

# Comparison of Systems Using Pairs-Out-of-Order

Paul B. Kantor and Kwong Bor Ng,  
Rutgers University  
*{kantor, kbng}@scils*

David Hull, Rank Xerox France

November 18, 1997

## 1 Introduction

Our study of the data to be extracted from the TREC results has two tracks. One track asks whether we can draw statistically significant conclusions about the effectiveness of the several systems, from datasets of the size generated by the TREC conferences. This is a subtle problem in multiple comparisons, lacking one of the most useful kinds of data for such studies: information about the “intrinsic variability” or “error term” for specific system-topic combinations. As might be expected, that ability to draw significant conclusions goes up with the amount of natural difference among the systems, but goes down as we increase the number of systems, and as we decrease the amount of inter-system difference that we wish to be able to detect with confidence. Detailed results, based upon the TREC4 data are presented in a companion paper and report.

In addition to the results of evaluation, which is based on comparison of the ranked retrieval lists with relevance judgments, we can also analyze the ranked lists themselves. The lists produced by a specific system, in response to each of the several assigned topics, tell us something about how the system works. In particular, if two systems produce lists which are very similar, we may say that they work in similar ways and hence are “similar systems”. In the present report we consider a specific rigorously defined measure of the similarity between two ranked lists of 1000 documents, with special attention to the fact that the lists will overlap in part, but not completely.

## 2 Display of System Dissimilarity

Once the differences between systems have been defined and calculated, we move to the issue of how to summarize the results of the calculations. The most rigorous summary is provided by a complete tabulation of all the “inter-system dissimilarity measures”. These are given in Tables 1 and 2. Given that data, it is interesting to search for ways of representing that data about similarities which improve human comprehension of the relations revealed. We have explored several ways of addressing this.

### 2.1 Clustering

One approach is clustering. Clustering is the name for a large class of techniques for assembling diverse objects into related groups. Clustering may be unconstrained, or hierarchical. All techniques currently used in statistical packages, and known to the IR community, are hierarchical. This means that clusters are built up by bringing together the most similar objects, and thereafter bringing together the most similar clusters, to form larger clusters. The result is a tree of clusters. Because the tree can look also like the roots of tree, when the single large cluster is placed at the top, the resulting diagram is called a dendrogram. We have developed scripts which form the dendrogram based on a specific measure of inter-cluster similarity, the complete linkage measure. In the complete linkage measure, the similarity of two clusters to be joined is measured by the largest dissimilarity between elements drawn from the union. In other words, clusters are built to minimize radius.

In addition, it is possible to make a rigorous investigation of all possible clusterings. In this approach every possible clustering is scored in two ways: maximum cluster diameter  $d$ , and minimum inter-cluster separation  $s$ . The ideal clustering would have a large  $s$  and small  $d$ . If a clustering  $\mathcal{C}$  has smaller  $d$  and larger  $s$  than clustering  $\mathcal{C}'$ , the cluster  $\mathcal{C}$  is said to “dominate” clustering  $\mathcal{C}'$ . Of course these two goals compete with each other, and no clustering is optimum with regard to both. However, some clusterings are “Pareto-optimal” which means that they are not dominated by any other clustering. We have been fortunate to obtain the cooperation of Prof. Pierre Hansen ([1, 2]) of the University of Montreal, who has developed a program for rigorously investigating this issue, and finding all of the Pareto-optimal clusterings. He has run this program against all of the inter-system dissimilarity data that we have calculated for both the adhoc and routing runs in

TREC4. The results are generally interpreted by looking for points which stand out from a general trend, which can be thought of as the “Pareto frontier”. For these particular data, the results of this more powerful analysis did not provide any useful insight.

## 2.2 Multi-Dimensional Scaling

We have also considered a technique called multi-dimensional scaling, which asks whether the systems can be represented by points in a Euclidean space of dimension higher than 2, in such a way that at least the rank ordering of the inter-system dissimilarities is monotonically mapped into Euclidean distance. The effectiveness of such an approach is tested by a measure called the “stress” of the mapping. Again, results were disappointing, in that stress remained high even up to the 4th dimension.

Thus we can not rigorously assert that we have either found clusters of similar systems, or at least, we can not assert that such clusters can be sensibly displayed in a few dimensions.

## 2.3 An Interactive Display

Nonetheless, we believe that the human mind can perceive and explore patterns even when they do not have completely rigorous statistical justification. To this end, we have developed a Java applet which displays systems in the two dimensions of the screen. This applet is run separately for the adhoc and the routing systems. The applet is an adaptation of the standard GraphLayout applet available in the Java Developers Kit jdk1.1.3(4). This places one system at a fixed point on the screen, and initially places the others randomly.

### 2.3.1 Dynamic Rules of the Display

These other points then move towards equilibrium under the combined influence of three forces: a Hooke’s law force moving them towards the equilibrium configuration ( $F = ks$ , where  $s$  is the displacement of the point from equilibrium); a logarithmic potential modification, which diminishes the “force” due to remote points; and an effective viscosity, which must be set quite high to keep the points from bouncing about as they are recomputed endlessly.

Of course the system cannot achieve equilibrium, in only two dimensions, and if the points are “reshaken”, they will settle into new configurations. To

make the display more useful we have modified it so that all distances from the single fixed node are “true”. In other words, if  $X$  appears closer to the fixed node ( $F$ ) than does  $Y$ , system  $X$  is really more similar to system  $F$  than is system  $Y$ .

