Information Retrieval **Chang Liu – cliu70**
Homework 2 **chang.liu@jhu.edu**

Part 3 extensions:

9. Implemented the interactive query so that user could enter their own query in the console. My program will then search the whole doc set and return the document list ordered by similarity. Each time user enter a query, I will dynamically convert the inputted raw query into both tokenized term set (use own implemented tokenizer) and stemmed term set (use nstemmer), this could be used in conjunction with another extra extension I made so that my interactive query function will work under both stemmed and unstemmed initialization which chose by the user at the program startup.

4. Augment the term set to include all bigrams in the document/query that do not contain stopwords. Each time a user chose to use bigrams set at the program initialization, my program will scan through both the stemmed and tokenized term set, add the consecutive words pairs in the set that are both not in the stopword list (common_word.stemmed and common_word) into an array. Then store the pair set into a new .bigrams file alone with the original set and use this new file as the term set. ***make_hist.prl*** will then calculate the frequency based on this set and store them in a new file named .hist.bigrams and use it for the further calculation.

Extra extensions:

Implemented *tokenize* in perl: located in ***interactive.prl***. This function is will convert the raw query into the tokenized format, which did exactly the same job as the provided tokenize and token1. After conversion, it will store all the results in the file. I used this function to dynamically convert my interactive query into tokens.

Implemented all the parameters permutation: At the program startup, user could specify any special parameter permutations they want to use. The program will then run using the specified parameters.

gen_bigrams.prl: This script is called when user chose to use term set that includes bigrams. It will generate a new term set file that contains both the original term set and bigram set.

Print out the full precision / recall table in part 2: My program can print out all the parameter permutations and their precision/recall at one place, which will be convenient when we need to compare performance on each permutation.

Show relevant doc: Implemented the SHOW_RELVNT function so that my program will print out all the starred (relevant) documents for a given query.

```
Part 2 Results

With permutation 3a):

1.


   ************************************************************
  Documents Most Similar To Query number 6
   ************************************************************
   Similarity   Doc#  Author      Title
   ==========   ====  ========    ==============================

*   0.10953065   2828 Clark        #  Hierarchical Geometric Mod
    0.10028232   2753 Pfefferkorn  #  A Heuristic Problem Solvin
    0.08859320   3035 Wetherbe     #  A Strategic Planning Metho
    0.07897459   2087 Pager        #  A Number System for the Pe
    0.07824874   2389 Eastman      #  Preliminary Report on a Sy
*   0.07657006   1543 Howard       #  Computer Formulation of th
    0.07470506   2721 Claudson     #  The Digital Simulation of
    0.07073989   2187 Amarel       #  Computer Science: A Concep
    0.06121554   2230 Bracchi      #  A Language for Treating Ge
    0.05876586   695  Carlson      #  Use of the Disk File on St
    0.05797305   2671 Stone        #  A Note on a Combinatorial
    0.05753394   2836 Loui         #  Weighted Derivation Trees
    0.05558877   2505 Roy          #  Reflection-Free Permutatio
    0.05352421   2707 Bitner       #  Backtrack Programming Tech
    0.05109013   605  --           #  Computer Simulation Of Cit
    0.04812039   2485 Nolan        #  Managing the Computer Reso
    0.04775190   2826 Burtnyk      #  Interactive Skeleton Techn
    0.04529328   3086 Fredman      #  On the Complexity of Compu
    0.04512575   2834 Bitner       #  Efficient Generation of th
    0.04485830   3040 Freuder      #  Synthesizing Constraint Ex
    0.04446212   2909 Wirth        #  What Can We Do about the U

Show the terms that overlap between the query and retrieved docs (y/n): y
============================================================
Vector Overlap          6       2828  Docfreq
============================================================
geometric              11         41  9
motion                 23          5  9
============================================================
Vector Overlap          6       2753  Docfreq
============================================================
planning               10         67  18
============================================================
Vector Overlap          6       3035  Docfreq
============================================================
planning               10         41  18
============================================================
Vector Overlap          6       2087  Docfreq
============================================================
combinatorial          10         20  20
============================================================
```

```
Vector Overlap          6       2389  Docfreq
============================================================
planning               10         25  18

Continue (y/n)?
============================================================
Vector Overlap          6       1543  Docfreq
============================================================
motion                 23         11  9
============================================================
Vector Overlap          6       2721  Docfreq
============================================================
planning               10         25  18
dynamics               13         33  4
============================================================
Vector Overlap          6       2187  Docfreq
============================================================
planning               10         25  18
============================================================
Vector Overlap          6       2230  Docfreq
============================================================
geometric              11          5  9
planning               10         25  18
============================================================
Vector Overlap          6        695  Docfreq
============================================================
arm                    13          6  3
motion                 23          5  9

Continue (y/n)?
============================================================
Vector Overlap          6       2671  Docfreq
============================================================
combinatorial          10         20  20



    ************************************************************
  Documents Most Similar To Query number 9
    ************************************************************
    Similarity   Doc#  Author      Title
    ==========   ==== ========     ============================

    0.30567077   2949 Tajibnapis   #  A Correctness Proof of a T
    0.26576856   1685 Schurmann    #  GAN, a System for Generati
    0.25519867   2621 Purdy        #  A High Security Log-in Pro
    0.19683258   2849 Metcalfe     #  Ethernet: Distributed Pack
*   0.19441496   3068 Popek        #  A Model for Verification o
*   0.19313026   2372 Conway       #  On the Implementation of S
    0.18710868   2776 Chambers     #  Computer Networks in Highe
    0.17622703   2969 Morgan       #  Optimal Program and Data L
```

```
      0.17456445   3141 Chang        #  An Improved Algorithm for
      0.17272583   3082 Lamport      #  Time, Clocks, and the Orde
      0.17086823   2900 Grapa        #  Some Theorems to Aid in So
      0.16984913   1750 Fuchel       #  Considerations in the Desi
      0.16432632   3174 Morris       #  Password Security: A Case
      0.16416133   2317 Rosen        #  Programming Systems and La
      0.16348983   2197 Nielsen      #  The Merit of Regional Comp
      0.16169326   2311 Benjamin     #  A Generational Perspective
      0.16024165   2951 Mamrak       #  Dynamic Response Time Pred
      0.15518572   2614 Crandall     #  Arrow to Precedence Networ
      0.14827486   2948 Heckel       #  A Terminal-Oriented Commun
      0.14798472   2345 Ashenhurst   #  Curriculum Recommendations
      0.14763691   1695 Dill         #  PLEXUS-An On-Line System f

Show the terms that overlap between the query and retrieved docs (y/n): y
============================================================
Vector Overlap          9       2949  Docfreq
============================================================
network                 7         43  64
networks                8         34  44
operating               6         13  121
distributed             9         62  27
============================================================
Vector Overlap          9       1685  Docfreq
============================================================
network                 7        117  64
networks                8         21  44
============================================================
Vector Overlap          9       2621  Docfreq
============================================================
Security               12         18  7
systems                 8         20  344
operating               6         13  121
============================================================
Vector Overlap          9       2849  Docfreq
============================================================
networks                8         21  44
local                   9          4  22
systems                 8          2  344
operating               6          3  121
distributed             9         52  27
============================================================
Vector Overlap          9       3068  Docfreq
============================================================
Security               12         18  7
systems                 8         13  344
operating               6          9  121

Continue (y/n)?
============================================================
Vector Overlap          9       2372  Docfreq
============================================================
Security               12         18  7
systems                 8         22  344
```

```
operating               6         16  121
============================================================
Vector Overlap          9       2776  Docfreq
============================================================
network                 7         11  64
networks                8         42  44
operating               6          3  121
============================================================
Vector Overlap          9       2969  Docfreq
============================================================
network                 7          3  64
networks                8         21  44
distributed             9         19  27
============================================================
Vector Overlap          9       3141  Docfreq
============================================================
systems                 8         17  344
operating               6         13  121
distributed             9         19  27
============================================================
Vector Overlap          9       3082  Docfreq
============================================================
networks                8         17  44
systems                 8         17  344
distributed             9          9  27

Continue (y/n)?
============================================================
Vector Overlap          9       2900  Docfreq
============================================================
network                 7          3  64
networks                8         17  44
distributed             9         19  27
```

```
    ************************************************************
  Documents Most Similar To Query number 22
    ************************************************************
    Similarity   Doc#  Author      Title
    ==========   ==== ========     ==============================

*   0.31145323   2678 Wright       #  Visible Surface Plotting P
*   0.30636232   2751 Phong        #  Illumination for Computer
*   0.23869886   2473 Macleod      #  Hidden-Line Plotting Progr
*   0.23144756   2369 Matsushita   #  Hidden Lines Elimination f
    0.22973724   2004 Bouknight    #  A Procedure for Generation
*   0.22742001   2827 Levin        #  A Parametric Algorithm for
*   0.22286138   2384 Williamson   #  Hidden-Line Plotting Progr
*   0.21184921   2829 Blinn        #  Texture and Reflection in
    0.21082257   2913 Crow         #  The Aliasing Problem in Co
```

```
217   *   0.20077396   2828 Clark        #  Hierarchical Geometric Mod
218   *   0.16111961   2692 Sutherland   #  Reentrant Polygon Clipping
219       0.15615843   1915 Galimberti   #  An Algorithm for Hidden Li
220       0.13000577   1978 Smith        #  The Use of Interactive Gra
221       0.11224889   2925 Fuchs        #  Optimal Surface Reconstruc
222       0.09839900   2924 Wu
223       0.07500335   2687 Jordan       #  A Cell Organized Raster Di
224       0.07334894   2809 Bauer        #  Positivity and Norms
225       0.07265295   2674 Barrett      #  Scan Conversion Algorithms
226       0.06571736   2152 Newman       #  Display Procedures
227       0.05924804   2987 Burton       #  Representation of Many-Sid
228       0.05868690   2003 Bracchi      #  An Interactive Software Sy
229
230
231   Show the terms that overlap between the query and retrieved docs (y/n): y
232   ============================================================
233   Vector Overlap         22       2678  Docfreq
234   ============================================================
235   surface                10         20  21
236   graphics                8         16  50
237   computer                3          7  519
238   hidden                 22         22  13
239   line                    7         14  81
240   ============================================================
241   Vector Overlap         22       2751  Docfreq
242   ============================================================
243   surface                10         35  21
244   algorithms              5          2  189
245   graphics                8         16  50
246   computer                3          9  519
247   hidden                 22         38  13
248   ============================================================
249   Vector Overlap         22       2473  Docfreq
250   ============================================================
251   surface                10         20  21
252   I                       5          8  176
253   hidden                 22         22  13
254   line                    7         14  81
255   ============================================================
256   Vector Overlap         22       2369  Docfreq
257   ============================================================
258   This                    3          1  470
259   graphics                8         33  50
260   computer                3         14  519
261   hidden                 22         27  13
262   line                    7         18  81
263   ============================================================
264   Vector Overlap         22       2004  Docfreq
265   ============================================================
266   surface                10         45  21
267   This                    3          1  470
268   graphics                8         16  50
269   computer                3         12  519
270   hidden                 22         22  13
```

```
line                    7         14  81

Continue (y/n)?
============================================================
Vector Overlap         22       2827  Docfreq
============================================================
surface                10         55  21
This                    3          1  470
graphics                8         16  50
computer                3          7  519
hidden                 22         27  13
============================================================
Vector Overlap         22       2384  Docfreq
============================================================
surface                10         20  21
hidden                 22         22  13
line                    7         14  81
============================================================
Vector Overlap         22       2829  Docfreq
============================================================
surface                10         35  21
This                    3          1  470
graphics                8         16  50
computer                3          7  519
hidden                 22         22  13
============================================================
Vector Overlap         22       2913  Docfreq
============================================================
surface                10         25  21
algorithms              5          2  189
This                    3          1  470
graphics                8         16  50
computer                3          7  519
hidden                 22         27  13
============================================================
Vector Overlap         22       2828  Docfreq
============================================================
surface                10         50  21
algorithms              5         22  189
computer                3          1  519
hidden                 22         22  13

Continue (y/n)?
============================================================
Vector Overlap         22       2692  Docfreq
============================================================
surface                10         25  21
algorithms              5         14  189
This                    3          1  470
graphics                8         16  50
I                       5          8  176
computer                3          7  519
hidden                 22         27  13
```

