Report

Phylogenetic tree of the "Nigerian Prince" e-mail scam

Author(s):
Gonnet, Gaston H.

Publication Date:
2005

Permanent Link:
https://doi.org/10.3929/ethz-a-006665923

Rights / License:
In Copyright - Non-Commercial Use Permitted
Phylogenetic tree of the "Nigerian Prince" email scam

Most people have received messages which at some point or other state "I would like to deposit 20,000,000.00 dollars in your account...". This is a common email fraud attempt which has been named the "Nigerian Prince" scam since the earliest versions of it were sent by a supposedly such member of royalty. The United States Postal Service has issued a warning on the subject and a google search will show hundreds of related hits.

The messages, although very different, have a similar pattern. In this bio-recipe we will try to construct the phylogenetic tree of these messages and hence reconstruct how it is likely that people have copied and modified the original message(s) and created new versions. If the authors had copied from a single source and then modified the messages to their own taste, this procedure would retrace the tree quite well. However it is possible that messages have been merged or copied from several sources, obscuring the process. The latter would be equivalent to lateral transfer of genetic information. It is also possible that similar messages have been reinvented from scratch. In any case, the whole procedure is an interesting example that may serve as a pattern for dealing with text, for constructing phylogenetic trees from diverse sources, separating the members in clusters and on some special forms of alignment. The resulting tree is also interesting in itself. Applications outside molecular biology are also possible, e.g. the construction of a phylogenetic tree of some piece of software that underwent development at different places/times, would follow exactly the same steps.

Reading the data: email messages

The raw email messages were collected for some time and kept in a standard plain text format. We read this file as is, and do all the processing in Darwin.

all := ReadRawFile( '/home/darwin/bio-recipes/NigerianPrince/mbox' );
length(all);

1095228

To store and process the messages we will define a class called Message. If it receives a single argument, it assumes that it is a raw message and proceeds to parse it. It is always convenient to write a method that will print the instances of the class, and finally we run CompleteClass which will generate some standard methods automatically.

Message := proc( id:string, shortid:string, subject:string,
header:string, body:string )

if nargs=1 then
  if id{1..5} <> 'From' then
    error(id,'message does not start with From') fi;

end:

CompleteClass(Message):
i0 := SearchString(' \n', id);
if i0 < 0 then error(id,'does not have a single line') fi;

id1 := id[1..i0];

i1 := SearchString(' \n\n', id);
if i1 < 0 then error(id,'message does not appear to have a body') fi;

header1 := id[i1+1..i1+i1];

i2 := CaseSearchString(' \nSubject: ', header1);
if i2 < 0 then subject1 := ''
else i3 := SearchString(' \n', i2+2+header1);

subject1 := header1[i2+1..i2+i2+2];
while length(subject1) > 4 and
upper(name(subject1)) = 'RE' do
    subject1 := 4+subject1 od
fi;

body1 := id[i1+3..-1];
Message( idi, ShortId(idi), subject1, header1,
    ConvertMessage(body1) )

elif nargs=5 then noeval(procname(args))
else error(args,'invalid message format') fi
end:

Message_print := proc( m:Message )
    printf("\n\%s (\%s)\nSubject: \%s\nbody: \%s\n",
        m[ id ], m[ shortid ], m[ subject ], m[ body ] )
end:

CompleteClass(Message):=

Converting the text to a comparable form

The following function converts a general text into a form suitable for comparison and alignment. The messages are written over an alphabet of 26 characters, in upper and lower case and numbers and special symbols are also used. The mapping of lowercase characters to uppercase characters, or the grouping of several symbols in the same class (e.g. punctuation) can be done with a mapping function included in the Dayhoff matrices.

ConvertMessage does this conversion, character by character. We search each character of the message in a string which contains all letters and digits (case-independent search). This is a simple and efficient way to find out if our character is a letter or a digit. The boolean variable spacing is used to keep only one spacing character per group. We will keep the first one, the rest will be just ignored. Finally, the result is accumulated in r, which may be shorter than the original string. One final exception is that in no case underscores characters are allowed in sequence alignment. The reason for this is that underscores indicate gaps, so if they would be allowed as sequence characters it would not be possible to determine where deletions happen. We transform all underscores to dashes.