### 2.3.2 Interactive Features of the Display

By “right-clicking” on any system  $S$  we can change the display so that  $S$  is the fixed system, and then drag it to the center of the display.

We have also provided a zoom capability which changes the scale of the display. We provide the option of changing the labels from system id-code to the dissimilarity of the system, from the system represented by the fixed node (in arbitrary units).

We hope that this interactive tool will facilitate interactive investigation of the relations among systems which are surely to be found in comparison of the ranked lists of documents which they produce in response to the TREC topics.

In the remainder of this report we summarize the computation of the inter-system dissimilarities, and the results for the TREC4 systems and topics.

## 3 Distance between ranked outputs of IR systems

Mathematically, a ranked list of  $N$  items can be broken down into  $\frac{N \times (N-1)}{2}$  ranked pairs. From such a set of ranked pairs, only one ranked list can be reconstructed. For example, for a ranked list with 4 items  $A, B, C, D$  such that  $A > B > C > D$ , (we use “ $>$ ” to represent position in the ranked list) the list generates  $\frac{4 \times (4-1)}{2} = 6$  ranked pairs:  $A > B$ ,  $A > C$ ,  $B > C$ ,  $B > D$ ,  $C > D$ , and, from these 6 ranked pairs, we can re-construct exactly one ranked list which contains all the 4 items, i.e., the original ranked list,  $A > B > C > D$ . In other words, the ranked list and the set of ranked pairs are alternate ways of representing the ordering among the elements (Of course, there could be inconsistent pair ranks, such as  $A > B$ ,  $B > C$  and  $C > A$ . Those do not occur in our situation.) Therefore, when comparing two ranked lists, instead of comparing two ranked lists directly, we can compare the set of ranked pairs determined by each list.

If we have two ranked lists which contain the same elements, we can break down these two lists into 2 set of ranked pairs. Each set should contain the

same number of pairs and every pair appearing in one set will also be a pair in the other set (but may be in different order.) For example, for two ranked lists:  $A > B > C$  and  $B > A > C$ , each generates 3 ranked pairs. For the first list, these are  $A > B$ ,  $A > C$ , and  $B > C$ . For the second list, they are  $B > A$ ,  $A > C$ , and  $B > C$ . Every pair in the first set is also a pair in the second. The first of these pairs is out of order with respect to each other, while the second and third pairs are in the same order.

This representation of ranked lists can be used to calculate a distance between two ranked lists (Kemeny [3]). If two ranked lists have the same elements but the elements are arranged in a different order, we can represent each list in terms of ranked pairs and count the number of out-of-order pairs between them. For example, for the lists “ $A > B > C$ ” and “ $B > A > C$ ”, there is 1 out-of-order pair (i.e., in the first list,  $A > B$ , while in the second list,  $B > A$ ). We call this count the unnormalized distance between the two lists. (In this case, 1).

In an IR ranked output environment, different ranking algorithms ranking all the documents in a collection with respect to a query produce two ranked lists containing same elements arranged in different order. In this case we can use the number of out-of-order pairs to measure the distance, or dis-similarity, between the ranked outputs of two IR systems.

In TREC, participants report only the top 1000 items of their output lists. Therefore, we must be able to compare lists when the elements in the two lists are not exactly the same.

When the two ranked lists have different elements, we may encounter two new situations: (1) For a given pair in one list, only one element is present in the other list; (2) For a given pair in one list, neither element is in the other list.

The first situation can be addressed directly. The missing element must be in the lower part of the second list, below the cutting point, therefore we know the order of the two elements in the second list. And, of course, we can observe their order in the first list.

The second situation is different because both missing elements must be in the lower part of the second list and we don't know their relative position. We argue that they can be either out-of-order (score 1) or not (score 0), with equal probability (i.e. there are as many permutations of the list in which the pair is in order as there are with it out of order). Therefore we treat the out-of-order scores for those pairs as “0.5”.

## 4 Computation of the Dissimilarities

We represent the out-of-order scores generally by the letter  $z$ . We can use  $z$ -scores as a measure for inter-system dissimilarity. Any monotonically decreasing function of the  $z$ -scores can be used as a measure for inter-system similarity.

### 4.1 Dendrogrammer

A dendrogram is a way of displaying every step of a hierarchical clustering process. Our dendrogrammer package takes TREC-format result files as input and produces a dendrogram according to either the distance (i.e., averaged and normalized  $z$ -score), or a particular measure of similarity ( $\frac{1}{1+distance}$ ), among the systems. We also allow the user to code alternate representations of proximity based on the dissimilarity.

The clustering is done by the complete linkage method (i.e., the distance between two clusters is calculated as the distance between their most dissimilar elements).

#### 4.1.1 Input File Structure

The input files are lists of document numbers, ranked according to some document retrieval algorithms, with respect to a topic. The record format of the input files is as follow:

```
xxx Q docno rank score system
```

where “xxx” is the topic number, Q is the type of method employed, docno is the document number, rank is the rank of that document (beginning at 0, corresponding to the highest relevance score), score is the relevance score assigned by the document retrieval algorithm, and system is the name of the corresponding information retrieval system.

The records are free formatted, with fields separated by white space.