3.

```
  ************************************************************
  Documents Most Similar To Document number 239
  ************************************************************
  Similarity   Doc#  Author      Title
  ==========   ====  ========    ==============================

  1.00000000   239   Verhoeff    #  Inefficiency of the Use of
  0.47631382   1032  Belzer      #  Theoretical Considerations
  0.20583021   651   Grems       #  A Survey of Languages and
  0.17848173   652   Sable       #  Use of Semantic Structure
  0.16159998   634   Salton      #  Manipulation of Trees in I
  0.15582079   1329  Mano        #  Simulation of Boolean Func
  0.15499013   1207  Dodd        #  Remarks on Simulation of B
  0.14173158   275   Sams        #  Dynamic Storage Allocation
  0.12039201   3012  Lucas       #  The Use of an Interactive
  0.11928562   635   Baker       #  A Note on Multiplying Bool
  0.11922194   2070  Hsiao       #  A Formal System for Inform
  0.11650461   2965  Hanani      #  An Optimal Evaluation of B
  0.11069690   2160  Wong        #  Canonical Structure in Att
  0.11035596   891   Whitley     #  Everyman's Information Ret
  0.10731072   1233  --          #  Conventions for the Use of
  0.10485182   292   Kehl        #  An Information Retrieval L
  0.10431438   1457  Salton      #  Data Manipulation and Prog
  0.10133893   3168  Laird       #  Comment on "An Optimal Eva
  0.10114213   2824  Duong       #  An Improvement to Martin's
  0.09787172   948   Healy       #  Note on the Use of Procedu
  0.09722451   2340  Martin      #  A Boolean Matrix Method fo


  ************************************************************
  Documents Most Similar To Document number 1236
  ************************************************************
  Similarity   Doc#  Author      Title
  ==========   ====  ========    ==============================

  1.00000000   1236  Salton      #  The SMART Automatic Docume
  0.26313981   1457  Salton      #  Data Manipulation and Prog
  0.25511322   634   Salton      #  Manipulation of Trees in I
  0.21953196   2307  Salton      #  Dynamic Document Processin
  0.18317411   2711  Salton      #  A Vector Space Model for A
  0.15439938   2575  Van         #  The Best-Match Problem in
  0.12829209   1536  Lesk        #  Dynamic Computation of Der
  0.12641095   2990  Yu
  0.12399758   3012  Lucas       #  The Use of an Interactive
  0.11582457   1699  Rubinoff    #  Experimental Evaluation of
  0.11150162   3135  Lesk        #  Detection of Three-Dimensi
  0.10703998   891   Whitley     #  Everyman's Information Ret
  0.10702277   1271  Davis       #  Secondary Key Retrieval Us
  0.10276252   2278  Tan         #  On Foster's Information St
```

```
   0.10058845   1927 Salton       #  Information Science in a P
   0.09803172   275  Sams         #  Dynamic Storage Allocation
   0.09451158   1515 Levien       #  A Computer System for Infe
   0.09192346   651  Grems        #  A Survey of Languages and
   0.09096082   1937 Day          #  CODAS: A Data Display Syst
   0.08962309   798  Scheff       #  A Catalogue Entry Retrieva
   0.08934509   2947 Schneider    #  SITAR: An Interactive Text



   ************************************************************
 Documents Most Similar To Document number 2740
   ************************************************************
   Similarity   Doc#  Author      Title
   ==========   ==== ========     ==============================

   1.00000000   2740 Lauesen      #  A Large Semaphore Based Op
   0.33996310   1749 Dijkstra     #  The Structure of the "THE"
   0.25771543   2379 Liskov       #  The Design of the Venus Op
   0.23048474   2920 Devillers    #  Game Interpretation of the
   0.22896835   2378 Gaines       #  An Operating System Based
   0.20851351   2228 Holt         #  Comments on Prevention of
   0.20388055   2500 Frailey      #  A Practical Approach to Ma
   0.19428052   2342 Gilbert      #  Interference Between Commu
   0.19191154   3043 Hansen       #  Distributed Processes: A C
   0.17824980   2280 Parnas       #  Comment on Deadlock Preven
   0.16791430   2080 Hansen       #  The Nucleus of a Multiprog
   0.15520010   2597 Hoare        #  Monitors: An Operating Sys
   0.15350404   2865 Owicki       #  Verifying Properties of Pa
   0.15328171   2320 Hansen       #  Structured Multiprogrammin
   0.15266582   2777 Parnas       #  On a Solution to the Cigar
   0.14506061   2618 Lamport      #  A New Solution of Dijkstra
   0.14427527   2376 Habermann    #  Synchronization of Communi
   0.14356215   2542 Graham       #  A Software Design and Eval
   0.14127104   2541 Balzer       #  An Overview of the ISPL Co
   0.14086411   2700 Lipton       #  Reduction: A Method of Pro
   0.13642901   1752 Oppenheimer  #  Resource Management for a
```

```
With permutation 3b):

1.

    ************************************************************
  Documents Most Similar To Query number 6
    ************************************************************
    Similarity   Doc#  Author      Title
    ==========   ====  ========    ============================

*   0.21002736   1543 Howard       #  Computer Formulation of th
    0.20351658   356  Ingerman     #  INTEREST (Algorithm 45)
    0.20038027   740  Wright       #  INTEREST (Algorithm 45)
*   0.18475864   2078 Eastman      #  Representations for Space
    0.18256636   438  Gorn         #  Mechanical Pragmatics: A T
    0.16838377   136  Ingerman     #  A Note on the Calculation
*   0.14208068   2828 Clark        #  Hierarchical Geometric Mod
    0.12849758   242  Wilson       #  Notes on Geometric Weighte
    0.12358801   1009 Weinberg     #  Solution of Combinatorial
    0.11656377   2389 Eastman      #  Preliminary Report on a Sy
    0.11393833   1186 Lynch        #  Recursive Solution of a Cl
    0.11278336   3035 Wetherbe     #  A Strategic Planning Metho
    0.10671978   2671 Stone        #  A Note on a Combinatorial
    0.10516681   2230 Bracchi      #  A Language for Treating Ge
    0.10283454   2417 Ehrlich      #  Four Combinatorial Algorit
    0.10040089   705  Blakely      #  Combinatorial Of M Things
    0.10011241   1398 Sterling     #  Robot Data Screening: A So
    0.10002990   704  Collins      #  Combinatorial of M Things
    0.09771736   2187 Amarel       #  Computer Science: A Concep
    0.09632651   2826 Burtnyk      #  Interactive Skeleton Techn
    0.09538226   2753 Pfefferkorn  #  A Heuristic Problem Solvin

Show the terms that overlap between the query and retrieved docs (y/n): y
============================================================
Vector Overlap          6       1543  Docfreq
============================================================
motion                 22         27  13
============================================================
Vector Overlap          6        356  Docfreq
============================================================
interest               15         11  70
============================================================
Vector Overlap          6        740  Docfreq
============================================================
interest               15         11  70
============================================================
Vector Overlap          6       2078  Docfreq
============================================================
plan                    9         22  35
robot                  13         27  3
============================================================
Vector Overlap          6        438  Docfreq
============================================================
motion                 22         16  13
```

```
 55
 56  Continue (y/n)?
 57  ============================================================
 58  Vector Overlap          6        136  Docfreq
 59  ============================================================
 60  interest               15         11  70
 61  ============================================================
 62  Vector Overlap          6       2828  Docfreq
 63  ============================================================
 64  geometr                11         55  13
 65  motion                 22          5  13
 66  ============================================================
 67  Vector Overlap          6        242  Docfreq
 68  ============================================================
 69  geometr                11         22  13
 70  ============================================================
 71  Vector Overlap          6       1009  Docfreq
 72  ============================================================
 73  combinatori             9         18  29
 74  ============================================================
 75  Vector Overlap          6       2389  Docfreq
 76  ============================================================
 77  plan                    9         36  35
 78
 79  Continue (y/n)?
 80  ============================================================
 81  Vector Overlap          6       1186  Docfreq
 82  ============================================================
 83  combinatori             9         18  29
 84
 85
 86
 87
 88
 89
 90      ************************************************************
 91    Documents Most Similar To Query number 9
 92      ************************************************************
 93      Similarity   Doc#  Author      Title
 94      ==========   ==== ========     ==============================
 95
 96      0.52918307   1685 Schurmann    #  GAN, a System for Generati
 97      0.41612638   2949 Tajibnapis   #  A Correctness Proof of a T
 98  *   0.39925599   3158 Denning      #  Secure Personal Computing
 99      0.38787696   2197 Nielsen      #  The Merit of Regional Comp
100      0.36611288   2614 Crandall     #  Arrow to Precedence Networ
101      0.35131952   2776 Chambers     #  Computer Networks in Highe
102  *   0.34660689   3068 Popek        #  A Model for Verification o
103  *   0.33798295   3111 Merkle       #  Secure Communications Over
104      0.33004837   2951 Mamrak       #  Dynamic Response Time Pred
105      0.31717058   2969 Morgan       #  Optimal Program and Data L
106      0.30267472   1261 Larsen       #  Modeling and Simulation of
107      0.30209452   2515 Corneil      #  Minimal Event-Node Network
108      0.30187313   2864 Madison      #  Characteristics of Program
```

```
109       0.29458811   1695 Dill          #  PLEXUS-An On-Line System f
110   *   0.29306130   2372 Conway        #  On the Implementation of S
111       0.29139158   1723 Fisher        #  Computer Construction of P
112       0.28146492   2454 Buzen         #  Computational Algorithms f
113   *   0.27973219   2870 Denning       #  A Lattice Model of Secure
114       0.27316692   2621 Purdy         #  A High Security Log-in Pro
115       0.27262397   2371 Walden        #  A System for Interprocess
116       0.26819417   2581 Miller        #  A Locally-Organized Parser
117
118
119   Show the terms that overlap between the query and retrieved docs (y/n): y
120   ============================================================
121   Vector Overlap          9       1685  Docfreq
122   ============================================================
123   network                14        135  90
124   system                  5          5  728
125   ============================================================
126   Vector Overlap          9       2949  Docfreq
127   ============================================================
128   distribut               6         52  123
129   network                14         82  90
130   system                  5          5  728
131   oper                    4          8  381
132   ============================================================
133   Vector Overlap          9       3158  Docfreq
134   ============================================================
135   secur                   9         46  31
136   network                14         32  90
137   ============================================================
138   Vector Overlap          9       2197  Docfreq
139   ============================================================
140   network                14         75  90
141   oper                    4          2  381
142   ============================================================
143   Vector Overlap          9       2614  Docfreq
144   ============================================================
145   network                14         39  90
146
147   Continue (y/n)?
148   ============================================================
149   Vector Overlap          9       2776  Docfreq
150   ============================================================
151   consider                7          3  66
152   network                14         53  90
153   oper                    4          2  381
154   ============================================================
155   Vector Overlap          9       3068  Docfreq
156   ============================================================
157   secur                   9         51  31
158   system                  5         16  728
159   oper                    4         21  381
160   ============================================================
161   Vector Overlap          9       3111  Docfreq
162   ============================================================
```

```
distribut               6         13  123
secur                   9         83  31
network                14         14  90
============================================================
Vector Overlap          9       2951  Docfreq
============================================================
network                14         60  90
system                  5         17  728
============================================================
Vector Overlap          9       2969  Docfreq
============================================================
distribut               6         13  123
network                14         32  90

Continue (y/n)?
============================================================
Vector Overlap          9       1261  Docfreq
============================================================
network                14         28  90
system                  5          1  728



    ************************************************************
  Documents Most Similar To Query number 22
    ************************************************************
    Similarity   Doc#  Author      Title
    ==========   ==== ========     ==============================

*   0.41477337   2678 Wright       #  Visible Surface Plotting P
*   0.37730262   2751 Phong        #  Illumination for Computer
*   0.37114874   2384 Williamson   #  Hidden-Line Plotting Progr
*   0.36280972   2473 Macleod      #  Hidden-Line Plotting Progr
*   0.32763273   2369 Matsushita   #  Hidden Lines Elimination f
*   0.32255454   2564 Ellis        #  Hidden-Line Plotting Progr
*   0.32255454   2637 Ellis        #  Hidden-Line Plotting Progr
*   0.30906059   2441 Williamson   #  Hidden-Line Plotting Progr
*   0.29602729   2638 Gaither      #  Hidden-Line Plotting Progr
    0.26945738   2004 Bouknight    #  A Procedure for Generation
*   0.25912337   2827 Levin        #  A Parametric Algorithm for
    0.24638178   1915 Galimberti   #  An Algorithm for Hidden Li
*   0.24533004   2829 Blinn        #  Texture and Reflection in
    0.20927606   2913 Crow         #  The Aliasing Problem in Co
*   0.20421055   2828 Clark        #  Hierarchical Geometric Mod
    0.19612527   52   Cook         #  An Efficient Method for Ge
    0.18365510   88   Hicks        #  An Efficient Method for Ge
    0.18066250   266  Robinson     #  Fitting Spheres by the Met
    0.16421981   1978 Smith        #  The Use of Interactive Gra
    0.16218509   2841 Clark        #  Designing Surfaces in 3-D

Show the terms that overlap between the query and retrieved docs (y/n): y
============================================================
Vector Overlap         22       2678  Docfreq
```

```
============================================================
algorithm               1          2  1342
surfac                  9         32  33
graphic                 7         14  80
comput                  2          4  926
hidden                 20         20  17
line                    6         13  110
============================================================
Vector Overlap         22       2751  Docfreq
============================================================
algorithm               1          2  1342
surfac                  9         32  33
graphic                 7         29  80
comput                  2          9  926
hidden                 20         36  17
============================================================
Vector Overlap         22       2384  Docfreq
============================================================
algorithm               1          2  1342
surfac                  9         18  33
hidden                 20         36  17
line                    6         23  110
============================================================
Vector Overlap         22       2473  Docfreq
============================================================
algorithm               1          2  1342
surfac                  9         18  33
hidden                 20         36  17
line                    6         23  110
============================================================
Vector Overlap         22       2369  Docfreq
============================================================
graphic                 7         29  80
comput                  2          9  926
hidden                 20         41  17
line                    6         26  110

Continue (y/n)?
============================================================
Vector Overlap         22       2564  Docfreq
============================================================
algorithm               1          2  1342
hidden                 20         15  17
line                    6         10  110
============================================================
Vector Overlap         22       2637  Docfreq
============================================================
algorithm               1          2  1342
hidden                 20         15  17
line                    6         10  110
============================================================
Vector Overlap         22       2441  Docfreq
============================================================
algorithm               1          2  1342
```

```
271   hidden                 20         15  17
272   line                    6         10  110
273   ============================================================
274   Vector Overlap         22       2638  Docfreq
275   ============================================================
276   algorithm               1          2  1342
277   hidden                 20         15  17
278   line                    6         10  110
279   ============================================================
280   Vector Overlap         22       2004  Docfreq
281   ============================================================
282   algorithm               1          4  1342
283   surfac                  9         41  33
284   graphic                 7         25  80
285   comput                  2         13  926
286   hidden                 20         20  17
287   line                    6         13  110
288
289   Continue (y/n)?
290   ============================================================
291   Vector Overlap         22       2827  Docfreq
292   ============================================================
293   algorithm               1          4  1342
294   surfac                  9         77  33
295   graphic                 7         14  80
296   comput                  2          4  926
297   hidden                 20         26  17
298
299
300
301
302
303
304   3.
305
306      ************************************************************
307     Documents Most Similar To Document number 239
308      ************************************************************
309      Similarity   Doc#  Author      Title
310      ==========   ==== ========     ==============================
311
312      1.00000000   239  Verhoeff     #  Inefficiency of the Use of
313      0.56849762   1032 Belzer       #  Theoretical Considerations
314      0.21175483   2965 Hanani       #  An Optimal Evaluation of B
315      0.20784559   3168 Laird        #  Comment on "An Optimal Eva
316      0.19699697   2160 Wong         #  Canonical Structure in Att
317      0.18748378   3169 Gudes        #  Note on "An Optimal Evalua
318      0.15633700   3012 Lucas        #  The Use of an Interactive
319      0.15361459   3134 Motzkin      #  The Use of Normal Multipli
320      0.14880001   1207 Dodd         #  Remarks on Simulation of B
321      0.14875752   1329 Mano         #  Simulation of Boolean Func
322      0.14387306   1457 Salton       #  Data Manipulation and Prog
323      0.14022802   891  Whitley      #  Everyman's Information Ret
324      0.12823256   651  Grems        #  A Survey of Languages and
```

```
  0.12480715   2278 Tan          #  On Foster's Information St
  0.12107457   1699 Rubinoff     #  Experimental Evaluation of
  0.11682826   635  Baker        #  A Note on Multiplying Bool
  0.11579026   275  Sams         #  Dynamic Storage Allocation
  0.11437755   634  Salton       #  Manipulation of Trees in I
  0.11216122   2345 Ashenhurst   #  Curriculum Recommendations
  0.11032076   2516 Salasin      #  Hierarchical Storage in In
  0.10784234   2140 Mullin       #  Retrieval-Update Speed Tra


  ************************************************************
Documents Most Similar To Document number 1236
  ************************************************************
  Similarity   Doc#  Author      Title
  ==========   ====  ========    ==============================

  1.00000000   1236 Salton       #  The SMART Automatic Docume
  0.34964558   634  Salton       #  Manipulation of Trees in I
  0.34598550   1699 Rubinoff     #  Experimental Evaluation of
  0.33443320   2711 Salton       #  A Vector Space Model for A
  0.30467298   2307 Salton       #  Dynamic Document Processin
  0.29321057   1457 Salton       #  Data Manipulation and Prog
  0.27846861   2575 Van          #  The Best-Match Problem in
  0.23597347   1681 Rubinoff     #  Easy English,a Language fo
  0.23453393   2990 Yu
  0.22245364   1032 Belzer       #  Theoretical Considerations
  0.21166700   3012 Lucas        #  The Use of an Interactive
  0.20299051   3134 Motzkin      #  The Use of Normal Multipli
  0.17817917   2293 Jones        #  Comment on Average Binary
  0.17475402   891  Whitley      #  Everyman's Information Ret
  0.17474206   2140 Mullin       #  Retrieval-Update Speed Tra
  0.16965934   1536 Lesk         #  Dynamic Computation of Der
  0.16182354   1935 Arora        #  Randomized Binary Search T
  0.16070823   1271 Davis        #  Secondary Key Retrieval Us
  0.15524160   651  Grems        #  A Survey of Languages and
  0.15418592   1680 Engvold      #  A General-Purpose Display
  0.15371785   2157 Flores       #  Average Binary Search Leng


  ************************************************************
Documents Most Similar To Document number 2740
  ************************************************************
  Similarity   Doc#  Author      Title
  ==========   ====  ========    ==============================

  1.00000000   2740 Lauesen      #  A Large Semaphore Based Op
  0.31222130   1749 Dijkstra     #  The Structure of the "THE"
  0.28679769   2379 Liskov       #  The Design of the Venus Op
  0.27574815   2597 Hoare        #  Monitors: An Operating Sys
  0.27110798   3043 Hansen       #  Distributed Processes: A C
  0.26309867   2700 Lipton       #  Reduction: A Method of Pro
  0.25401819   2618 Lamport      #  A New Solution of Dijkstra
  0.24792167   2376 Habermann    #  Synchronization of Communi
  0.24362829   2866 Howard       #  Proving Monitors
```

```
379       0.23251429   2500 Frailey      #  A Practical Approach to Ma
380       0.22777362   2228 Holt         #  Comments on Prevention of
381       0.22438410   2920 Devillers    #  Game Interpretation of the
382       0.22230500   2280 Parnas       #  Comment on Deadlock Preven
383       0.22161433   3128 Reed         #  Synchronization with Event
384       0.21423306   1611 Klein        #  Scheduling Project Network
385       0.21074517   2482 Howard       #  Mixed Solutions for the De
386       0.19534673   2080 Hansen       #  The Nucleus of a Multiprog
387       0.19258408   2320 Hansen       #  Structured Multiprogrammin
388       0.18766565   2777 Parnas       #  On a Solution to the Cigar
389       0.18088493   2738 Parnas       #  Use of the Concept of Tran
390       0.17666555   2378 Gaines       #  An Operating System Based
391
392
393
394
```

Precision / Recall:

****
** Precision / Recall averaged over 33 queries **
****

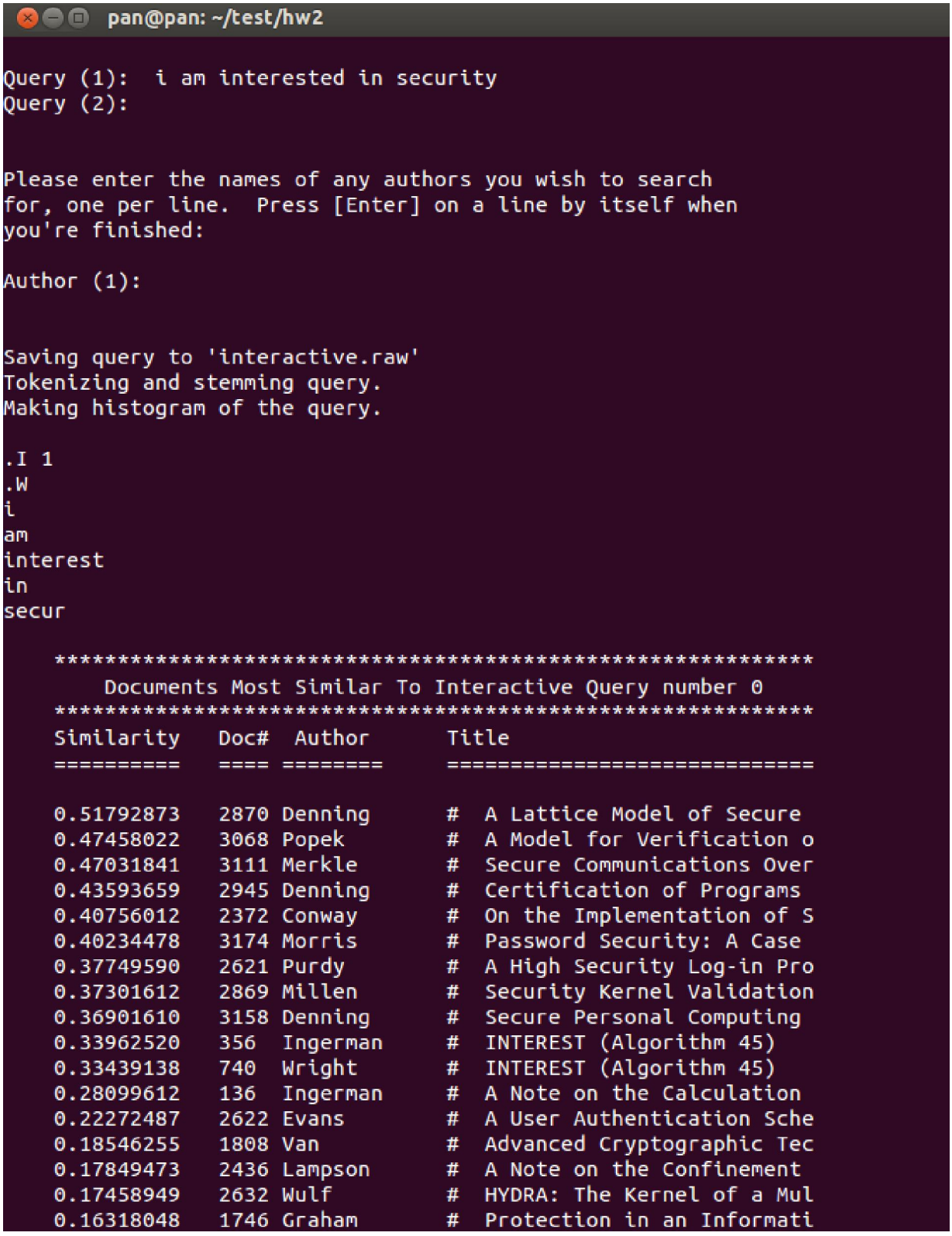
Permutation Name	P 0.25	P 0.50	P 0.75	P 1.00	P mean1	P mean2	P norm	R norm
RAW TF	0.367	0.241	0.118	0.047	0.242	0.297	0.630	0.904
Boolean	0.179	0.114	0.061	0.016	0.118	0.131	0.532	0.883
Dice	0.481	0.346	0.222	0.099	0.350	0.390	0.713	0.931
Jaccard	0.262	0.137	0.032	0.011	0.144	0.185	0.448	0.616
Overlap	0.540	0.347	0.213	0.083	0.367	0.427	0.716	0.933
Unstem	0.514	0.270	0.114	0.053	0.300	0.370	0.614	0.827
No stwd	0.541	0.385	0.247	0.081	0.391	0.429	0.730	0.937
Region 1111	0.546	0.362	0.242	0.086	0.383	0.439	0.733	0.933
Region 1114	0.456	0.297	0.178	0.061	0.310	0.353	0.700	0.929
Default	0.549	0.385	0.247	0.098	0.394	0.436	0.732	0.932

Precision / Recall with bigrams

****
** Precision / Recall averaged over 33 queries **
****

Permutation Name	P 0.25	P 0.50	P 0.75	P 1.00	P mean1	P mean2	P norm	R norm
RAW TF	0.429	0.261	0.127	0.052	0.273	0.340	0.642	0.907
Boolean	0.220	0.146	0.079	0.018	0.148	0.183	0.554	0.888
Dice	0.525	0.351	0.242	0.085	0.372	0.431	0.725	0.932
Jaccard	0.343	0.168	0.107	0.041	0.206	0.288	0.553	0.733
Overlap	0.555	0.361	0.223	0.090	0.380	0.429	0.728	0.934
Unstem	0.470	0.284	0.114	0.055	0.290	0.360	0.616	0.828
No stwd	0.588	0.405	0.257	0.085	0.417	0.476	0.741	0.937
Region 1111	0.583	0.410	0.250	0.091	0.414	0.467	0.747	0.934
Region 1114	0.539	0.347	0.202	0.060	0.363	0.416	0.718	0.931
Default	0.565	0.385	0.255	0.095	0.402	0.461	0.743	0.933

Interactive Query:

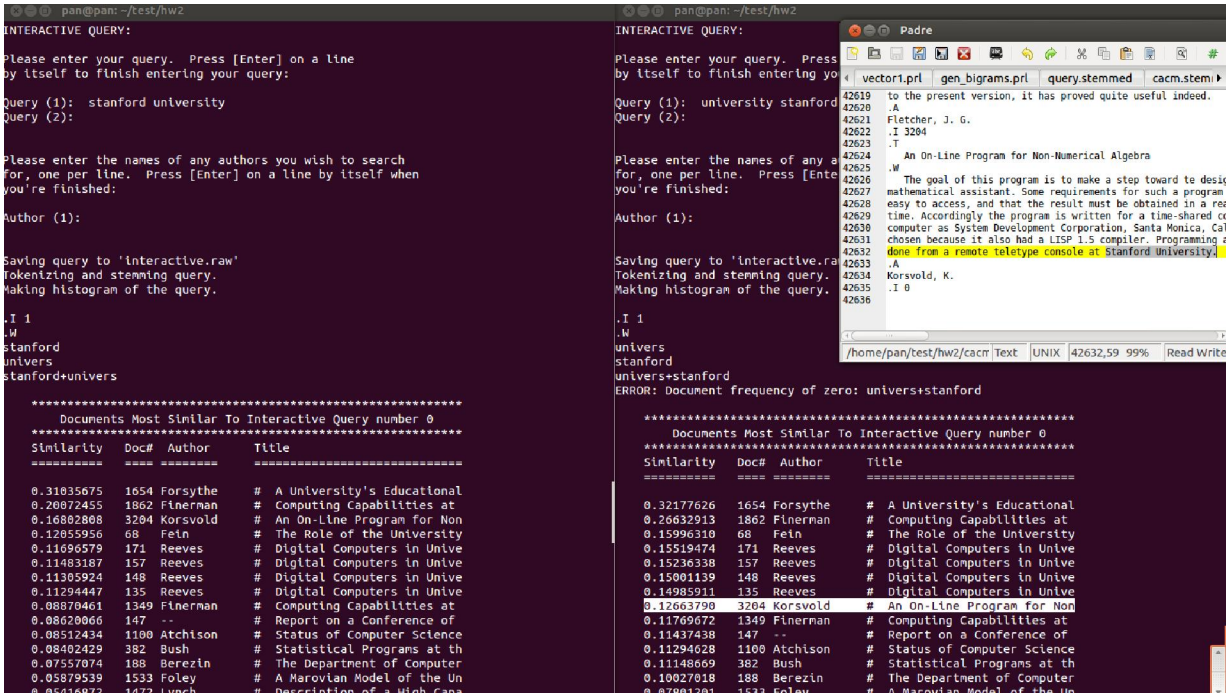


```
pan@pan: ~/test/hw2

Query (1):  i am interested in security
Query (2):


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

Author (1):


Saving query to 'interactive.raw'
Tokenizing and stemming query.
Making histogram of the query.

.I 1
.W
i
am
interest
in
secur

    ************************************************************
        Documents Most Similar To Interactive Query number 0
    ************************************************************
    Similarity   Doc#  Author      Title
    ==========   ====  ========    ============================

    0.51792873   2870 Denning      #  A Lattice Model of Secure
    0.47458022   3068 Popek        #  A Model for Verification o
    0.47031841   3111 Merkle       #  Secure Communications Over
    0.43593659   2945 Denning      #  Certification of Programs
    0.40756012   2372 Conway       #  On the Implementation of S
    0.40234478   3174 Morris       #  Password Security: A Case
    0.37749590   2621 Purdy        #  A High Security Log-in Pro
    0.37301612   2869 Millen       #  Security Kernel Validation
    0.36901610   3158 Denning      #  Secure Personal Computing
    0.33962520   356  Ingerman     #  INTEREST (Algorithm 45)
    0.33439138   740  Wright       #  INTEREST (Algorithm 45)
    0.28099612   136  Ingerman     #  A Note on the Calculation
    0.22272487   2622 Evans        #  A User Authentication Sche
    0.18546255   1808 Van          #  Advanced Cryptographic Tec
    0.17849473   2436 Lampson      #  A Note on the Confinement
    0.17458949   2632 Wulf         #  HYDRA: The Kernel of a Mul
    0.16318048   1746 Graham       #  Protection in an Informati
```

Bigram Improvement:

As you can see in the screen shot, we used bigram sets in both queries:
In the left part we used the ***stanford university*** through the interactive query.
In the right part we used the ***university stanford*** through the interactive query.

The document #3204 has the bigrams ***stanford university***, thus it ranks highly in the left part, whereas in the right part it ranks lower because we don't have the bigram ***university stanford*** in our document.



gen_bigrams.prl

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

#########################################################
## This perl script is used to generate all the bigrams
## within the .stemmed files and .tokenized files (except
## the files for interactive queries which will be generated
## dynamically each time a use make inputs.
##
## @INPUT                       @OUTPUT
## cacm.stemmed         >       cacm.stemmed.bigrams
## cacm.stemmed.hist    >       cacm.stemmed.hist.bigrams
## cacm.tokenized       >       cacm.tokenized.bigrams
## cacm.tokenized.hist  >       cacm.tokenized.hist.bigrams
## query.stemmed        >       query.stemmed.bigrams
## query.stemmed.hist   >       query.stemmed.hist.bigrams
## query.tokenized      >       query.tokenized.bigrams
## query.tokenized.hist >       query.tokenized.hist.bigrams
##
#########################################################

use Carp;
use FileHandle;

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

# 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("cacm");
&gen_bigrams("query");

exit(0);


#########################################################
## 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");
```

```
90
91          while (defined( $word = <$stemmed_file_fh> )) {
92                  chomp $word;
93
94                  # if we encounter a normal word, we check if it eligible for bigram
95                  if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {   # start of query tokens
96                    # if both words are not in the stoplist, we added to the bigrams array
97                    if (! exists $stoplist_stemmed_hash{ $last_word } and ! exists $stoplist_stemmed_hash{ $word }) {
98                          push (@bigrams, "$last_word+$word");
99                    }
100                 }
101
102                 # if next doc/query, pop all the elements in the array into file
103                 if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
104                         # after we scanned each doc/qry, all the bigrams are in the array now
105                         # we append all the bigrams to the end of doc/qry
106                         foreach $bigram (@bigrams) {
107                                 print RAWFILE "$bigram\n";
108                         }
109                         # clean up the array
110                         @bigrams = ();
111                 }
112
113                 # we shift last_word by 1 and write the current word into the output file
114                 $last_word = $word;
115                 print RAWFILE "$word\n";
116         }
117
118         # do it once again to make sure we have everything write into the file
119         if ($#bigrams != -1) {
120                 foreach $bigram (@bigrams) {
121                         print RAWFILE "$bigram\n";
122                 }
123         }
124
125         # close the bigram file after we created it
126         close(RAWFILE);
127
128         # generate hist file
129         system ("$stem_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
130
131         # clean up the array
132         @bigrams = ();
133
134         #########
135         # make the tokenlized bigrams
136         #########
137         open(RAWFILE, ">$tokened_file\.bigrams");
138
139         while (defined( $word = <$tokened_file_fh> )) {
140                 chomp $word;
141
142                 # if we encounter a normal word, we check if it eligible for bigram
143                 if ($last_word =~ /^[a-zA-Z]/ and $word =~ /^[a-zA-Z]/) {   # start of query tokens
144                   # if both words are not in the stoplist, we added to the bigrams array
145                   if (! exists $stoplist_hash{ $last_word } and ! exists $stoplist_hash{ $word }) {
146                         push (@bigrams, "$last_word+$word");
147                   }
148                 }
149
150                 # if next doc/query, pop all the elements in the array into file
151                 if ($word =~ /^\.[A-Z]/ and $#bigrams != -1) {
152                         # after we scanned each doc/qry, all the bigrams are in the array now
153                         # we append all the bigrams to the end of doc/qry
154                         foreach $bigram (@bigrams) {
155                                 print RAWFILE "$bigram\n";
156                         }
157                         # clean up the array
158                         @bigrams = ();
159                 }
160
161                 # we shift last_word by 1 and write the current word into the output file
162                 $last_word = $word;
163                 print RAWFILE "$word\n";
164         }
165
166         # do it once again to make sure we have everything write into the file
167         if ($#bigrams != -1) {
168                 foreach $bigram (@bigrams) {
169                         print RAWFILE "$bigram\n";
170                 }
171         }
172
173         # close the bigram file after we created it
174         close(RAWFILE);
175
176         # generate hist file
177         system ("$tokn_hist_com") and die "Failed $DIR/make_hist.prl: $!\n";
178
179         # clean up the array
180         @bigrams = ();
```

```
181     }
```

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     }
```

vector1.prl

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

use strict;

use Carp;
use FileHandle;
use List::MoreUtils qw(firstidx);

#########################################################
##  VECTOR1
##
##  Usage:   vector1     (no command line arguments)
##
##  The function &main_loop below gives the menu for the system.
##
##  This is an example program that shows how the core
##  of a vector-based IR engine may be implemented in Perl.
##
##  Some of the functions below are unimplemented, and some
##  are only partially implemented. Suggestions for additions
##  are given below and in the assignment handout.
##
##  You should feel free to modify this program directly,
##  and probably use this as a base for your implemented
##  extensions.  As with all assignments, the range of
##  possible enhancements is open ended and creativity
##  is strongly encouraged.
#########################################################


#########################################################
## Program Defaults and Global Variables
#########################################################

my $DIR  = "../hw2";
my $HOME = ".";

my $token_docs = "$DIR/cacm";           # tokenized cacm journals
my $corps_freq = "$DIR/cacm";           # frequency of each token in the journ.
my $stoplist   = "$DIR/common_words";   # common uninteresting words
my $titles     = "$DIR/titles.short";   # titles of each article in cacm
my $token_qrys = "$DIR/query";          # tokenized canned querys
my $query_freq = "$DIR/query";          # frequency of each token in the querys
my $query_relv = "$DIR/query\.rels";    # relevance of a journal entry to a
                                        #  given query

# these files are created in your $HOME directory

my $token_intr = "$HOME/interactive";    # file created for interactive queries
my $inter_freq = "$HOME/interactive";    # frequency of each token in above


# @doc_vector
#
#   An array of hashes, each array index indicating a particular document's
#   weight "vector".

my @doc_vector = ( );

# @qry_vector
#
#   An array of hashes, each array index indicating a particular query's
#   weight "vector".

my @qry_vector = ( );

# %docs_freq_hash
#
# associative array which holds <token, frequency> pairs where
#
#   token     = a particular word or tag found in the cacm corpus
#   frequency = the total number of times the token appears in
#               the corpus.

my %docs_freq_hash = ( );

# %corp_freq_hash
#
# associative array which holds <token, frequency> pairs where
#
#   token     = a particular word or tag found in the corpus
#   frequency = the total number of times the token appears per
#               document-- that is a token is counted only once
#               per document if it is present (even if it appears
#               several times within that document).

