if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {    # start of query tokens
                  # if both words are not in the stoplist, we added to the bigrams array
                  if (! exists $stoplist_hash{ $last_word } and ! exists $stoplist_hash{ $word }) {
                        push (@bigrams, "$last_word+$word");
                  }
                }

                # if next doc/query, pop all the elements in the array into file
                if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
                        # after we scanned each doc/qry, all the bigrams are in the array now
                        # we append all the bigrams to the end of doc/qry
                        foreach $bigram (@bigrams) {
                                print RAWFILE "$bigram\n";
                        }
                        # clean up the array
                        @bigrams = ();
                }

                # we shift last_word by 1 and write the current word into the output file
                $last_word = $word;
                print RAWFILE "$word\n";
        }

        # do it once again to make sure we have everything write into the file
        if ($#bigrams != -1) {
                foreach $bigram (@bigrams) {
                        print RAWFILE "$bigram\n";
                }
        }

        # close the bigram file after we created it
        close(RAWFILE);

        # generate hist file
        system ("$tokn_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";

        # clean up the array
        @bigrams = ();
}
```