#### 4.1.2 File Unpacking

The dendrogrammer is a tar file, you have to unpack it by the Unix command:

```
tar xf trec.dendrogrammer.tar
```

After unpacking, you should find a new directory named **trec.dendrogrammer** with the following files inside. These files must be present in the current directory when running the dendrogrammer:

**alphabet.txt** For translating system names to alphabet (single characters)

**get.combination.awk** To generate a list of system-pairs

**get.all.z.04.sh** To calculate  $z$ -scores (inter-systems out-of-order scores) for system-pairs. This calls `nist96.07.sh`

**nist96.07.sh** To calculate the  $z$  score for one system-pair

**nist96.02.z3** To calculate  $z_3$ , one component of  $z$ , defined below.

**trans.sys.to.alpha.awk** To translate system names to the one-character alphabet

**dendrogrammer.awk** To construct the dendrogram

**make.matrix.awk** To construct the square matrix of inter-system scores (system labels appear on the top and in the first column)

**renice.matrix.awk** To refine matrix to a single triangle matrix format (suitable for use in SPSS)

**complete.clusterer.awk** To construct clusters using the complete linkage method.

**get.inter.system.distance.awk** To construct a list of the average (over topics) of normalized distance ( $d = Average(z)$ ) or similarity  $\frac{1}{1+d}$  for each system-pair.

## 4.2 Package Use

The user must specify 4 parameters when invoking the program. They are (1) proximity measure (e.g., similarity, distance, or other by measure, by setting the flag `-t`);

(2) input directory (by flag `-i`);

(3) output directory (by flag `-o`); and

(4) output file names extension (by flag `-e`).

All files in the input directory with file names beginning with “input” will be used as input files (e.g. “input.something”. The files in the results directory of NIST’s ftp site use the same naming convention)

## 5 Output Files

The output directory is specified by flag “-o”. There are 6 output files, they are:

**sys.alpha.dictionary.“extension”** Dictionary of system name vs. alphabet; extension is user-specified by flag -e

**z.table.“extension”** Record format:

```
sys1.topicno sys2.topicno N1 N2 m z1 z2 z3 z4 z5 z.
```

In this format, *sys1* and *sys2* are the names of the two information retrieval systems; *topicno* is corresponding topic number; *N1* and *N2* are sizes of the input lists (i.e., number of documents retrieved) for *sys1* and *sys2* respectively; *m* is the number of common documents retrieved by both systems.

The next five numbers are raw pair-out-of-order counts of several kinds. (Details are given in the following section).

*z1* is the out-of-order scores for pairs with one element in the intersection of the two lists, and one element unique to the list generated by *sys1*.

*z2* is the out-of-order scores for pairs with one element in the intersection of the two lists, and one element unique to the list generated by *sys2*.

*z3* is the out-of-order scores for pairs with both elements in the intersection.

*z4* is the out-of-order scores for pairs with one element unique to *sys1* and the other unique to *sys2*.

*z5* is the out-of-order scores for pairs with both elements appearing in only one of the two lists.

The final raw number of Pairs-Out-of-Order (POO) *z* is the sum of *z1*, *z2*, *z3*, *z4* and *z5*.

**inter.system.dis.sim.table.“extension”** If proximity measure is distance or similarity. Record format: *sys1 sys2 dis sim*.

**“distance|similarity”.dendrogram.“extension”**

**“distance|similarity”.matrix.“extension”**

**forsspss.sps.“extension”** Syntax for PC or MAC spss. You need to change the input file name in the second line of forsspss.sps.“extension” to reflect the correct input data file name and directory when you download the syntax file and data file to a PC or MAC. You may also need to change the drive name if you don’t put the input data file on drive c.

**“distance|similarity”.forsspss.dat.“extension”** Data file of PC or MAC spss.

### 5.1 Detailed Formulae for the Component $z$ -scores

$z1 = \frac{(N1)(N1+1)}{2} - \frac{m(m+1)}{2} - R1$  (In this expression,  $R1$  is sum of ranks of documents that are retrieved by  $sys1$  but not by  $sys2$ )

$z2 = \frac{(N2)(N2+1)}{2} - \frac{m(m+1)}{2} - R2$  (Where  $R2$  is sum of ranks of documents that are retrieved by  $sys2$  but not by  $sys1$ )

$z3$  = The return value of a function which recursively divides the common elements into two sets and then calculates the out-of-order scores for each of these sets iteratively.

$z4 = (N1 - m)(N2 - m)$

$z5 = \frac{1}{2} \left[ \frac{(N1-m-1)(N1-m)}{2} + \frac{(N2-m-1)(N2-m)}{2} \right]$

$z = z1 + z2 + z3 + z4 + z5$

### 5.2 Details of the Iterative Algorithm for $z3$

The algorithm is as follow: For two lists  $S$  and  $L$  with the same elements, we can calculate the out-of-order score by first splitting the list  $L$  into two halves,  $T$  and  $B$ , where  $T$  is the top half of the list and  $B$  is bottom half (It is not necessary for  $T$  and  $B$  to be exactly equal.) Now there are only two kinds of pairs.

1. **between split lists** : one item from  $T$ , the other from  $B$ .
2. **within split lists** : both items are from the same split list.

Let  $r_i$  be the rank of the  $i$ -th element of  $T$  in  $S$ , and let  $t$  be the size of  $T$ . For the first condition, the number of out-of-order pairs can be calculated by the following formula which is similar to the Wilcoxon rank-sum test:

$$z = \sum_{i=1}^t r_i - \frac{t(t+1)}{2}$$

For the second condition, we can again form two split lists. The number of out-of-order pairs for each of them can be calculated by splitting the list into top and bottom again (e.g., split  $T$  into  $top - T$  and  $bottom - T$ ; split  $B$  into  $top - B$  and  $bottom - B$ ) such that for each of them, there are only two conditions of out-of-order:

1. between newly split lists
2. within newly split lists

For the new condition 2, we can split the newly split lists again. This recursive splitting can be looped until the lists are un-splittable (i.e., there is only one item in each list).