my %corp_freq_hash = ( );
```

```
88
89    # %stoplist_hash
90    #
91    # common list of uninteresting words which are likely irrelvant
92    # to any query.
93    #
94    #   Note: this is an associative array to provide fast lookups
95    #         of these boring words
96
97    my %stoplist_hash  = ( );
98
99    # @titles_vector
100   #
101   # vector of the cacm journal titles. Indexed in order of apperance
102   # within the corpus.
103
104   my @titles_vector  = ( );
105
106   # %relevance_hash
107   #
108   # a hash of hashes where each <key, value> pair consists of
109   #
110   #   key   = a query number
111   #   value = a hash consisting of document number keys with associated
112   #           numeric values indicating the degree of relevance the
113   #           document has to the particular query.
114
115   my %relevance_hash = ( );
116
117   # @doc_simula
118   #
119   # array used for storing query to document or document to document
120   # similarity calculations (determined by cosine_similarity, etc. )
121
122   my @doc_simula = ( );
123
124   # @res_vector
125   #
126   # array used for storing the document numbers of the most relevant
127   # documents in a query to document or document to document calculation.
128
129   my @res_vector = ( );
130
131   # $prec_mean1
132   #
133   # the variable to store the result of Prec_mean1
134
135   my $prec_mean1 = undef;
136
137   # $prec_mean2
138   #
139   # the variable to store the result of Prec_mean2
140
141   my $prec_mean2 = undef;
142
143   # $recall_norm
144   #
145   # the variable to store the result of Recall_norm
146
147   my $recall_norm = undef;
148
149   # $prec_norm
150   #
151   # the variable to store the result of Prec_norm
152
153   my $prec_norm = undef;
154
155   # make an hashtable to store the rank, docn pair so we can sort it
156   my %rank_docn = ();
157   # used to store the rec, prec table for each i in TOTAL_RELEVANT
158   my %rec_prec = ();
159   # the modified method parameters
160   my $method = undef;
161
162   # vars to store the total number of documents and quiries
163   my $total_docs = undef;
164   my $total_qrys = undef;
165
166   # global vars for the weight
167   my $TITLE_BASE_WEIGHT = 3;     # weight given a title token
168   my $KEYWD_BASE_WEIGHT = 4;     # weight given a key word token
169   my $AUTHR_BASE_WEIGHT = 3;     # weight given an an author token
170   my $ABSTR_BASE_WEIGHT = 1;     # weight given an abstract word token
171
172   # global variable for similarity, defalut is "cosine"
173   my $sim = "cosine";
174
175   # $bigrams = 1 if we are using bigrams in term set
176   # 0 means using default term set
```

```
177    my $bigrams = 0;
178
179    # start program
180
181    &main_loop;
182
183    ########################################################
184    ##  INIT_FILES
185    ##
186    ##  This function specifies the names and locations of
187    ##  input files used by the program.
188    ##
189    ##  Parameter:  $type   ("stemmed" or "unstemmed")
190    ##
191    ##  If $type == "stemmed", the filenames are initialized
192    ##  to the versions stemmed with the Porter stemmer, while
193    ##  in the default ("unstemmed") case initializes to files
194    ##  containing raw, unstemmed tokens.
195    ########################################################
196
197    sub init_files {
198
199        if ("stemmed" eq (shift || "")) {
200
201            if ($bigrams == 0) { # using the default term set
202                $token_docs .= "\.stemmed";
203                $corps_freq .= "\.stemmed\.hist";
204                $stoplist   .= "\.stemmed";
205                $token_qrys .= "\.stemmed";
206                $query_freq .= "\.stemmed\.hist";
207                $token_intr .= "\.stemmed";
208                $inter_freq .= "\.stemmed\.hist";
209            }
210            else {  # using the bigrams term set
211                $token_docs .= "\.stemmed\.bigrams";
212                $corps_freq .= "\.stemmed\.hist\.bigrams";
213                $stoplist   .= "\.stemmed";
214                $token_qrys .= "\.stemmed\.bigrams";
215                $query_freq .= "\.stemmed\.hist\.bigrams";
216                $token_intr .= "\.stemmed\.bigrams";
217                $inter_freq .= "\.stemmed\.hist\.bigrams";
218            }
219        }
220        else {
221            if ($bigrams == 0) { # using the default term set
222                $token_docs .= "\.tokenized";
223                $corps_freq .= "\.tokenized\.hist";
224                $token_qrys .= "\.tokenized";
225                $query_freq .= "\.tokenized\.hist";
226                $token_intr .= "\.tokenized";
227                $inter_freq .= "\.tokenized\.hist";
228            }
229            else { # using the bigrams term set
230                $token_docs .= "\.tokenized\.bigrams";
231                $corps_freq .= "\.tokenized\.hist\.bigrams";
232                $token_qrys .= "\.tokenized\.bigrams";
233                $query_freq .= "\.tokenized\.hist\.bigrams";
234                $token_intr .= "\.tokenized\.bigrams";
235                $inter_freq .= "\.tokenized\.hist\.bigrams";
236            }
237        }
238    }
239
240    ########################################################
241    ##  INIT_CORP_FREQ
242    ##
243    ##  This function reads in corpus and document frequencies from
244    ##  the provided histogram file for both the document set
245    ##  and the query set. This information will be used in
246    ##  term weighting.
247    ##
248    ##  It also initializes the arrays representing the stoplist,
249    ##  title list and relevance of document given query.
250    ########################################################
251
252    sub init_corp_freq {
253        my $corps_freq_fh = new FileHandle $corps_freq, "r"
254            or croak "Failed [corps_freq_fh] $corps_freq";
255
256        my $query_freq_fh = new FileHandle $query_freq, "r"
257            or croak "Failed [query_freq_fh] $query_freq";
258
259        my $stoplist_fh   = new FileHandle $stoplist  , "r"
260            or croak "Failed [stoplist_fh] $stoplist";
261
262        my $titles_fh     = new FileHandle $titles    , "r"
263            or croak "Failed [titles_fh] $titles";
264
265        my $query_relv_fh = new FileHandle $query_relv, "r"
```

```
266             or croak "Failed [query_relv_fh] $query_relv";
267
268         my $line = undef;
269
270         while (defined( $line = <$corps_freq_fh> )) {
271
272             # so on my computer split will return a first element of undef
273             # if the leading characters are white space, so I eat the white
274             # space to insure that the split works right.
275
276             my ($str) = ($line =~ /^\s*(\S.*)/);
277
278             my ($doc_freq,
279                 $cor_freq,
280                 $term    ) = split /\s+/, $str;
281             $docs_freq_hash{ $term } = $doc_freq;
282             $corp_freq_hash{ $term } = $cor_freq;
283         }
284
285
286         while (defined( $line = <$query_freq_fh> )) {
287
288             my ($str) = ($line =~ /^\s*(\S.*)/);
289
290             my ($doc_freq,
291                 $cor_freq,
292                 $term    ) = split /\s+/, $str;
293
294             $docs_freq_hash{ $term } += $doc_freq;
295             $corp_freq_hash{ $term } += $cor_freq;
296         }
297
298
299         while (defined( $line = <$stoplist_fh> )) {
300
301             chomp $line;
302             $stoplist_hash{ $line } = 1;
303         }
304
305
306         push @titles_vector, "";       # push one empty value onto @titles_vector
307                                        # so that indices correspond with title
308                                        # numbers.
309
310         while (defined( $line = <$titles_fh> )) {
311
312             chomp $line;
313             push @titles_vector, $line;
314         }
315
316
317         while (defined( $line = <$query_relv_fh> )) {
318
319             my ($str) = ($line =~ /^\s*(\S.*)/);
320
321             my ($qry_num,
322                 $rel_doc)  = split /\s+/, $str;
323
324             $relevance_hash{ "$qry_num" }{ "$rel_doc" } = 1;
325         }
326
327     }
328
329
330     ########################################################
331     ##  INIT_DOC_VECTORS
332     ##
333     ##  This function reads in tokens from the document file.
334     ##  When a .I token is encountered, indicating a document
335     ##  break, a new vector is begun. When individual terms
336     ##  are encountered, they are added to a running sum of
337     ##  term frequencies. To save time and space, it is possible
338     ##  to normalize these term frequencies by inverse document
339     ##  frequency (or whatever other weighting strategy is
340     ##  being used) while the terms are being summed or in
341     ##  a posthoc pass.  The 2D vector array
342     ##
343     ##    $doc_vector[ $doc_num ]{ $term }
344     ##
345     ##  stores these normalized term weights.
346     ##
347     ##  It is possible to weight different regions of the document
348     ##  differently depending on likely importance to the classification.
349     ##  The relative base weighting factors can be set when
350     ##  different segment boundaries are encountered.
351     ##
352     ##  This function is currently set up for simple TF weighting.
353     ########################################################
354
```

```
355    sub init_doc_vectors {
356
357        # my $TITLE_BASE_WEIGHT = 4;     # weight given a title token
358        # my $KEYWD_BASE_WEIGHT = 3;     # weight given a key word token
359        # my $ABSTR_BASE_WEIGHT = 1;     # weight given an abstract word token
360        # my $AUTHR_BASE_WEIGHT = 4;     # weight given an an author token
361
362        my $token_docs_fh = new FileHandle $token_docs, "r"
363            or croak "Failed $token_docs";
364
365        my $word    = undef;
366
367        my $doc_num =  0;    # current document number and total docs at end
368        my $tweight =  0;    # current weight assigned to document token
369
370        push @doc_vector, { };     # push one empty value onto @doc_vector so that
371                                   # indices correspond with document numbers
372
373        while (defined( $word = <$token_docs_fh> )) {
374
375            chomp $word;
376            # last if $word =~ /^\.I 0/; # indicates end of file so kick out
377
378            if ($word =~ /^\.I/) {     # indicates start of a new document
379
380                push @doc_vector, { };
381                $doc_num++;
382
383                next;
384            }
385
386            $tweight = $TITLE_BASE_WEIGHT and next if $word =~ /^\.T/;
387            $tweight = $KEYWD_BASE_WEIGHT and next if $word =~ /^\.K/;
388            $tweight = $ABSTR_BASE_WEIGHT and next if $word =~ /^\.W/;
389            $tweight = $AUTHR_BASE_WEIGHT and next if $word =~ /^\.A/;
390
391            if ($word =~ /[a-zA-Z]/ and ! exists $stoplist_hash{ $word }) {
392
393    #           print $word, "\n";
394    #           print $docs_freq_hash{ $word }, "\n";
395                if (defined( $docs_freq_hash{ $word } )) {
396
397    #               print $word, "\n";
398                    $doc_vector[$doc_num]{ $word } += $tweight;
399                }
400                else {
401                    print "ERROR: Document frequency of zero: ", $word, "\n";
402                }
403            }
404        }
405
406        # optionally normalize the raw term frequency
407        #
408        # foreach my $hash (@doc_vector) {
409        #     foreach my $key (keys %{ $hash }) {
410        #         $hash{ $key } = log( $doc_num / $docs_freq_hash{ $key });
411        #     }
412        # }
413
414
415        # calculate RAW TF on doc_vector
416        foreach my $hash (@doc_vector) {
417            foreach my $key (keys %{$hash}) {
418                $hash->{$key} = ($hash->{$key}* log($doc_num / $docs_freq_hash{ $key }));
419            }
420        }
421        return $doc_num;
422    }
423
424    ##########################################################
425    ##  INIT_QRY_VECTORS
426    ##
427    ##  This function should be nearly identical to the step
428    ##  for initializing document vectors.
429    ##
430    ##  This function is currently set up for simple TF weighting.
431    ##########################################################
432
433    sub init_qry_vectors {
434
435        my $QUERY_BASE_WEIGHT = 2;
436        my $QUERY_AUTH_WEIGHT = 2;
437
438        my $token_qrys_fh = new FileHandle $token_qrys, "r"
439            or croak "Failed $token_qrys";
440
441        my $word = undef;
442
443        my $tweight =  0;
```

```
444         my $qry_num =  0;
445 
446         push @qry_vector, { };    # push one empty value onto @qry_vectors so that
447                                   # indices correspond with query numbers
448 
449         while (defined( $word = <$token_qrys_fh> )) {
450 
451             chomp $word;
452 
453             if ($word =~ /^\.I/) {
454 
455                 push @qry_vector, { };
456                 $qry_num++;
457 
458                 next;
459             }
460 
461             $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
462             $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;
463 
464             if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {
465 
466                 if (! exists $docs_freq_hash{ $word }) {
467                     print "ERROR: Document frequency of zero: ", $word, "\n";
468                 }
469                 else {
470                     $qry_vector[$qry_num]{ $word } += $tweight;
471                 }
472             }
473         }
474 
475         # optionally normalize the raw term frequency
476         #
477         # foreach my $hash (@qry_vector) {
478         #     foreach my $key (keys %{ $hash }) {
479         #         $hash{ $key } = log( $qry_num / $docs_freq_hash{ $key });
480         #     }
481         # }
482 
483 
484         # calculate RAW TF on qry_vector
485         foreach my $hash (@qry_vector) {
486             foreach my $key (keys %{$hash}) {
487                 $hash->{$key} = ($hash->{$key} * log($total_docs/ $docs_freq_hash{ $key }));
488             }
489         }
490         return $qry_num;
491     }
492 
493 
494     ########################################################
495     ## INIT_METHOD
496     ##
497     ## Initialize the method parameters based on the user
498     ## inputs.
499     ##
500     ########################################################
501 
502     sub init_method {
503 
504             print <<"EndOfMenu";
505 
506             ============================================================
507             ==     Welcome to the 600.466 Vector-based IR Engine
508             ==
509             ==                 Choose Term Set
510             ============================================================
511 
512             Choose your term set ...
513 
514             OPTIONS:
515               1 = Use the default term set
516               2 = Augment the default term set with Bigrams
517 
518               0 = Quit
519 
520             ============================================================
521 
522     EndOfMenu
523         ;
524 
525         my $option = <STDIN>;
526         chomp $option;
527         if ($option !~ /[0-2]/) {
528             $option = 1;
529         }
530 
531         if($option == 0) {
532             exit 0;
```

```
533         }
534         elsif($option == 2) {
535             $bigrams = 1;
536             system ("perl", "$DIR/gen_bigrams.prl") and die "Failed $DIR/gen_bigrams.prl: $!\n";
537         }
538         else {
539             $bigrams = 0;
540         }
541 
542             print <<"EndOfMenu";
543 
544             ============================================================
545             ==     Welcome to the 600.466 Vector-based IR Engine
546             ==
547             ==                 Program Initialization
548             ============================================================
549 
550             System setup ...
551 
552             OPTIONS:
553               1 = Run the program using default model parameters
554               2 = Manually set the model parameters
555               3 = Print a table with all the model parameter permutations
556 
557               0 = Quit
558 
559             ============================================================
560 
561     EndOfMenu
562         ;
563 
564         $option = <STDIN>;
565         chomp $option;
566         if ($option !~ /[0-3]/) {
567             $option = 1;
568         }
569 
570         if($option == 0) {
571             exit 0;
572         }
573         elsif($option == 3) {
574             &print_full_table;
575         }
576         elsif($option == 2) {
577             # Manually initialize the program
578             print <<"EndOfMenu";
579 
580             ============================================================
581             ==     Welcome to the 600.466 Vector-based IR Engine
582             ==
583             ==                 Choose Method Parameter
584             ============================================================
585 
586             Please choose the method parameters that you want to modify ...
587 
588             OPTIONS:
589               1 = Term weighting permutations
590               2 = Similarity measures
591               3 = Stemming
592               4 = Stopwords
593               5 = Region weighting
594 
595               0 = Quit
596 
597             ============================================================
598 
599     EndOfMenu
600             ;
601             $option = <STDIN>;
602             chomp $option;
603             if ($option !~ /[0-5]/) {
604                 $option = 1;
605             }
606 
607             if($option == 0) {
608                 exit 0;
609             }
610             elsif($option == 1) { # Change Term weighting permutations
611                 print <<"EndOfMenu";
612 
613             ============================================================
614             ==     Welcome to the 600.466 Vector-based IR Engine
615             ==
616             ==              Term weighting permutations
617             ==                  Choose Permutation
618             ============================================================
619 
620             Please choose the permutation you want to use ...
621 
```