## 6 Usage of the program

```
trec.dendrogrammer.09.sh -t [dis|sim|"new"] -i indir -o outdir -e extension
```

**-t dis** : Dendrogram is based on distance

**-t sim** : Dendrogram is based on similarity

**-t new** : Dendrogram is based on a formula; "new" should be the name of the program that uses the new formula; the input of this program should be z.table and the output should be a list in format: sys1 sys2 newscore

**-i indir** : Input files directory

**-o outdir** : Output files directory

**-e extension** : Output file names extension

**Caution:** inputdir and outputdir must be subdirectories of the current directory, otherwise you have to give the full path name, e.g.,  
trec.dendrogrammer.09.sh -t sim -i /usr/indata -o /usr/outdata -e abc

**output files:**

```
/usr/outdata/forspss.sps.abc  
/usr/outdata/inter.system.dis.sim.table.abc  
/usr/outdata/similarity.dendrogram.abc  
/usr/outdata/similarity.forspss.dat.abc  
/usr/outdata/similarity.matrix.abc  
/usr/outdata/sys.alpha.dictionary.abc  
/usr/outdata/z.table.abc
```

## 7 Reference

### References

- [1] Delattre, M. and Hansen, P. (1980) "Bicriterion cluster analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no.4, pp.277-291.
- [2] Hansen, P. and Delattre, M. (1978) "Computer-link cluster analysis by graph coloring," Journal of American Statistical Association, vol. 73, pp.397-403.
- [3] Kemeny, J.G. (1964) Random essays on mathematics, education and computers. Englewood Cliffs, N.J.: Prentice-Hall.

## 8 Adhoc Systems Compared

The following multipage table shows the systems ranked in order of decreasing similarity to the system whose name appears at the top of the column.

Rank	ACQADH	Brkly10	Brkly9	CLARTF	CLARTN	CnQst1	CnQst2
1	CrnlAE	pircs2	CrnlAE	CLARTN	CLARTF	INQ202	CnQst1
2	CrnlAL	CrnlAE	pircs1	CrnlAE	CrnlAE	CnQst2	pircs2
3	citym1	CrnlAL	CrnlAL	pircs2	pircs2	CrnlAL	INQ202
4	pircs2	pircs1	pircs2	citya1	citya1	pircs2	uwgcl1
5	citya1	citya1	citya1	citym1	citym1	INQ201	CrnlAL
6	siems1	citym1	Brkly10	CrnlAL	CrnlAL	CrnlAE	CrnlAE
7	pircs1	INQ202	citym1	pircs1	pircs1	siems1	pircs1
8	CLARTF	CnQst1	siems1	Brkly10	Brkly10	pircs1	INQ201
9	INQ202	siems1	INQ201	CnQst2	CnQst2	nyuge4	citya1
10	CnQst2	INQ201	INQ202	CnQst1	CnQst1	citri2	nyuge4
11	Brkly10	CLARTF	citri2	INQ202	nyuge4	citya1	citym1
12	nyuge4	CnQst2	citri1	nyuge4	INQ202	citym1	siems1
13	CnQst1	nyuge4	CLARTF	siems1	siems1	Brkly10	CLARTF
14	INQ201	citri2	CnQst1	INQ201	INQ201	citri1	Brkly10
15	gmu2	citri1	nyuge3	uwgcl1	uwgcl1	nyuge3	citri1
16	citri2	CLARTN	DCU951	nyuge3	DCU951	CLARTF	citri2
17	nyuge3	uwgcl1	CnQst2	DCU951	nyuge3	uwgcl1	gmu1
18	CLARTN	nyuge3	nyuge4	gmu1	citri1	gmu2	nyuge3
19	uwgcl1	Brkly9	CLARTN	citri1	citri2	padreA	CLARTN
20	citri1	DCU951	gmu2	citri2	Brkly9	DCU951	padreA
21	DCU951	gmu2	uwgcl1	gmu2	gmu2	gmu1	DCU951
22	Brkly9	padreA	padreA	Brkly9	gmu1	CLARTN	gmu2
23	virtu4	gmu1	virtu4	padreA	ACQADH	virtu4	padreZ
24	gmu1	padreZ	ACQADH	padreZ	padreA	padreZ	virtu4
25	padreZ	ACQADH	gmu1	ACQADH	padreZ	Brkly9	INTXT2
26	padreA	virtu4	padreZ	virtu4	virtu4	ACQADH	ACQADH
27	INTXT2	INTXT2	INTXT2	INTXT2	INTXT2	INTXT2	Brkly9
28	fsclt2	fsclt1	issah1	fsclt2	issah1	fsclt1	fsclt2
29	fsclt1	fsclt2	issah2	fsclt1	issah2	fsclt2	fsclt1
30	issah2	issah1	fsclt2	issah1	fsclt2	issah2	issah2
31	issah1	issah2	fsclt1	issah2	fsclt1	issah1	issah1
32	DCU952	DCU952	DCU952	DCU952	DCU952	DCU952	DCU952

Table 1: List of the ordered neighbors of each system in the Adhoc component of TREC 4, e.g., the tenth closest system to Brkly10 is INQ201

Rank	CrnlAE	CrnlAL	DCU951	DCU952	INQ201	INQ202	INTXT2
1	CrnlAL	CrnlAE	nyuge3	issah1	INQ202	INQ201	gmu1
2	siems1	siems1	INQ201	issah2	CrnlAL	CrnlAL	uwgcl1
3	citya1	INQ201	citri1	DCU951	siems1	siems1	CnQst2
4	pircs2	INQ202	CrnlAL	citri1	CrnlAE	pircs2	INQ202
5	pircs1	pircs2	INQ202	CLARTN	pircs1	CrnlAE	padreZ
6	citym1	pircs1	citri2	nyuge3	citri2	CnQst1	pircs2
7	INQ202	citya1	CrnlAE	citri2	pircs2	pircs1	CrnlAE
8	INQ201	citym1	pircs1	nyuge4	citri1	nyuge4	CrnlAL
9	Brkly10	CnQst1	nyuge4	CLARTF	CnQst1	nyuge3	padreA
10	CnQst1	citri2	gmu2	citya1	citya1	citri2	CLARTF
11	CLARTF	nyuge4	siems1	citym1	nyuge3	citya1	nyuge4
12	nyuge4	nyuge3	pircs2	CnQst2	citym1	citri1	Brkly10
13	citri2	gmu2	citya1	CnQst1	DCU951	citym1	CnQst1
14	citri1	citri1	CnQst1	pircs1	nyuge4	gmu2	citya1
15	CnQst2	Brkly10	citym1	INQ201	gmu2	CnQst2	siems1
16	nyuge3	DCU951	virtu4	pircs2	padreA	DCU951	citym1
17	DCU951	CnQst2	CnQst2	CrnlAE	CnQst2	padreA	INQ201
18	gmu2	CLARTF	padreA	INQ202	Brkly10	Brkly10	pircs1
19	CLARTN	padreA	Brkly10	CrnlAL	virtu4	virtu4	fsclt1
20	Brkly9	virtu4	CLARTF	uwgcl1	CLARTF	CLARTF	citri2
21	padreA	CLARTN	issah1	Brkly10	uwgcl1	uwgcl1	fsclt2
22	uwgcl1	Brkly9	issah2	padreA	Brkly9	gmu1	citri1
23	ACQADH	uwgcl1	CLARTN	Brkly9	gmu1	CLARTN	nyuge3
24	virtu4	gmu1	uwgcl1	siems1	CLARTN	Brkly9	virtu4
25	gmu1	ACQADH	Brkly9	padreZ	padreZ	padreZ	gmu2
26	padreZ	padreZ	gmu1	gmu2	ACQADH	ACQADH	DCU951
27	INTXT2	INTXT2	padreZ	virtu4	issah1	INTXT2	CLARTN
28	fsclt2	fsclt1	ACQADH	ACQADH	INTXT2	fsclt1	ACQADH
29	fsclt1	fsclt2	INTXT2	gmu1	issah2	fsclt2	Brkly9
30	issah2	issah2	fsclt1	INTXT2	fsclt1	issah2	issah1
31	issah1	issah1	fsclt2	fsclt1	fsclt2	issah1	issah2
32	DCU952	DCU952	DCU952	fsclt2	DCU952	DCU952	DCU952

Table 2: Previous table continued

Rank	citri1	citri2	citya1	citym1	fsclt1	fsclt2	gmu1
1	citri2	citri1	citym1	citya1	fsclt2	fsclt1	uwgcl1
2	INQ201	siems1	CrnlAE	CrnlAE	gmu1	gmu1	CnQst2
3	CrnlAL	INQ201	CrnlAL	CrnlAL	uwgcl1	uwgcl1	CnQst1
4	INQ202	CrnlAL	pircs1	pircs2	padreA	padreA	INQ202
5	nyuge3	INQ202	pircs2	pircs1	padreZ	padreZ	fsclt1
6	DCU951	nyuge3	INQ201	INQ201	CnQst1	CnQst1	fsclt2
7	CrnlAE	CrnlAE	INQ202	INQ202	CnQst2	CnQst2	pircs2
8	pircs1	pircs1	siems1	Brkly10	INQ202	INQ202	nyuge4
9	siems1	pircs2	Brkly10	CLARTF	pircs2	pircs2	CrnlAL
10	pircs2	CnQst1	CLARTF	siems1	INQ201	INQ201	padreZ
11	citya1	DCU951	CnQst1	CnQst1	INTXT2	INTXT2	INTXT2
12	CnQst1	citya1	citri1	nyuge4	nyuge4	nyuge4	CrnlAE
13	nyuge4	nyuge4	nyuge4	CnQst2	CrnlAL	CrnlAE	padreA
14	issah1	gmu2	citri2	citri1	CrnlAE	pircs1	CLARTF
15	citym1	citym1	CnQst2	citri2	Brkly10	CrnlAL	INQ201
16	CnQst2	CnQst2	nyuge3	nyuge3	pircs1	citya1	pircs1
17	gmu2	Brkly10	DCU951	CLARTN	citya1	Brkly10	citya1
18	Brkly10	padreA	CLARTN	DCU951	citri1	citri1	siems1
19	issah2	virtu4	uwgcl1	uwgcl1	citym1	citym1	citym1
20	padreA	issah1	gmu2	gmu2	citri2	citri2	Brkly10
21	CLARTF	CLARTF	Brkly9	Brkly9	nyuge3	CLARTF	nyuge3
22	uwgcl1	Brkly9	padreA	padreA	CLARTF	nyuge3	citri1
23	Brkly9	issah2	gmu1	ACQADH	siems1	siems1	citri2
24	virtu4	uwgcl1	ACQADH	gmu1	DCU951	DCU951	virtu4
25	CLARTN	CLARTN	virtu4	padreZ	gmu2	CLARTN	DCU951
26	gmu1	gmu1	padreZ	virtu4	CLARTN	gmu2	gmu2
27	padreZ	ACQADH	INTXT2	INTXT2	issah2	issah2	CLARTN
28	ACQADH	padreZ	issah1	issah1	virtu4	virtu4	Brkly9
29	INTXT2	INTXT2	issah2	issah2	Brkly9	Brkly9	ACQADH
30	fsclt2	fsclt2	fsclt2	fsclt2	issah1	issah1	issah2
31	fsclt1	fsclt1	fsclt1	fsclt1	ACQADH	ACQADH	issah1
32	DCU952	DCU952	DCU952	DCU952	DCU952	DCU952	DCU952