```
622           OPTIONS:
623             1 = Raw TF weighting
624             2 = * TF IDF weighting
625             3 = Boolean weighting
626
627             0 = Quit
628
629           ============================================================
630
631     EndOfMenu
632                 ;
633                 $option = <STDIN>;
634                 chomp $option;
635                 if ($option !~ /[0-5]/) {
636                     $option = 2;
637                 }
638
639                 if($option == 0) {
640                     exit 0;
641                 }
642                 elsif($option == 1) {
643                     # RAW TF weighting
644
645                     $method = "RAW TF";
646
647                     # initializarion
648                     &init_files ( "stemmed" );
649                     &init_corp_freq;
650
651                     print "INITIALIZING VECTORS ... \n";
652
653                     # init docs vector
654                     my $token_docs_fh = new FileHandle $token_docs, "r"
655                     or croak "Failed $token_docs";
656
657                     my $word    = undef;
658
659                     my $doc_num =  0;    # current document number and total docs at end
660                     my $tweight =  0;    # current weight assigned to document token
661
662                     push @doc_vector, { };     # push one empty value onto @doc_vector so that
663                                                # indices correspond with document numbers
664
665                     while (defined( $word = <$token_docs_fh> )) {
666
667                         chomp $word;
668
669                         last if $word =~ /^\.I 0/; # indicates end of file so kick out
670
671                         if ($word =~ /^\.I/) {     # indicates start of a new document
672
673                             push @doc_vector, { };
674                             $doc_num++;
675
676                             next;
677                         }
678
679                         $tweight = $TITLE_BASE_WEIGHT and next if $word =~ /^\.T/;
680                         $tweight = $KEYWD_BASE_WEIGHT and next if $word =~ /^\.K/;
681                         $tweight = $ABSTR_BASE_WEIGHT and next if $word =~ /^\.W/;
682                         $tweight = $AUTHR_BASE_WEIGHT and next if $word =~ /^\.A/;
683
684                         if ($word =~ /[a-zA-Z]/ and ! exists $stoplist_hash{ $word }) {
685
686                             if (defined( $docs_freq_hash{ $word } )) {
687
688                                 $doc_vector[$doc_num]{ $word } += $tweight;
689                             }
690                             else {
691                                 print "ERROR: Document frequency of zero: ", $word, "\n";
692                             }
693                         }
694                     }
695
696                     $total_docs = $doc_num;
697
698                     # init query vector
699                     my $QUERY_BASE_WEIGHT = 2;
700                     my $QUERY_AUTH_WEIGHT = 2;
701
702                     my $token_qrys_fh = new FileHandle $token_qrys, "r"
703                         or croak "Failed $token_qrys";
704
705                     $word = undef;
706
707                     $tweight =  0;
708                     my $qry_num =  0;
709
710                     push @qry_vector, { };    # push one empty value onto @qry_vectors so that
```

```
                                    # indices correspond with query numbers

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

            chomp $word;

            if ($word =~ /^\.I/) {

                push @qry_vector, { };
                $qry_num++;

                next;
            }

            $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
            $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;

            if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {

                if (! exists $docs_freq_hash{ $word }) {
                    print "ERROR: Document frequency of zero: ", $word, "\n";
                }
                else {
                    $qry_vector[$qry_num]{ $word } += $tweight;
                }
            }
        }

        $total_qrys = $qry_num;


    }
    elsif($option == 3) {
        # Boolean weighting

        $method = "Boolean";

        # initializarion
        &init_files ( "stemmed" );
        &init_corp_freq;

        print "INITIALIZING VECTORS ... \n";

        # init docs vector
        my $token_docs_fh = new FileHandle $token_docs, "r"
        or croak "Failed $token_docs";

        my $word    = undef;

        my $doc_num =  0;    # current document number and total docs at end
        my $tweight =  0;    # current weight assigned to document token

        push @doc_vector, { };     # push one empty value onto @doc_vector so that
                                   # indices correspond with document numbers

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

            chomp $word;

            last if $word =~ /^\.I 0/; # indicates end of file so kick out

            if ($word =~ /^\.I/) {     # indicates start of a new document

                push @doc_vector, { };
                $doc_num++;

                next;
            }

            $tweight = $TITLE_BASE_WEIGHT and next if $word =~ /^\.T/;
            $tweight = $KEYWD_BASE_WEIGHT and next if $word =~ /^\.K/;
            $tweight = $ABSTR_BASE_WEIGHT and next if $word =~ /^\.W/;
            $tweight = $AUTHR_BASE_WEIGHT and next if $word =~ /^\.A/;

            if ($word =~ /[a-zA-Z]/ and ! exists $stoplist_hash{ $word }) {

                if (defined( $docs_freq_hash{ $word } )) {

                    $doc_vector[$doc_num]{ $word } = 1;
                }
                else {
                    $doc_vector[$doc_num]{ $word } = 0;
                    # print "ERROR: Document frequency of zero: ", $word, "\n";
                }
            }
        }

        $total_docs = $doc_num;

```

```
800                 # init query vector
801                 my $QUERY_BASE_WEIGHT = 2;
802                 my $QUERY_AUTH_WEIGHT = 2;
803
804                 my $token_qrys_fh = new FileHandle $token_qrys, "r"
805                     or croak "Failed $token_qrys";
806
807                 $word = undef;
808
809                 $tweight =  0;
810                 my $qry_num =  0;
811
812                 push @qry_vector, { };    # push one empty value onto @qry_vectors so that
813                                           # indices correspond with query numbers
814
815                 while (defined( $word = <$token_qrys_fh> )) {
816
817                     chomp $word;
818
819                     if ($word =~ /^\.I/) {
820
821                         push @qry_vector, { };
822                         $qry_num++;
823
824                         next;
825                     }
826
827                     $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
828                     $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;
829
830                     if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {
831
832                         if (! exists $docs_freq_hash{ $word }) {
833                             $qry_vector[$qry_num]{ $word } = 0;
834                         }
835                         else {
836                             $qry_vector[$qry_num]{ $word } = 1;
837                         }
838                     }
839                 }
840                 $total_qrys = $qry_num;
841             }
842             else {
843                 # use default * TF IDF weighting
844                 $method = "Default";
845
846                 # initializarion
847                 &init_files ( "stemmed" );
848                 &init_corp_freq;
849
850                 print "INITIALIZING VECTORS ... \n";
851
852                 $total_docs = &init_doc_vectors;
853                 $total_qrys = &init_qry_vectors;
854             }
855         }
856         elsif($option == 2) { # Change Similarity measures
857             print <<"EndOfMenu";
858
859         ============================================================
860         ==     Welcome to the 600.466 Vector-based IR Engine
861         ==
862         ==                 Similarity  measures
863         ==                  Choose Permutation
864         ============================================================
865
866         Please choose the permutation you want to use ...
867
868         OPTIONS:
869           1 = * Cosine similarity
870           2 = Dice similarity
871           3 = Jaccard similarity
872           4 = Overlap similarity
873
874           0 = Quit
875
876         ============================================================
877
878     EndOfMenu
879             ;
880             $option = <STDIN>;
881             chomp $option;
882             if ($option !~ /[0-4]/) {
883                 $option = 1;
884             }
885
886             if($option == 0) {
887                 exit 0;
888             }
```

```
            elsif($option == 2) {
                # use Dice similarity
                $method = "Dice";
                $sim = "dice";
            }
            elsif($option == 3) {
                # use Jaccard similarity
                $method = "Jaccard";
                $sim = "jaccard";
            }
            elsif($option == 4) {
                # use Overlap similarity
                $method = "overlap";
                $sim = "overlap";
            }
            else {
                # use default * Cosine similarity
                $method = "Default";
            }
            # initializarion
            &init_files ( "stemmed" );
            &init_corp_freq;

            print "INITIALIZING VECTORS ... \n";

            $total_docs = &init_doc_vectors;
            $total_qrys = &init_qry_vectors;
        }
        elsif($option == 3) { # Change Stemming
            print <<"EndOfMenu";

        ============================================================
        ==     Welcome to the 600.466 Vector-based IR Engine
        ==
        ==                       Stemming
        ==                   Choose Permutation
        ============================================================

        Please choose the permutation you want to use ...

        OPTIONS:
          1 = Use raw, unstemmed tokens (all converted to lower case)
          2 = * Use tokends stemmend by the Porter stemmer

          0 = Quit

        ============================================================

EndOfMenu
            ;
            $option = <STDIN>;
            chomp $option;
            if ($option !~ /[0-2]/) {
                $option = 2;
            }

            if($option == 0) {
                exit 0;
            }
            elsif($option == 1) {
                # use raw, unstemmed tokens (all converted to lower case)
                $method = "Unstem";

                # init_files using unstemmed
                &init_files ( "unstemmed" );
            }
            else {
                # use default * tokends stemmend by the Porter stemmer
                $method = "Default";

                # init_files using stemmed
                &init_files ( "stemmed" );
            }
            &init_corp_freq;

            print "INITIALIZING VECTORS ... \n";

            $total_docs = &init_doc_vectors;
            $total_qrys = &init_qry_vectors;
        }
        elsif($option == 4) { # Change Stopwords
            print <<"EndOfMenu";

        ============================================================
        ==     Welcome to the 600.466 Vector-based IR Engine
        ==
        ==                       Stopwords
        ==                   Choose Permutation
        ============================================================
```

```
978
979           Please choose the permutation you want to use ...
980
981           OPTIONS:
982             1 = * Exclude stopwords from term vectors
983             2 = Include all tokens, including punctuation
984
985             0 = Quit
986
987           ============================================================
988
989     EndOfMenu
990                 ;
991                 $option = <STDIN>;
992                 chomp $option;
993                 if ($option !~ /[0-2]/) {
994                     $option = 1;
995                 }
996
997                 if($option == 0) {
998                     exit 0;
999                 }
1000                elsif($option == 2) {
1001                    # Include all tokens, including punctuation
1002                    $method = "No stwd";
1003
1004                    # initializarion
1005                    &init_files ( "stemmed" );
1006                    &init_corp_freq;
1007
1008                    %stoplist_hash = ();
1009                    print "INITIALIZING VECTORS ... \n";
1010
1011                    $total_docs = &init_doc_vectors;
1012                    $total_qrys = &init_qry_vectors;
1013                }
1014                else {
1015                    # use default * tokends stemmend by the Porter stemmer
1016                    $method = "Default";
1017
1018                    # initializarion
1019                    &init_files ( "stemmed" );
1020                    &init_corp_freq;
1021
1022                    print "INITIALIZING VECTORS ... \n";
1023
1024                    $total_docs = &init_doc_vectors;
1025                    $total_qrys = &init_qry_vectors;
1026                }
1027            }
1028            elsif($option == 5) { # Change Region weighting
1029                print <<"EndOfMenu";
1030
1031           ============================================================
1032           ==     Welcome to the 600.466 Vector-based IR Engine
1033           ==
1034           ==                    Region weighting
1035           ==                   Choose Permutation
1036           ============================================================
1037
1038           Please choose the permutation you want to use ...
1039
1040           OPTIONS:
1041             1 = Weight titles, keywords, author list and abstract words equally
1042             2 = * Use relative weights of titles=3x, keywords=4x, author list=3x, abstract=1x
1043             3 = Use relative weights of titles=1x, keywords=1x, author list=1x, abstract=4x
1044
1045             0 = Quit
1046
1047           ============================================================
1048
1049     EndOfMenu
1050                 ;
1051                 $option = <STDIN>;
1052                 chomp $option;
1053                 if ($option !~ /[0-3]/) {
1054                     $option = 2;
1055                 }
1056
1057                 if($option == 0) {
1058                     exit 0;
1059                 }
1060                 elsif($option == 1) {
1061                     # Weight titles, keywords, author list and abstract words equally
1062                     $method = "Reg 1111";
1063
1064                     # reset global vars for the weight
1065                     $TITLE_BASE_WEIGHT = 1;      # weight given a title token
1066                     $KEYWD_BASE_WEIGHT = 1;      # weight given a key word token
```

```
1067                    $AUTHR_BASE_WEIGHT = 1;     # weight given an an author token
1068                    $ABSTR_BASE_WEIGHT = 1;     # weight given an abstract word token
1069                }
1070                elsif($option == 3) {
1071                    # Weight titles, keywords, author list and abstract words equally
1072                    $method = "Reg 1114";
1073
1074                    # reset global vars for the weight
1075                    $TITLE_BASE_WEIGHT = 1;     # weight given a title token
1076                    $KEYWD_BASE_WEIGHT = 1;     # weight given a key word token
1077                    $AUTHR_BASE_WEIGHT = 1;     # weight given an an author token
1078                    $ABSTR_BASE_WEIGHT = 4;     # weight given an abstract word token
1079                }
1080                else {
1081                    # use default * tokends stemmend by the Porter stemmer
1082                    $method = "Default";
1083                }
1084
1085                # initializarion
1086                &init_files ( "stemmed" );
1087                &init_corp_freq;
1088
1089                print "INITIALIZING VECTORS ... \n";
1090
1091                $total_docs = &init_doc_vectors;
1092                $total_qrys = &init_qry_vectors;
1093            }
1094            else {
1095                $method = "Default";
1096
1097                # initializarion
1098                &init_files ( "stemmed" );
1099                &init_corp_freq;
1100
1101                print "INITIALIZING VECTORS ... \n";
1102
1103                $total_docs = &init_doc_vectors;
1104                $total_qrys = &init_qry_vectors;
1105            }
1106        }
1107        else {
1108            $method = "Default";
1109
1110            # initializarion
1111            &init_files ( "stemmed" );
1112            &init_corp_freq;
1113
1114            print "INITIALIZING VECTORS ... \n";
1115
1116            $total_docs = &init_doc_vectors;
1117            $total_qrys = &init_qry_vectors;
1118        }
1119    }
1120
1121    ##########################################################
1122    ## MAIN_LOOP
1123    ##
1124    ## Parameters: currently no explicit parameters.
1125    ##             performance dictated by user imput.
1126    ##
1127    ## Initializes document and query vectors using the
1128    ## input files specified in &init_files. Then offers
1129    ## a menu and switch to appropriate functions in an
1130    ## endless loop.
1131    ##
1132    ## Possible extensions at this level:  prompt the user
1133    ## to specify additional system parameters, such as the
1134    ## similarity function to be used.
1135    ##
1136    ## Currently, the key parameters to the system (stemmed/unstemmed,
1137    ## stoplist/no-stoplist, term weighting functions, vector
1138    ## similarity functions) are hardwired in.
1139    ##
1140    ## Initializing the document vectors is clearly the
1141    ## most time consuming section of the program, as 213334
1142    ## to 258429 tokens must be processed, weighted and added
1143    ## to dynamically growing vectors.
1144    ##
1145    ##########################################################
1146
1147    sub main_loop {
1148        # original initializarion
1149        # &init_files ( "stemmed" );
1150        # &init_corp_freq;
1151
1152        # print "INITIALIZING VECTORS ... \n";
1153
1154        # my $total_docs = &init_doc_vectors;
1155        # my $total_qrys = &init_qry_vectors;
```

```
1156        # my $option = undef;
1157
1158        # Customized init
1159        &init_method;
1160
1161    # * Below is the original program ****************************************
1162
1163        while (1) {
1164            print <<"EndOfMenu";
1165
1166            ============================================================
1167            ==     Welcome to the 600.466 Vector-based IR Engine
1168            ==
1169            == Total Documents: $total_docs
1170            == Total Queries:   $total_qrys
1171            ==
1172            == Permutation Name:   $method
1173            ============================================================
1174
1175            OPTIONS:
1176              1 = Find documents most similar to a given query or document
1177              2 = Compute precision/recall for the full query set
1178              3 = Compute cosine similarity between two queries/documents
1179
1180              0 = Quit
1181
1182            ============================================================
1183
1184    EndOfMenu
1185        ;
1186
1187            print "Enter Option: ";
1188
1189            my $option = <STDIN>;
1190            chomp $option;
1191            if ($option !~ /[0-3]/) {
1192                $option = 1;
1193            }
1194            exit 0 if $option == 0;
1195
1196            &full_precision_recall_test and next if $option == 2;
1197            &do_full_cosine_similarity  and next if $option == 3;
1198
1199            # default and choice 1 is
1200
1201            &get_and_show_retrieved_set and next if $option == 1;
1202        }
1203    }
1204
1205
1206    ##########################################################
1207    ## GET_AND_SHOW_RETRIEVED_SET
1208    ##
1209    ##  This function requests key retrieval parameters,
1210    ##  including:
1211    ##
1212    ##  A) Is a query vector or document vector being used
1213    ##     as the retrieval seed? Both are vector representations
1214    ##     but they are stored in different data structures,
1215    ##     and one may optionally want to treat them slightly
1216    ##     differently.
1217    ##
1218    ##  B) Enter the number of the query or document vector to
1219    ##     be used as the retrieval seed.
1220    ##
1221    ##     Alternately, one may wish to request a new query
1222    ##     from standard input here (and call the appropriate
1223    ##     tokenization, stemming and term-weighting routines).
1224    ##
1225    ##  C) Request the maximum number of retrieved documents
1226    ##     to display.
1227    ##
1228    ##  Perl note: one reads a line from a file <FILE> or <STDIN>
1229    ##             by the assignment $string=<STDIN>; Beware of
1230    ##             string equality testing, as these strings
1231    ##             will have a newline (\n) attached.
1232    ##########################################################
1233
1234    sub get_and_show_retrieved_set {
1235
1236        print << "EndOfMenu";
1237
1238        Find documents similar to:
1239            (1) a query from 'query.raw'
1240            (2) an interactive query
1241            (3) another document
1242    EndOfMenu
1243        ;
1244
```