Table 3: Previous table continued

Rank	gmu2	issah1	issah2	nyuge3	nyuge4	padreA	padreZ
1	CrnlAL	issah2	issah1	nyuge4	nyuge3	INQ202	uwgcl1
2	siems1	citri1	citri1	INQ202	INQ202	INQ201	gmu1
3	INQ202	citri2	citri2	INQ201	CrnlAL	CrnlAL	CnQst2
4	nyuge3	DCU951	DCU951	CrnlAL	pircs2	CnQst1	pircs2
5	INQ201	nyuge3	nyuge3	DCU951	CrnlAE	siems1	CnQst1
6	virtu4	INQ201	INQ201	citri2	INQ201	pircs2	CrnlAE
7	CrnlAE	citya1	pircs1	citri1	CnQst1	CrnlAE	CrnlAL
8	nyuge4	pircs1	INQ202	gmu2	siems1	nyuge3	INQ202
9	citri2	INQ202	citya1	siems1	pircs1	nyuge4	Brkly10
10	DCU951	CnQst1	CnQst1	pircs2	citya1	citri1	CLARTF
11	pircs2	pircs2	nyuge4	CrnlAE	citri2	pircs1	citya1
12	pircs1	nyuge4	pircs2	pircs1	citym1	CnQst2	citym1
13	CnQst1	CnQst2	CnQst2	citya1	CnQst2	citya1	nyuge4
14	citya1	citym1	citym1	CnQst1	gmu2	citri2	pircs1
15	citri1	padreA	CrnlAL	virtu4	citri1	uwgcl1	INTXT2
16	citym1	CrnlAL	CrnlAE	citym1	DCU951	gmu1	fsclt1
17	CnQst2	CrnlAE	padreA	padreA	Brkly10	DCU951	fsclt2
18	padreA	uwgcl1	uwgcl1	CnQst2	CLARTF	citym1	INQ201
19	Brkly10	DCU952	CLARTF	Brkly10	uwgcl1	virtu4	padreA
20	CLARTF	CLARTF	siems1	CLARTF	padreA	gmu2	siems1
21	Brkly9	CLARTN	DCU952	CLARTN	virtu4	Brkly10	citri1
22	uwgcl1	Brkly10	CLARTN	uwgcl1	gmu1	CLARTF	CLARTN
23	ACQADH	siems1	Brkly10	Brkly9	CLARTN	fsclt1	citri2
24	CLARTN	gmu1	gmu1	gmu1	padreZ	fsclt2	DCU951
25	gmu1	Brkly9	fsclt2	issah1	ACQADH	INTXT2	nyuge3
26	padreZ	gmu2	fsclt1	issah2	Brkly9	padreZ	gmu2
27	INTXT2	padreZ	gmu2	ACQADH	INTXT2	CLARTN	Brkly9
28	fsclt1	fsclt2	Brkly9	padreZ	issah2	Brkly9	ACQADH
29	fsclt2	fsclt1	padreZ	INTXT2	issah1	issah1	virtu4
30	issah2	virtu4	virtu4	fsclt1	fsclt1	issah2	issah2
31	issah1	INTXT2	ACQADH	fsclt2	fsclt2	ACQADH	issah1
32	DCU952	ACQADH	INTXT2	DCU952	DCU952	DCU952	DCU952

Table 4: Previous table continued

Rank	pircs1	pircs2	siems1	uwgcl1	virtu4
1	pircs2	pircs1	CrnlAL	CnQst2	gmu2
2	CrnlAE	CrnlAE	CrnlAE	gmu1	siems1
3	CrnlAL	CrnlAL	INQ201	CnQst1	INQ202
4	citya1	INQ202	INQ202	pircs2	nyuge3
5	INQ201	citya1	citri2	INQ202	INQ201
6	citym1	citym1	pircs1	nyuge4	CrnlAL
7	siems1	INQ201	pircs2	citya1	nyuge4
8	INQ202	CnQst1	CnQst1	CrnlAE	DCU951
9	CnQst1	Brkly10	gmu2	padreZ	CnQst1
10	Brkly10	siems1	citya1	citym1	CrnlAE
11	citri1	nyuge4	nyuge3	CLARTF	citri2
12	citri2	CnQst2	citri1	CrnlAL	pircs2
13	CnQst2	CLARTF	nyuge4	Brkly10	padreA
14	nyuge4	citri2	citym1	INQ201	pircs1
15	nyuge3	citri1	Brkly10	pircs1	CnQst2
16	DCU951	nyuge3	CnQst2	padreA	citya1
17	CLARTF	DCU951	DCU951	siems1	citri1
18	gmu2	gmu2	virtu4	INTXT2	citym1
19	Brkly9	uwgcl1	CLARTF	CLARTN	Brkly10
20	CLARTN	CLARTN	padreA	citri1	CLARTF
21	padreA	padreA	Brkly9	citri2	uwgcl1
22	uwgcl1	Brkly9	uwgcl1	DCU951	gmu1
23	virtu4	gmu1	CLARTN	nyuge3	CLARTN
24	gmu1	virtu4	gmu1	fsclt1	Brkly9
25	ACQADH	padreZ	ACQADH	fsclt2	ACQADH
26	padreZ	ACQADH	padreZ	gmu2	INTXT2
27	INTXT2	INTXT2	INTXT2	virtu4	padreZ
28	issah2	fsclt2	fsclt1	Brkly9	fsclt1
29	issah1	fsclt1	fsclt2	ACQADH	fsclt2
30	fsclt2	issah1	issah2	issah1	issah2
31	fsclt1	issah2	issah1	issah2	issah1
32	DCU952	DCU952	DCU952	DCU952	DCU952

Table 5: Previous table continued

## 9 Routing Systems Compared

Rank	ACQROU	Brkly11	Brkly12	CrnlRE	CrnlRL	HNC11	HNC21
1	itidp1	Brkly12	xerox2	CrnlRL	CrnlRE	HNC21	HNC11
2	xerox2	INQ204	xerox1	cityr1	cityr1	itidp1	itidp1
3	xerox1	xerox2	pircsC	cityr2	INQ204	losPA3	losPA3
4	itidp2	xerox1	pircsL	INQ204	cityr2	xerox2	xerox2
5	INQ204	pircsL	nyuge1	xerox2	pircsC	xerox1	xerox1
6	CrnlRL	pircsC	nyuge2	xerox1	pircsL	nyuge1	nyuge1
7	pircsC	INQ203	losPA2	pircsC	xerox2	pircsC	nyuge2
8	pircsL	CrnlRL	losPA3	pircsL	xerox1	pircsL	losPA2
9	nyuge1	CrnlRE	INQ204	INQ203	INQ203	losPA2	pircsC
10	cityr1	itidp1	INQ203	itidp1	itidp1	nyuge2	pircsL
11	nyuge2	cityr1	cityr2	nyuge2	nyuge2	cityr1	cityr2
12	CrnlRE	cityr2	cityr1	nyuge1	nyuge1	cityr2	cityr1
13	INQ203	nyuge2	itidp1	losPA3	losPA3	CrnlRE	CrnlRE
14	HNC21	nyuge1	CrnlRE	losPA2	losPA2	INQ204	INQ204
15	cityr2	losPA2	CrnlRL	virtu3	virtu3	CrnlRL	CrnlRL
16	HNC11	itidp2	Brkly11	itidp2	itidp2	INQ203	INQ203
17	losPA3	losPA3	itidp2	UCF100	UCF100	itidp2	itidp2
18	losPA2	virtu3	HNC11	HNC11	Brkly12	Brkly12	Brkly12
19	Brkly12	HNC11	virtu3	Brkly12	HNC11	virtu3	virtu3
20	UCF100	UCF100	HNC21	HNC21	HNC21	UCF100	UCF100
21	virtu3	HNC21	UCF100	uwgcl1	Brkly11	Brkly11	ACQROU
22	Brkly11	uwgcl1	uwgcl1	Brkly11	uwgcl1	uwgcl1	Brkly11
23	ORAdL2	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU	uwgcl1
24	uwgcl1	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2
25	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1

Table 6: List of the ordered neighbors of each system in the Routing component of TREC 4, e.g., the tenth closest system to ACAROU is cityr1

Rank	INQ203	INQ204	ORAdL1	ORAdL2	UCF100	cityr1	cityr2
1	INQ204	INQ203	ORAdL2	ORAdL1	pircsL	cityr2	cityr1
2	cityr2	cityr1	xerox2	nyuge2	pircsC	INQ204	INQ204
3	cityr1	pircsL	xerox1	nyuge1	cityr1	pircsL	pircsL
4	pircsL	cityr2	INQ203	losPA3	cityr2	pircsC	pircsC
5	pircsC	pircsC	cityr2	xerox2	xerox2	INQ203	INQ203
6	xerox2	xerox2	INQ204	xerox1	xerox1	CrnlRL	xerox2
7	xerox1	xerox1	pircsC	itidp1	nyuge1	xerox2	xerox1
8	CrnlRL	CrnlRL	pircsL	pircsC	nyuge2	xerox1	CrnlRE
9	CrnlRE	nyuge2	cityr1	pircsL	losPA3	CrnlRE	CrnlRL
10	nyuge2	CrnlRE	nyuge2	cityr1	losPA2	nyuge2	nyuge2
11	nyuge1	nyuge1	nyuge1	cityr2	INQ204	nyuge1	nyuge1
12	losPA2	itidp1	losPA3	losPA2	INQ203	losPA2	losPA3
13	itidp1	losPA2	CrnlRE	INQ204	virtu3	losPA3	losPA2
14	losPA3	losPA3	losPA2	CrnlRE	CrnlRE	itidp1	itidp1
15	virtu3	itidp2	itidp1	INQ203	CrnlRL	virtu3	virtu3
16	itidp2	virtu3	CrnlRL	CrnlRL	itidp1	itidp2	UCF100
17	Brkly12	Brkly12	itidp2	itidp2	HNC11	UCF100	itidp2
18	UCF100	UCF100	HNC21	HNC21	itidp2	Brkly12	Brkly12
19	HNC11	HNC11	Brkly12	HNC11	Brkly12	HNC11	HNC11
20	HNC21	HNC21	UCF100	virtu3	HNC21	HNC21	HNC21
21	Brkly11	Brkly11	HNC11	Brkly12	uwgcl1	uwgcl1	uwgcl1
22	uwgcl1	uwgcl1	virtu3	UCF100	Brkly11	Brkly11	Brkly11
23	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU
24	ORAdL2	ORAdL2	Brkly11	Brkly11	ORAdL2	ORAdL2	ORAdL2
25	ORAdL1	ORAdL1	uwgcl1	uwgcl1	ORAdL1	ORAdL1	ORAdL1