```
1245        print "Choice: ";
1246
1247        my    $comp_type = <STDIN>;
1248        chomp $comp_type;
1249
1250        if   ($comp_type !~ /^[1-3]$/) { $comp_type = 1; }
1251
1252        print "\n";
1253
1254
1255        # if not an interactive query than we need to retrieve which
1256        # query/document we want to use from the corpus
1257
1258        my $vect_num = 1;
1259
1260        if ($comp_type != 2) {
1261            print "Target Document/Query number: ";
1262
1263                  $vect_num  = <STDIN>;
1264            chomp $vect_num;
1265
1266            if   ($vect_num !~ /^[1-9]/) { $vect_num  = 1; }
1267
1268            print "\n";
1269        }
1270
1271
1272        print "Show how many matching documents (20): ";
1273
1274        my    $max_show  = <STDIN>;
1275        chomp $max_show;
1276
1277        if   ($max_show !~ /[0-9]/) { $max_show  = 20; }
1278
1279        if    ($comp_type == 3) {
1280
1281            print "Document to Document comparison\n";
1282
1283            &get_retrieved_set( $doc_vector[$vect_num] );
1284            &shw_retrieved_set( $max_show,
1285                                $vect_num,
1286                                $doc_vector[$vect_num],
1287                                "Document" );
1288        }
1289        elsif ($comp_type == 2) {
1290
1291            print "Interactive Query to Document comparison\n";
1292
1293            my $int_vector = &set_interact_vec;  # vector created by interactive
1294                                                 #  query
1295
1296            &get_retrieved_set( $int_vector );
1297            &shw_retrieved_set( $max_show,
1298                                0,
1299                                $int_vector,
1300                                "Interactive Query" );
1301        }
1302        else {
1303
1304            print "Query to Document comparison\n";
1305            # print "\n\n\n$qry_vector[$vect_num]\n\n\n";
1306            &get_retrieved_set( $qry_vector[$vect_num] );
1307            # print "\n\n\nXXX\n\n\n";
1308            &shw_retrieved_set( $max_show,
1309                                $vect_num,
1310                                $qry_vector[$vect_num],
1311                                "Query" );
1312            &comp_recall($vect_num );
1313            &show_relvnt( $vect_num,
1314                          $qry_vector[$vect_num],
1315                          "Query" );
1316        }
1317    }
1318
1319    ##########################################################
1320    ## SET_INTERACT_VEC
1321    ##
1322    ##  Initialize the vector for interactive queries
1323    ##########################################################
1324
1325    sub set_interact_vec {
1326
1327        system ("perl", "$DIR/interactive.prl") and die "Failed $DIR/interactive.prl: $!\n";
1328
1329        my $QUERY_BASE_WEIGHT = 2;
1330        my $QUERY_AUTH_WEIGHT = 2;
1331
1332        my $token_qrys_fh = new FileHandle $token_intr, "r"
1333            or croak "Failed $token_intr";
```

```
1334
1335        my $int_vector = { };
1336        my $word       = undef;
1337
1338        my $tweight =  0;
1339        my $qry_num =  0;
1340
1341        while (defined( $word = <$token_qrys_fh> )) {
1342
1343            chomp $word;
1344            print $word, "\n";
1345
1346            next if $word =~ /^\.I/;   # start of query tokens
1347
1348            $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
1349            $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;
1350
1351            if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {
1352
1353                if (! exists $docs_freq_hash{ $word }) {
1354                    print "ERROR: Document frequency of zero: ", $word, "\n";
1355                }
1356                else {
1357                    $$int_vector{ $word } += $tweight;
1358                }
1359            }
1360        }
1361
1362        # calculate RAW TF on qry_vector
1363        foreach my $word (keys $int_vector) {
1364            $$int_vector{ $word } = ($$int_vector{ $word } * log($total_docs/ $docs_freq_hash{ $word }));
1365        }
1366
1367        return $int_vector
1368    }
1369
1370
1371    ##########################################################
1372    ## GET_RETRIEVED_SET
1373    ##
1374    ##  Parameters:
1375    ##
1376    ##  $qry_vector{} - the query vector to be compared with the
1377    ##                  document set. May also be another document
1378    ##                  vector.
1379    ##
1380    ##  This function computes the document similarity between the
1381    ##  given vector $qry_vector{} and all vectors in the document
1382    ##  collection storing these values in the array @doc_simula
1383    ##
1384    ##  An array of the document numbers is then sorted by this
1385    ##  similarity function, forming the rank order of documents
1386    ##  for use in the retrieval set.
1387    ##
1388    ##  The -1 in the simcomp similarity comparision function
1389    ##  makes the sorted list in descending order.
1390    ##########################################################
1391
1392    sub get_retrieved_set {
1393
1394        my $qry_vector = shift;
1395        my $tot_number = (scalar @doc_vector) - 1;
1396        my $index      = 0;
1397
1398        @doc_simula = ( );   # insure that storage vectors are empty before we
1399        @res_vector = ( );   # calculate vector similarities
1400
1401        push @doc_simula, 0.0;    # push one empty value so that indices
1402                                  # correspond with document values
1403
1404        for $index ( 1 .. $tot_number) {
1405            # print "$index, ";
1406            if($sim eq "dice") {
1407                push @doc_simula, &dice_sim( $qry_vector, $doc_vector[$index] );
1408            }
1409            elsif($sim eq "jaccard") {
1410                push @doc_simula, &jaccard_sim( $qry_vector, $doc_vector[$index] );
1411            }
1412            elsif($sim eq "overlap") {
1413                push @doc_simula, &overlap_sim( $qry_vector, $doc_vector[$index] );
1414            }
1415            else {
1416                push @doc_simula, &cosine_sim_a( $qry_vector, $doc_vector[$index] );
1417            }
1418        }
1419        @res_vector =
1420          sort { -1 * ($doc_simula[$a] <=> $doc_simula[$b]); } 1 .. $tot_number;
1421        # print "@res_vector\n\n\n\n";
1422        # print "\n\n\n";
```

```
1423    }
1424
1425    ##########################################################
1426    ## SHW_RETRIEVED_SET
1427    ##
1428    ## Assumes the following global data structures have been
1429    ## initialized, based on the results of &get_retrieved_set.
1430    ##
1431    ## 1) @res_vector - contains the document numbers sorted in
1432    ##                  rank order
1433    ## 2) @doc_simula - The similarity measure for each document,
1434    ##                  computed by &get_retrieved_set.
1435    ##
1436    ## Also assumes that the following have been initialized in
1437    ## advance:
1438    ##
1439    ##       $titles[ $doc_num ]    - the document title for a
1440    ##                                document number, $doc_num
1441    ##       $relevance_hash{ $qry_num }{ $doc_num }
1442    ##                              - is $doc_num relevant given
1443    ##                                query number, $qry_num
1444    ##
1445    ## Parameters:
1446    ##   $max_show   - the maximum number of matched documents
1447    ##                 to display.
1448    ##   $qry_num    - the vector number of the query
1449    ##   $qry_vect   - the query vector (passed by reference)
1450    ##   $comparison - "Query" or "Document" (type of vector
1451    ##                 being compared to)
1452    ##
1453    ## In the case of "Query"-based retrieval, the relevance
1454    ## judgements for the returned set are displayed. This is
1455    ## ignored when doing document-to-document comparisons, as
1456    ## there are nor relevance judgements.
1457    ##
1458    ##########################################################
1459
1460    sub shw_retrieved_set {
1461
1462        my $max_show   = shift;
1463        my $qry_num    = shift;
1464        my $qry_vect   = shift;
1465        my $comparison = shift;
1466
1467        print << "EndOfList";
1468
1469        ************************************************************
1470            Documents Most Similar To $comparison number $qry_num
1471        ************************************************************
1472        Similarity   Doc#  Author      Title
1473        ==========   ==== ========     ============================
1474
1475    EndOfList
1476        ;
1477
1478        my $rel_num = ($qry_num =~ /^\d$/) ? "0$qry_num" : $qry_num;
1479        my $index   = 0;
1480
1481        for $index ( 0 .. $max_show ) {
1482            my $ind = $res_vector[$index];
1483
1484            if (($comparison =~ /Query/) and
1485                ($relevance_hash{ $rel_num }{ $ind })) {
1486                print "\* ";
1487            }
1488            else {
1489                print "  ";
1490            }
1491            my ($similarity) = ($doc_simula[$ind]    =~ /^([0-9]+\.\d{0,8})/);
1492            if($doc_simula[$ind] == 0 or $doc_simula[$ind] == 1) {
1493                  $similarity = sprintf "%0.8f", $doc_simula[$ind];
1494                }
1495            my  $title       = substr $titles_vector[$ind], 0, 47;
1496
1497            print "  ", $similarity, "   ", $title, "\n";
1498        }
1499
1500        print "\n";
1501        print "Show the terms that overlap between the query and ";
1502        print "retrieved docs (y/n): ";
1503
1504        my  $show_terms = <STDIN>;
1505        if ($show_terms !~ /[nN]/) {
1506
1507            my $index = 0;
1508
1509            for $index ( 0 .. $max_show ) {
1510                my $ind = $res_vector[$index];
1511
```

```
            show_overlap( $qry_vect,
                          $doc_vector[$ind],
                          $qry_num,
                          $ind );

            if ($index % 5 == 4) {

                print "\n";
                print "Continue (y/n)? ";

                my  $cont = <STDIN>;
                if ($cont =~ /[nN]/) {
                    last;
                }
            }
        }
    }
}


#########################################################
## COMPUTE_PREC_RECALL
##
## Like &shw_retrieved_set, this function makes use of the following
## data structures which may either be passed as parameters or
## used as global variables. These values are set by the function
## &get_retrieved_set.
##
## 1) doc_simula[ $rank ] - contains the document numbers sorted
##                          in rank order based on the results of
##                          the similarity function
##
## 2) res_vector[ $docn ] - The similarity measure for each document,
##                          relative to the query vector ( computed by
##                          &get_retrieved_set).
##
## Also assumes that the following have been initialzied in advance:
##       $titles[ $docn ]       - the document title for a document
##                                number $docn
##       $relevance_hash{ $qvn }{ $docn }
##                              - is $docn relevant given query number
##                                $qvn
##
##  The first step of this function should be to take the rank ordering
##  of the documents given a similarity measure to a query
##  (i.e. the list docs_sorted_by_similarity[$rank]) and make a list
##  of the ranks of just the relevant documents. In an ideal world,
##  if there are k=8 relevant documents for a query, for example, the list
##  of rank orders should be (1 2 3 4 5 6 7 8) - i.e. the relevant documents
##  are the top 8 entries of all documents sorted by similarity.
##  However, in real life the relevant documents may be ordered
##  much lower in the similarity list, with rank orders of
##  the 8 relevant of, for example, (3 27 51 133 159 220 290 1821).
##
##  Given this list, compute the k (e.g. 8) recall/precison pairs for
##  the list (as discussed in class). Then to determine precision
##  at fixed levels of recall, either identify the closest recall
##  level represented in the list and use that precision, or
##  do linear interpolation between the closest values.
##
##  This function should also either return the various measures
##  of precision/recall specified in the assignment, or store
##  these values in a cumulative sum for later averaging.
#########################################################

sub comp_recall {
    # initial the global vars here everytime we compute
    %rank_docn = ();
    %rec_prec = ();

    # print "Entering the comp_recal function!\n";
    my $qry_num    = shift;
    my $rel_num = ($qry_num =~ /^\d$/) ? "0$qry_num" : $qry_num;
    my $total_relevant = keys $relevance_hash{$rel_num};
    my $i = 1;

    my $rank_sum = 0;   # sum of the i ranks
    my $rank_sum_log = 0;       # sum of the i log ranks
    my $i_sum = 0;      # sum of i .. TOTAL_RELEVANT
    my $i_sum_log = 0;  # sum of log i .. TOTAL_RELEVANT

    while(my ($key,$value) = each($relevance_hash{$rel_num})) {
        my $rank = (firstidx { $_ eq $key } @res_vector) + 1;
        $rank_docn{$rank} =  $key;
    }

    foreach my $rank (sort { $a <=> $b } keys %rank_docn) {
        my $rec = $i / $total_relevant;
        my $prec = $i / $rank;
```

```
1601            # print "$i\t$rank\t$rec\t$prec\n";
1602            $rec_prec{$rec} = $prec;
1603            $rank_sum += $rank;
1604            $i_sum += $i;
1605            $rank_sum_log += log($rank);
1606            $i_sum_log += log($i);
1607
1608            $i++;
1609        }
1610
1611        # calculate Recall_norm
1612        $recall_norm = 1 - (($rank_sum - $i_sum) / ($total_relevant * ($total_docs - $total_relevant)));
1613
1614        # calculate Prec_norm
1615        $prec_norm = 1 - (($rank_sum_log - $i_sum_log) / ($total_docs * log($total_docs) - ($total_docs - $total_relevant) * log
      ($total_docs - $total_relevant) - $total_relevant * log($total_relevant)));
1616
1617        # calculate Prec_mean1
1618        $prec_mean1 = ( (&get_prec(0.25) + &get_prec(0.50) + &get_prec(0.75)) / 3 );
1619
1620        # calculate Prec_mean2
1621        my $prec_mean2_sum = 0;
1622        for(my $i=0;$i<10;$i++) {
1623            $prec_mean2_sum += &get_prec($i/10);
1624        }
1625        $prec_mean2 = ( 0.1 * $prec_mean2_sum );
1626
1627        # test
1628        # print "\n\n\nprec_mean1\tprec_mean2\trecall_norm\tprec_norm\t\n\n";
1629        # print "$prec_mean1\t$prec_mean2\t$recall_norm\t$prec_norm\t";
1630    }
1631
1632    ##########################################################
1633    ## SHOW_RELVNT
1634    ##
1635    ## UNIMPLEMENTED
1636    ##
1637    ## This function should take the rank orders and similarity
1638    ## arrays described in &show_retrieved_set and &comp_recall
1639    ## and print out only the relevant documents, in an order
1640    ## and manner of presentation very similar to &show_retrieved_set.
1641    ##########################################################
1642
1643    sub show_relvnt {
1644        # print "To be implemented\n";
1645        my $qry_num    = shift;
1646        my $qry_vect   = shift;
1647        my $comparison = shift;
1648
1649        print << "EndOfList";
1650
1651        ************************************************************
1652            All Relevant Documents To $comparison number $qry_num
1653        ************************************************************
1654        Similarity   Doc#  Author      Title
1655        ==========   ==== ========     ============================
1656
1657    EndOfList
1658        ;
1659
1660        my $rel_num = ($qry_num =~ /^\d$/) ? "0$qry_num" : $qry_num;
1661        my $index   = 0;
1662        my @rel_array = ();
1663
1664        for $index ( 0 .. $#res_vector ) {
1665            my $ind = $res_vector[$index];
1666
1667            if (($comparison =~ /Query/) and
1668                ($relevance_hash{ $rel_num }{ $ind })) {
1669                push (@rel_array, $index);
1670                print "\* ";
1671                my ($similarity) = ($doc_simula[$ind]    =~ /^([0-9]+\.\d{0,8})/);
1672                if($doc_simula[$ind] == 0) {
1673                    $similarity = sprintf "%0.8f", $doc_simula[$ind];
1674                }
1675                my  $title       = substr $titles_vector[$ind], 0, 47;
1676
1677                print "  ", $similarity, "   ", $title, "\n";
1678            }
1679
1680
1681        }
1682
1683        print "\n";
1684        print "Show the terms that overlap between the query and ";
1685        print "retrieved docs (y/n): ";
1686
1687        my  $show_terms = <STDIN>;
1688        if ($show_terms !~ /[nN]/) {
```