Table 7: Previous table continued

Rank	itidp1	itidp2	losPA2	losPA3	nyuge1	nyuge2
1	itidp2	itidp1	losPA3	losPA2	nyuge2	nyuge1
2	xerox2	xerox2	xerox1	nyuge1	pircsC	pircsL
3	xerox1	xerox1	xerox2	xerox1	pircsL	pircsC
4	nyuge1	pircsL	pircsC	xerox2	losPA3	xerox2
5	nyuge2	pircsC	pircsL	nyuge2	xerox2	losPA3
6	pircsC	nyuge2	nyuge1	pircsC	xerox1	xerox1
7	pircsL	INQ204	nyuge2	pircsL	losPA2	losPA2
8	INQ204	nyuge1	cityr2	cityr2	INQ204	INQ204
9	cityr1	cityr1	cityr1	cityr1	cityr1	cityr1
10	cityr2	INQ203	INQ204	itidp1	cityr2	cityr2
11	losPA3	cityr2	itidp1	INQ204	itidp1	itidp1
12	CrnlRL	CrnlRL	INQ203	CrnlRE	INQ203	INQ203
13	CrnlRE	losPA2	CrnlRE	CrnlRL	CrnlRE	CrnlRE
14	losPA2	CrnlRE	CrnlRL	INQ203	CrnlRL	CrnlRL
15	INQ203	losPA3	virtu3	virtu3	virtu3	virtu3
16	HNC11	Brkly12	Brkly12	Brkly12	itidp2	itidp2
17	HNC21	HNC11	UCF100	UCF100	Brkly12	Brkly12
18	Brkly12	virtu3	itidp2	HNC11	UCF100	UCF100
19	virtu3	HNC21	HNC11	itidp2	HNC11	HNC11
20	UCF100	UCF100	HNC21	HNC21	HNC21	HNC21
21	Brkly11	Brkly11	uwgcl1	uwgcl1	uwgcl1	uwgcl1
22	ACQROU	ACQROU	Brkly11	Brkly11	Brkly11	Brkly11
23	uwgcl1	uwgcl1	ACQROU	ACQROU	ACQROU	ACQROU
24	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2
25	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1

Table 8: Previous table continued

Rank	pircsC	pircsL	uwgcl1	virtu3	xerox1	xerox2
1	pircsL	pircsC	virtu3	pircsL	xerox2	xerox1
2	xerox2	xerox2	pircsL	pircsC	pircsL	pircsL
3	xerox1	xerox1	pircsC	cityr2	pircsC	pircsC
4	nyuge2	INQ204	cityr1	cityr1	INQ204	INQ204
5	INQ204	nyuge2	cityr2	xerox2	losPA2	losPA2
6	nyuge1	nyuge1	xerox2	xerox1	losPA3	losPA3
7	losPA2	cityr2	nyuge2	nyuge2	nyuge2	nyuge2
8	cityr2	cityr1	xerox1	nyuge1	nyuge1	nyuge1
9	cityr1	losPA2	nyuge1	losPA3	cityr2	cityr2
10	losPA3	INQ203	INQ204	INQ204	cityr1	cityr1
11	INQ203	losPA3	losPA2	losPA2	itidp1	itidp1
12	CrnlRL	CrnlRL	INQ203	CrnlRE	INQ203	INQ203
13	CrnlRE	CrnlRE	CrnlRL	CrnlRL	CrnlRE	CrnlRE
14	itidp1	itidp1	losPA3	INQ203	CrnlRL	CrnlRL
15	virtu3	virtu3	CrnlRE	UCF100	virtu3	virtu3
16	itidp2	itidp2	UCF100	itidp1	itidp2	itidp2
17	Brkly12	Brkly12	itidp1	itidp2	Brkly12	Brkly12
18	UCF100	UCF100	Brkly12	Brkly12	UCF100	UCF100
19	HNC11	HNC11	itidp2	HNC11	HNC11	HNC11
20	HNC21	HNC21	HNC11	HNC21	HNC21	HNC21
21	uwgcl1	uwgcl1	HNC21	uwgcl1	Brkly11	Brkly11
22	Brkly11	Brkly11	Brkly11	Brkly11	uwgcl1	uwgcl1
23	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU	ACQROU
24	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2	ORAdL2
25	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1	ORAdL1

Table 9: Previous table continued