```
1689
1690            my $index = 0;
1691
1692            for $index ( 0 .. $#rel_array ) {
1693                my $ind = $res_vector[$rel_array[$index]];
1694
1695                show_overlap( $qry_vect,
1696                              $doc_vector[$ind],
1697                              $qry_num,
1698                              $ind );
1699
1700                if ($index % 5 == 4) {
1701
1702                    print "\n";
1703                    print "Continue (y/n)? ";
1704
1705                    my  $cont = <STDIN>;
1706                    if ($cont =~ /[nN]/) {
1707                        last;
1708                    }
1709                }
1710            }
1711        }
1712    }
1713
1714
1715    ######################################################
1716    ## SHOW_OVERLAP
1717    ##
1718    ## Parameters:
1719    ##  - Two vectors ($qry_vect and $doc_vect), passed by
1720    ##    reference.
1721    ##  - The number of the vectors for display purposes
1722    ##
1723    ## PARTIALLY IMPLEMENTED:
1724    ##
1725    ## This function should show the terms that two vectors
1726    ## have in common, the relative weights of these terms
1727    ## in the two vectors, and any additional useful information
1728    ## such as the document frequency of the terms, etc.
1729    ##
1730    ## Useful for understanding the reason why documents
1731    ## are judged as relevant.
1732    ##
1733    ## Present in a sorted order most informative to the user.
1734    ##
1735    ######################################################
1736
1737    sub show_overlap {
1738
1739        my $qry_vect = shift;
1740        my $doc_vect = shift;
1741        my $qry_num  = shift;
1742        my $doc_num  = shift;
1743
1744        print "============================================================\n";
1745        printf( "%-15s  %8d   %8d\t%s\n",
1746               "Vector Overlap",
1747               $qry_num        ,
1748               $doc_num        ,
1749               "Docfreq"       );
1750        print "============================================================\n";
1751
1752        my $term_one   = undef;
1753        my $weight_one = undef;
1754
1755        while (($term_one, $weight_one) = each %{ $qry_vect }) {
1756            if (exists $$doc_vect{ $term_one }) {
1757
1758                printf( "%-15s  %8d   %8d\t%d\n"    ,
1759                       $term_one                    ,
1760                       $weight_one                  ,
1761                       $$doc_vect{ $term_one }      ,
1762                       $docs_freq_hash{ $term_one } );
1763            }
1764        }
1765    }
1766
1767
1768    ######################################################
1769    ## DO_FULL_COSINE_SIMILARITY
1770    ##
1771    ##  Prompts for a document number and query number,
1772    ##  and then calls a function to show similarity.
1773    ##
1774    ##  Could/should be expanded to handle a variety of
1775    ##  similarity measures.
1776    ######################################################
1777
```

```
1778    sub do_full_cosine_similarity {
1779
1780        print "\n";
1781        print "1st Document number: ";
1782
1783        my    $num_one = <STDIN>;
1784        chomp $num_one;
1785
1786        print "\n";
1787        print "2nd Document number: ";
1788
1789        my    $num_two = <STDIN>;
1790        chomp $num_two;
1791
1792        $num_one = 1 if $num_one !~ /[0-9]/;
1793        $num_two = 1 if $num_two !~ /[0-9]/;
1794        full_cosine_similarity( $doc_vector[$num_one],
1795                                $doc_vector[$num_two],
1796                                $num_one,
1797                                $num_two );
1798    }
1799
1800
1801    ######################################################
1802    ## FULL_COSINE_SIMILARITY
1803    ##
1804    ## UNIMPLEMENTED
1805    ##
1806    ## This function should compute cosine similarity between
1807    ## two vectors and display the information that went into
1808    ## this calculation, useful for debugging purposes.
1809    ## Similar in structure to &show_overlap.
1810    ######################################################
1811
1812    sub full_cosine_similarity {
1813
1814        my $qry_vect = shift;
1815        my $doc_vect = shift;
1816        my $qry_num = shift;
1817        my $doc_num = shift;
1818
1819        # print "The rest is up to you . . . \n";
1820        @doc_simula = ( );   # insure that storage vectors are empty before we
1821        @res_vector = ( );   # calculate vector similarities
1822
1823        my $similarity = cosine_sim_a( $qry_vect, $doc_vect );
1824
1825        print << "EndOfList";
1826
1827        ************************************************************
1828            Similarity of Document number $qry_num and $doc_num
1829        ************************************************************
1830        Similarity   Doc#  Author      Title
1831        ==========   ==== ========     ============================
1832
1833    EndOfList
1834        ;
1835
1836        print "  ";
1837        my $title       = substr $titles_vector[$qry_num], 0, 47;
1838        print "  ", (sprintf "%0.9f", $similarity), "   ", $title, "\n";
1839        print "  ";
1840        $title       = substr $titles_vector[$doc_num], 0, 47;
1841        print "  ", (sprintf "%0.9f", $similarity), "   ", $title, "\n";
1842
1843        print "\n";
1844        print "Show the terms that overlap between the query and ";
1845        print "retrieved docs (y/n): ";
1846
1847        my  $show_terms = <STDIN>;
1848        if ($show_terms !~ /[nN]/) {
1849            print "============================================================\n";
1850            printf( "%-15s  %8d   %8d\t%s\n",
1851                   "Vector Overlap",
1852                   $qry_num        ,
1853                   $doc_num        ,
1854                   "Docfreq"       );
1855            print "============================================================\n";
1856
1857            my $term_one   = undef;
1858            my $weight_one = undef;
1859
1860            while (($term_one, $weight_one) = each %{ $qry_vect }) {
1861                if (exists $$doc_vect{ $term_one }) {
1862
1863                    printf( "%-15s  %8d   %8d\t%d\n"    ,
1864                           $term_one                    ,
1865                           $weight_one                  ,
1866                           $$doc_vect{ $term_one }      ,
```

```
                            $docs_freq_hash{ $term_one } );
                }
            }
        }
    }


    ########################################################
    ##  FULL_PRECISION_RECALL_TEST
    ##
    ##  This function should test the various precision/recall
    ##  measures discussed in the assignment and store cumulative
    ##  statistics over all queries.
    ##
    ##  As each query takes a few seconds to process, print
    ##  some sort of feedback for each query so the user
    ##  has something to watch.
    ##
    ##  It is helpful to also log this information to a file.
    ########################################################

    sub full_precision_recall_test {
        # Suggestion: if using global variables to store cumulative
        #             statistics, initialize them here.

    #    for my $ind ( 1 .. $tot_queries ) {
    #
    #       &get_retrieved_set( $qry_vector[$ind] );
    #       &comp_recall( $relevance_hash{ $ind }, $ind );
    #
    #       # Suggestion: Collect cumulative statistics here or in
    #       #             global variables set in the above funtion
    #    }

        # Suggestion: Print some sort of summary here.
        # print "Entering full_precision_recall_test\n";
        print << "EndOfList";

                                    ************************************************************
                                    **      Precision / Recall averaged over 33 queries       **
                                    ************************************************************

        Permutation Name   \t\t P 0.25        P 0.50        P 0.75        P 1.00        P mean1        P mean2        P norm
        R norm
        ==================\t
\t========      ========      ========      ========      =========      =========      ========
          ========

    EndOfList
        ;

        &calc_precision_recall_test;

        print "\n\n\n\n\n\n\n\n";
        return;


    }

    ########################################################
    ##  CALC_FULL_PRECISION_RECALL_TEST
    ##
    ##  sub function of precision_recall_test
    ##  This function is actually doing the algorithm of the
    ##  precision_recall_test function
    ########################################################
    sub calc_precision_recall_test {
        # Temp variables to store the sum of the precision/recall
        my $recall_norm_sum = 0;
        my $prec_norm_sum = 0;
        my $prec_mean1_sum = 0;
        my $prec_mean2_sum = 0;
        my $prec_025_sum = 0;
        my $prec_050_sum = 0;
        my $prec_075_sum = 0;
        my $prec_100_sum = 0;

        # Temp variables to store the avg of the precision/recall
        my $recall_norm_avg = 0;
        my $prec_norm_avg = 0;
        my $prec_mean1_avg = 0;
        my $prec_mean2_avg = 0;
        my $prec_025_avg = 0;
        my $prec_050_avg = 0;
        my $prec_075_avg = 0;
        my $prec_100_avg = 0;

        # default case
        for my $ind (1 .. $total_qrys) {
```

```
1953            &get_retrieved_set( $qry_vector[$ind] );
1954            # print "$ind\n";
1955            &comp_recall( $ind );
1956            $recall_norm_sum += $recall_norm;
1957            $prec_norm_sum += $prec_norm;
1958            $prec_mean1_sum += $prec_mean1;
1959            $prec_mean2_sum += $prec_mean2;
1960            $prec_025_sum += &get_prec(0.25);
1961            $prec_050_sum += &get_prec(0.50);
1962            $prec_075_sum += &get_prec(0.75);
1963            $prec_100_sum += &get_prec(1.00);
1964        }
1965
1966        # Temp variables to store the avg of the precision/recall
1967        $recall_norm_avg = $recall_norm_sum / $total_qrys;
1968        $prec_norm_avg = $prec_norm_sum / $total_qrys;
1969        $prec_mean1_avg = $prec_mean1_sum / $total_qrys;
1970        $prec_mean2_avg = $prec_mean2_sum / $total_qrys;
1971        $prec_025_avg = $prec_025_sum / $total_qrys;
1972        $prec_050_avg = $prec_050_sum / $total_qrys;
1973        $prec_075_avg = $prec_075_sum / $total_qrys;
1974        $prec_100_avg = $prec_100_sum / $total_qrys;
1975
1976        # Format vars
1977        my $recall_norm_avg_fat = sprintf "%0.3f", $recall_norm_avg;
1978        my $prec_norm_avg_fat = sprintf "%0.3f", $prec_norm_avg;
1979        my $prec_mean1_avg_fat = sprintf "%0.3f", $prec_mean1_avg;
1980        my $prec_mean2_avg_fat = sprintf "%0.3f", $prec_mean2_avg;
1981        my $prec_025_avg_fat = sprintf "%0.3f", $prec_025_avg;
1982        my $prec_050_avg_fat = sprintf "%0.3f", $prec_050_avg;
1983        my $prec_075_avg_fat = sprintf "%0.3f", $prec_075_avg;
1984        my $prec_100_avg_fat = sprintf "%0.3f", $prec_100_avg;
1985
1986        print "    $method\t\t\t  $prec_025_avg_fat        $prec_050_avg_fat          $prec_075_avg_fat         $prec_100_avg_fa
      t         $prec_mean1_avg_fat          $prec_mean2_avg_fat          $prec_norm_avg_fat         $recall_norm_avg_fat";
1987        print "\n";
1988    }
1989
1990
1991    ######################################################
1992    ## COSINE_SIM_A
1993    ##
1994    ## Computes the cosine similarity for two vectors
1995    ## represented as associate arrays.
1996    ######################################################
1997
1998    sub cosine_sim_a {
1999
2000        my $vec1 = shift;
2001        my $vec2 = shift;
2002
2003        my $num     = 0;
2004        my $sum_sq1 = 0;
2005        my $sum_sq2 = 0;
2006
2007        my @val1 = values %{ $vec1 };
2008        my @val2 = values %{ $vec2 };
2009
2010        # determine shortest length vector. This should speed
2011        # things up if one vector is considerable longer than
2012        # the other (i.e. query vector to document vector).
2013
2014        if ((scalar @val1) > (scalar @val2)) {
2015            my $tmp  = $vec1;
2016               $vec1 = $vec2;
2017               $vec2 = $tmp;
2018        }
2019
2020        # calculate the cross product
2021
2022        my $key = undef;
2023        my $val = undef;
2024
2025        while (($key, $val) = each %{ $vec1 }) {
2026            $num += $val * ($$vec2{ $key } || 0);
2027        }
2028
2029        # calculate the sum of squares
2030
2031        my $term = undef;
2032
2033        foreach $term (@val1) { $sum_sq1 += $term * $term; }
2034        foreach $term (@val2) { $sum_sq2 += $term * $term; }
2035
2036        # Handle the special case when interactive query return 0 results ...
2037        if(($sum_sq1 * $sum_sq2) == 0 and $num == 0) {
2038            return 0;
2039        }
2040        if(($sum_sq1 * $sum_sq2) == 0 and $num != 0) {
```

```
2041            return 1;
2042        }
2043        return ( $num / sqrt( $sum_sq1 * $sum_sq2 ));
2044    }
2045
2046
2047    ######################################################
2048    ##  COSINE_SIM_B
2049    ##
2050    ##  This function assumes that the sum of the squares
2051    ##  of the term weights have been stored in advance for
2052    ##  each document and are passed as arguments.
2053    ######################################################
2054
2055    sub cosine_sim_b {
2056
2057        my $vec1 = shift;
2058        my $vec2 = shift;
2059
2060        my $sum_sq1 = shift;
2061        my $sum_sq2 = shift;
2062
2063        my $num     = 0;
2064        my $key     = undef;
2065        my $val     = undef;
2066
2067        while (($key, $val) = each %{ $vec1 }) {
2068            $num += $val * $$vec2{ $key };
2069        }
2070
2071        # Handle the special case when interactive query return 0 results ...
2072        if(($sum_sq1 * $sum_sq2) == 0 and $num == 0) {
2073                return 0;
2074        }
2075        if(($sum_sq1 * $sum_sq2) == 0 and $num != 0) {
2076            return 1;
2077        }
2078        return ( $num / sqrt( $sum_sq1 * $sum_sq2 ));
2079    }
2080
2081
2082    ######################################################
2083    ##  DICE_SIM
2084    ##
2085    ##  This function calculate Dice
2086    ######################################################
2087
2088    sub dice_sim {
2089
2090        my $vec1 = shift;
2091        my $vec2 = shift;
2092
2093        my $num     = 0;
2094        my $sum_sq1 = 0;
2095        my $sum_sq2 = 0;
2096
2097        my @val1 = values %{ $vec1 };
2098        my @val2 = values %{ $vec2 };
2099
2100        # determine shortest length vector. This should speed
2101        # things up if one vector is considerable longer than
2102        # the other (i.e. query vector to document vector).
2103
2104        if ((scalar @val1) > (scalar @val2)) {
2105            my $tmp  = $vec1;
2106               $vec1 = $vec2;
2107               $vec2 = $tmp;
2108        }
2109
2110        # calculate the cross product
2111
2112        my $key = undef;
2113        my $val = undef;
2114
2115        while (($key, $val) = each %{ $vec1 }) {
2116            $num += $val * ($$vec2{ $key } || 0);
2117        }
2118
2119        $num *= 2;
2120
2121        # calculate the sum of squares
2122
2123        my $term = undef;
2124
2125        foreach $term (@val1) { $sum_sq1 += $term; }
2126        foreach $term (@val2) { $sum_sq2 += $term; }
2127
2128        # Handle the special case when interactive query return 0 results ...
2129        if(($sum_sq1 + $sum_sq2) == 0 and $num == 0) {
```

```
2130            return 0;
2131        }
2132        if(($sum_sq1 + $sum_sq2) == 0 and $num != 0) {
2133            return 1;
2134        }
2135
2136        return ( $num / ($sum_sq1 + $sum_sq2));
2137    }
2138
2139    ######################################################
2140    ##  JACCARD_SIM
2141    ##
2142    ##  This function calculate Jaccard
2143    ######################################################
2144
2145    sub jaccard_sim {
2146
2147        my $vec1 = shift;
2148        my $vec2 = shift;
2149
2150        my $num     = 0;
2151        my $sum_sq1 = 0;
2152        my $sum_sq2 = 0;
2153
2154        my @val1 = values %{ $vec1 };
2155        my @val2 = values %{ $vec2 };
2156
2157        # determine shortest length vector. This should speed
2158        # things up if one vector is considerable longer than
2159        # the other (i.e. query vector to document vector).
2160
2161        if ((scalar @val1) > (scalar @val2)) {
2162            my $tmp  = $vec1;
2163               $vec1 = $vec2;
2164               $vec2 = $tmp;
2165        }
2166
2167        # calculate the cross product
2168
2169        my $key = undef;
2170        my $val = undef;
2171
2172        while (($key, $val) = each %{ $vec1 }) {
2173            $num += $val * ($$vec2{ $key } || 0);
2174        }
2175
2176        # calculate the sum of squares
2177
2178        my $term = undef;
2179
2180        foreach $term (@val1) { $sum_sq1 += $term; }
2181        foreach $term (@val2) { $sum_sq2 += $term; }
2182        # print "$num, $sum_sq1, $sum_sq2\n";
2183        # Handle the special case when interactive query return 0 results ...
2184        if(($sum_sq1 + $sum_sq2 - $num) == 0 and $num == 0) {
2185            return 0;
2186        }
2187        if(($sum_sq1 + $sum_sq2 - $num) == 0 and $num != 0) {
2188            return 1;
2189        }
2190        return ( $num / ($sum_sq1 + $sum_sq2 - $num));
2191    }
2192
2193    ######################################################
2194    ##  OVERLAP_SIM
2195    ##
2196    ##  This function calculate Overlap
2197    ######################################################
2198
2199    sub overlap_sim {
2200
2201        my $vec1 = shift;
2202        my $vec2 = shift;
2203
2204        my $num     = 0;
2205        my $sum_sq1 = 0;
2206        my $sum_sq2 = 0;
2207
2208        my @val1 = values %{ $vec1 };
2209        my @val2 = values %{ $vec2 };
2210
2211        # determine shortest length vector. This should speed
2212        # things up if one vector is considerable longer than
2213        # the other (i.e. query vector to document vector).
2214
2215        if ((scalar @val1) > (scalar @val2)) {
2216            my $tmp  = $vec1;
2217               $vec1 = $vec2;
2218               $vec2 = $tmp;
```

```
2219        }
2220
2221        # calculate the cross product
2222
2223        my $key = undef;
2224        my $val = undef;
2225
2226        while (($key, $val) = each %{ $vec1 }) {
2227            $num += $val * ($$vec2{ $key } || 0);
2228        }
2229
2230        # calculate the sum of squares
2231
2232        my $term = undef;
2233
2234        foreach $term (@val1) { $sum_sq1 += $term; }
2235        foreach $term (@val2) { $sum_sq2 += $term; }
2236
2237        # Handle the special case when interactive query return 0 results ...
2238        if( (($sum_sq1 < $sum_sq2) ? $sum_sq1 : $sum_sq2) == 0 and $num == 0 ) {
2239            return 0;
2240        }
2241        if( (($sum_sq1 < $sum_sq2) ? $sum_sq1 : $sum_sq2) == 0 and $num == 1 ) {
2242            return 1;
2243        }
2244
2245        return ( $num / (($sum_sq1 < $sum_sq2) ? $sum_sq1 : $sum_sq2));
2246    }
2247
2248
2249    ######################################################
2250    ##  LOG10
2251    ##
2252    ##  Calculate the value of log10
2253    ######################################################
2254    sub log10 {
2255        my $n = shift;
2256        return log($n)/log(10);
2257    }
2258
2259
2260    ######################################################
2261    ##  FAC
2262    ##
2263    ##  Calculate the n!
2264    ######################################################
2265    sub fac {
2266        my $n = shift;
2267        if(1 == $n) {
2268            return 1;
2269        }
2270        elsif(0 == $n) {
2271            return 0;
2272        }
2273        else {
2274            return ($n * fac($n- 1));
2275        }
2276    }
2277
2278
2279    ######################################################
2280    ##  GET_PREC
2281    ##  @level the level of the prec: e.x. 0.25, 0.50
2282    ##
2283    ##  Calculate the precision given the level
2284    ######################################################
2285    sub get_prec {
2286        my $level = shift;
2287        my $low = undef;
2288        my $high = undef;
2289
2290        my @key_rec_prec = sort keys %rec_prec;
2291        my $i = 0;
2292
2293        # special case when there is only one document
2294        if($#key_rec_prec == 0) {
2295            return $rec_prec{$key_rec_prec[0]};
2296        }
2297
2298        for($i=0;$i<=$#key_rec_prec;$i++) {
2299            if($key_rec_prec[$i] == $level) {
2300                return $rec_prec{$level};
2301            }
2302            # do the Linear interpolation between these two bound points
2303            if($key_rec_prec[$i] > $level) {
2304                if($i == 0) {
2305                    return ( $rec_prec{$key_rec_prec[0]} + ((($level - $key_rec_prec[0]) / ($key_rec_prec[1] - $key_rec_prec[0])
    ) * ($rec_prec{$key_rec_prec[1]} - $rec_prec{$key_rec_prec[0]})) );
2306                }
```

```
2307                else {
2308                    return ( $rec_prec{$key_rec_prec[$i - 1]} + (($rec_prec{$key_rec_prec[$i]} - $rec_prec{$key_rec_prec[$i - 1]
    }) * (($level - $key_rec_prec[$i - 1]) / ($key_rec_prec[$i] - $key_rec_prec[$i - 1]))) );
2309                }
2310            }
2311        }
2312
2313
2314    }
2315
2316    ######################################################
2317    ##  CLEANUP
2318    ##
2319    ##  This function is just used to re-init the variables
2320    ######################################################
2321
2322    sub cleanup {
2323        # clean up and reset all the variables
2324        @doc_vector = ( );
2325        @qry_vector = ( );
2326        %docs_freq_hash = ( );
2327        %corp_freq_hash = ( );
2328        %stoplist_hash  = ( );
2329        @titles_vector  = ( );
2330        %relevance_hash = ( );
2331        @doc_simula = ( );
2332        @res_vector = ( );
2333        $prec_mean1 = undef;
2334        $prec_mean2 = undef;
2335        $recall_norm = undef;
2336        $prec_norm = undef;
2337        %rank_docn = ();
2338        %rec_prec = ();
2339        $method = undef;
2340        $total_docs = undef;
2341        $total_qrys = undef;
2342        $token_docs = "$DIR/cacm";           # tokenized cacm journals
2343        $corps_freq = "$DIR/cacm";           # frequency of each token in the journ.
2344        $stoplist   = "$DIR/common_words";   # common uninteresting words
2345        $titles     = "$DIR/titles.short";   # titles of each article in cacm
2346        $token_qrys = "$DIR/query";          # tokenized canned querys
2347        $query_freq = "$DIR/query";          # frequency of each token in the querys
2348        $query_relv = "$DIR/query\.rels";    # relevance of a journal entry to a
2349        # these files are created in your $HOME directory
2350
2351        $token_intr = "$HOME/interactive";    # file created for interactive queries
2352        $inter_freq = "$HOME/interactive";    # frequency of each token in above
2353                                              #  given query
2354        $TITLE_BASE_WEIGHT = 3;     # weight given a title token
2355        $KEYWD_BASE_WEIGHT = 4;     # weight given a key word token
2356        $AUTHR_BASE_WEIGHT = 3;     # weight given an an author token
2357        $ABSTR_BASE_WEIGHT = 1;     # weight given an abstract word token
2358
2359        $sim = "cosine_sim_a";      # reset the $sim to cosine_sim_a
2360
2361    }
2362
2363    ######################################################
2364    ##  PRINT_FULL_TABLE
2365    ##
2366    ##  Print out the full precision/recall table by chaning
2367    ##  one parameter at a time
2368    ######################################################
2369
2370    sub print_full_table {
2371        print << "EndOfList";
2372
2373                                        ************************************************************
2374                                        **      Precision / Recall averaged over 33 queries       **
2375                                        ************************************************************
2376
2377        Permutation Name   \t\t P 0.25        P 0.50        P 0.75        P 1.00        P mean1        P mean2        P norm
        R norm
2378        ====================\t
\t========      ========      ========      ========      =========      =========      ========
          ========
2379
2380    EndOfList
2381        ;
2382
2383        # RAW TF weighting
2384
2385        $method = "RAW TF";
2386
2387        # initializarion
2388        &init_files ( "stemmed" );
2389        &init_corp_freq;
2390
2391        # init docs vector
```

```
2392        my $token_docs_fh = new FileHandle $token_docs, "r"
2393        or croak "Failed $token_docs";
2394
2395        my $word    = undef;
2396
2397        my $doc_num =  0;    # current document number and total docs at end
2398        my $tweight =  0;    # current weight assigned to document token
2399
2400        push @doc_vector, { };     # push one empty value onto @doc_vector so that
2401                                   # indices correspond with document numbers
2402
2403        while (defined( $word = <$token_docs_fh> )) {
2404
2405            chomp $word;
2406
2407            last if $word =~ /^\.I 0/; # indicates end of file so kick out
2408
2409            if ($word =~ /^\.I/) {     # indicates start of a new document
2410
2411                push @doc_vector, { };
2412                $doc_num++;
2413
2414                next;
2415            }
2416
2417            $tweight = $TITLE_BASE_WEIGHT and next if $word =~ /^\.T/;
2418            $tweight = $KEYWD_BASE_WEIGHT and next if $word =~ /^\.K/;
2419            $tweight = $ABSTR_BASE_WEIGHT and next if $word =~ /^\.W/;
2420            $tweight = $AUTHR_BASE_WEIGHT and next if $word =~ /^\.A/;
2421
2422            if ($word =~ /[a-zA-Z]/ and ! exists $stoplist_hash{ $word }) {
2423
2424                if (defined( $docs_freq_hash{ $word } )) {
2425
2426                    $doc_vector[$doc_num]{ $word } += $tweight;
2427                }
2428                else {
2429                    print "ERROR: Document frequency of zero: ", $word, "\n";
2430                }
2431            }
2432        }
2433
2434        $total_docs = $doc_num;
2435
2436        # init query vector
2437        my $QUERY_BASE_WEIGHT = 2;
2438        my $QUERY_AUTH_WEIGHT = 2;
2439
2440        my $token_qrys_fh = new FileHandle $token_qrys, "r"
2441            or croak "Failed $token_qrys";
2442
2443        $word = undef;
2444
2445        $tweight =  0;
2446        my $qry_num =  0;
2447
2448        push @qry_vector, { };    # push one empty value onto @qry_vectors so that
2449                                  # indices correspond with query numbers
2450
2451        while (defined( $word = <$token_qrys_fh> )) {
2452
2453            chomp $word;
2454
2455            if ($word =~ /^\.I/) {
2456
2457                push @qry_vector, { };
2458                $qry_num++;
2459
2460                next;
2461            }
2462
2463            $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
2464            $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;
2465
2466            if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {
2467
2468                if (! exists $docs_freq_hash{ $word }) {
2469                    print "ERROR: Document frequency of zero: ", $word, "\n";
2470                }
2471                else {
2472                    $qry_vector[$qry_num]{ $word } += $tweight;
2473                }
2474            }
2475        }
2476
2477        $total_qrys = $qry_num;
2478
2479        &calc_precision_recall_test;
2480        &cleanup;
```

```
2481
2482
2483        # Boolean weighting
2484
2485        $method = "Boolean";
2486
2487        # initializarion
2488        &init_files ( "stemmed" );
2489        &init_corp_freq;
2490
2491        # init docs vector
2492        $token_docs_fh = new FileHandle $token_docs, "r"
2493        or croak "Failed $token_docs";
2494
2495        $word    = undef;
2496
2497        $doc_num =  0;    # current document number and total docs at end
2498        $tweight =  0;    # current weight assigned to document token
2499
2500        push @doc_vector, { };     # push one empty value onto @doc_vector so that
2501                                   # indices correspond with document numbers
2502
2503        while (defined( $word = <$token_docs_fh> )) {
2504
2505            chomp $word;
2506
2507            last if $word =~ /^\.I 0/; # indicates end of file so kick out
2508
2509            if ($word =~ /^\.I/) {     # indicates start of a new document
2510
2511                push @doc_vector, { };
2512                $doc_num++;
2513
2514                next;
2515            }
2516
2517            $tweight = $TITLE_BASE_WEIGHT and next if $word =~ /^\.T/;
2518            $tweight = $KEYWD_BASE_WEIGHT and next if $word =~ /^\.K/;
2519            $tweight = $ABSTR_BASE_WEIGHT and next if $word =~ /^\.W/;
2520            $tweight = $AUTHR_BASE_WEIGHT and next if $word =~ /^\.A/;
2521
2522            if ($word =~ /[a-zA-Z]/ and ! exists $stoplist_hash{ $word }) {
2523
2524                if (defined( $docs_freq_hash{ $word } )) {
2525
2526                    $doc_vector[$doc_num]{ $word } = 1;
2527                }
2528                else {
2529                    $doc_vector[$doc_num]{ $word } = 0;
2530                    # print "ERROR: Document frequency of zero: ", $word, "\n";
2531                }
2532            }
2533        }
2534
2535        $total_docs = $doc_num;
2536
2537        # init query vector
2538        $QUERY_BASE_WEIGHT = 2;
2539        $QUERY_AUTH_WEIGHT = 2;
2540
2541        $token_qrys_fh = new FileHandle $token_qrys, "r"
2542            or croak "Failed $token_qrys";
2543
2544        $word = undef;
2545
2546        $tweight =  0;
2547        $qry_num =  0;
2548
2549        push @qry_vector, { };    # push one empty value onto @qry_vectors so that
2550                                  # indices correspond with query numbers
2551
2552        while (defined( $word = <$token_qrys_fh> )) {
2553
2554            chomp $word;
2555
2556            if ($word =~ /^\.I/) {
2557
2558                push @qry_vector, { };
2559                $qry_num++;
2560
2561                next;
2562            }
2563
2564            $tweight = $QUERY_BASE_WEIGHT and next if $word =~ /^\.W/;
2565            $tweight = $QUERY_AUTH_WEIGHT and next if $word =~ /^\.A/;
2566
2567            if ($word =~ /[a-zA-Z]/ && ! exists $stoplist_hash{ $word }) {
2568
2569                if (! exists $docs_freq_hash{ $word }) {
```

```
2570                    $qry_vector[$qry_num]{ $word } = 0;
2571                }
2572                else {
2573                    $qry_vector[$qry_num]{ $word } = 1;
2574                }
2575            }
2576        }
2577        $total_qrys = $qry_num;
2578        &calc_precision_recall_test;
2579        &cleanup;
2580
2581        # use Dice similarity
2582        $method = "Dice";
2583        $sim = "dice";
2584        &init_files ( "stemmed" );
2585        &init_corp_freq;
2586        $total_docs = &init_doc_vectors;
2587        $total_qrys = &init_qry_vectors;
2588        &calc_precision_recall_test;
2589        &cleanup;
2590
2591        # use Jaccard similarity
2592        $method = "Jaccard";
2593        $sim = "jaccard";
2594        &init_files ( "stemmed" );
2595        &init_corp_freq;
2596        $total_docs = &init_doc_vectors;
2597        $total_qrys = &init_qry_vectors;
2598        &calc_precision_recall_test;
2599        &cleanup;
2600
2601        # use Overlap similarity
2602        $method = "Overlap";
2603        $sim = "overlap";
2604        &init_files ( "stemmed" );
2605        &init_corp_freq;
2606        $total_docs = &init_doc_vectors;
2607        $total_qrys = &init_qry_vectors;
2608        &calc_precision_recall_test;
2609        &cleanup;
2610
2611        # use raw, unstemmed tokens (all converted to lower case)
2612        $method = "Unstem";
2613        # init_files using unstemmed
2614        &init_files ( "unstemmed" );
2615        &init_corp_freq;
2616        $total_docs = &init_doc_vectors;
2617        $total_qrys = &init_qry_vectors;
2618        &calc_precision_recall_test;
2619        &cleanup;
2620
2621        # Include all tokens, including punctuation
2622        $method = "No stwd";
2623        # initializarion
2624        &init_files ( "stemmed" );
2625        &init_corp_freq;
2626        %stoplist_hash = ();
2627        $total_docs = &init_doc_vectors;
2628        $total_qrys = &init_qry_vectors;
2629        &calc_precision_recall_test;
2630        &cleanup;
2631
2632        # Weight titles, keywords, author list and abstract words equally
2633        $method = "Region 1111";
2634        # reset global vars for the weight
2635        $TITLE_BASE_WEIGHT = 1;     # weight given a title token
2636        $KEYWD_BASE_WEIGHT = 1;     # weight given a key word token
2637        $AUTHR_BASE_WEIGHT = 1;     # weight given an an author token
2638        $ABSTR_BASE_WEIGHT = 1;     # weight given an abstract word token
2639        # initializarion
2640        &init_files ( "stemmed" );
2641        &init_corp_freq;
2642        $total_docs = &init_doc_vectors;
2643        $total_qrys = &init_qry_vectors;
2644        &calc_precision_recall_test;
2645        &cleanup;
2646
2647        # Weight titles, keywords, author list and abstract words equally
2648        $method = "Region 1114";
2649        # reset global vars for the weight
2650        $TITLE_BASE_WEIGHT = 1;     # weight given a title token
2651        $KEYWD_BASE_WEIGHT = 1;     # weight given a key word token
2652        $AUTHR_BASE_WEIGHT = 1;     # weight given an an author token
2653        $ABSTR_BASE_WEIGHT = 4;     # weight given an abstract word token
2654        # initializarion
2655        &init_files ( "stemmed" );
2656        &init_corp_freq;
2657        $total_docs = &init_doc_vectors;
2658        $total_qrys = &init_qry_vectors;
```

```
2659        &calc_precision_recall_test;
2660        &cleanup;
2661
2662
2663        # Default parameters
2664        $method = "Default";
2665        # initializarion
2666        &init_files ( "stemmed" );
2667        &init_corp_freq;
2668
2669        $total_docs = &init_doc_vectors;
2670        $total_qrys = &init_qry_vectors;
2671
2672        &calc_precision_recall_test;
2673        print "\n\n\n\n\n\n\n";
2674    }
```