ConvertMessage := proc( s:character )
    ls := length(s);
    r := CreateString(ls);
    lr := 0;
    spacing := true;

for i to 1s do
  k := SearchString(s[i], 'abcdefghijklmnopqrstuvwxyz');
  if k >= 0 then
    lr := lr+1; r[lr] := s[i]; spacing := false
  elif not spacing then
    lr := lr+1;
    r[lr] := If( s[i] = '-', ', ', s[i] );
    spacing := true
  fi
od;
r[1..lr]
end:

At this point it is convenient to define the function that will map the characters to positions in the scoring matrix. In our terminology this is a Mapping function. Numbers are rare in these messages, and their actual value does not have a very relevant meaning, so we will group all digits in a single symbol. The dimension of the scoring matrix will be 28. This is composed of 26 letters, 1 symbol for any of the 10 digits and 1 symbol for spacing and punctuation. Using the convenience of SearchString this is written as:

Dim := 28:
map := proc( ch: string )
  if length(ch) <> 1 then error('invalid argument') fi;
  k := SearchString(ch, 'abcdefghijklmnopqrstuvwxyz');
  if k >= 0 then k+1
  else SearchString(ch, '0123456789') >= 0 then 27
  else 28 fi
end:

The function ShortId is an auxiliary function to select the first part of the sending email address. This is typically the userid, and it is intended as a quick identification of the email message. It searches for an "@" or a blank as a separator and return whatever comes after the "From" until the separator.

ShortId := proc( s: string )
  if s[1..5] = 'From' then procline( 5+s )
  else i := SearchString(' ', s );
    if i > 0 then return( s[1..i] ) fi;
    i := SearchString( ' ', s );
    if i > 0 then return( s[1..i] ) fi;
    s fi
end:

Now we are ready to process all the messages in the string "all". Each message starts with a new line with the word "From" followed by a space, which is the standard email convention. We place all the messages in a list.

msgs := [];
do i := CaseSearchString( '\nFrom ', all );
  if i < 0 then
    if length(all) > 10 and all[1..5] = 'From ' then
      msgs := append( msgs, Message(all) ) fi;
    break
  else msgs := append( msgs, Message(all[1..i+1]) ) ;
    all := i+1+all
  fi
od:
For curiosity and checking, we can print one message.

```plaintext
print(msgs[2]);
```

From petersigho@management.com Fri, 3 Nov 2000 21:27:40 (petersigho)
Subject: INVESTMENT PROPOSAL

body: ECO INTERNATIONAL BANK HEADQUARTERS
PLOT 84 AJOSE ADEOGUN STREET
VICTORIA ISLAND LAGOS
ATTN: PRESIDENT/CHAIRMAN

STRICTLY A PRIVATE BUSINESS PROPOSAL
I am DR PETER IGHO a manager in the Bills and Exchange at the Foreign Remittance Department of the ECO INTERNATIONAL BANK. I am writing this letter to ask for your support and cooperation to carry out this business opportunity in my department. We discovered an abandoned sum of $15,000,000.00 fifteen million United States Dollars only in an account that belongs to one of our foreign customers who died along with his entire family of a wife and two children in November 1997 in a plane crash. Since we heard of his death, we have been expecting his next of kin to come over and put claims for his money as the heir, because we cannot release the fund from his account unless someone applies for claim as the next of kin to the deceased as indicated in our banking guidelines. Unfortunately, neither their family member nor distant relative has ever appeared to claim the said fund. Upon this discovery, I and other officials in my department have agreed to make business with you and release the total amount into your account as the heir of the fund since no one came for it or discovered he maintained account with our bank, otherwise the fund will be returned to the banks treasury as unclaimed fund. We have agreed that our ratio of sharing will be as stated thus: 20 for you as foreign partner, 75 for us the officials in my department and 5 for the settlement of all local and foreign expenses incurred by us and you during the course of this business. Upon the successful completion of this transfer, I and one of my colleagues will come to your country and mind our share. It is from our 75 we intend to import Agricultural Machineries into my country as a way of recycling the fund. To commence this transaction, we require you to immediately indicate your interest by a return e-mail and enclose your private contact telephone number, fax number full name and address and your designated bank coordinates to enable us file letter of claim to the appropriate departments for necessary approvals before the transfer can be made. Note also, this transaction must be kept strictly confidential because of its nature. I look forward to receiving your prompt response.

FAX NUMBER: 234-1-759291
Best Regard
DR PETER IGHO

Eliminating identical duplicates

Pairs of messages which are identical serve no purpose whatsoever. They will be part of the same subtree, they contribute no useful information about mutations (there are none) but cost a lot of additional computation. Hence it is desirable to eliminate the exact repetitions. The following loop compares every pair of messages and if they are identical removes the second one from the list. Since the list may shorten during the process, the loops are controlled by a while condition.
for i while i <= length(msgs) do
    for j from i+1 while j <= length(msgs) do
        if msgs[i,body] = msgs[j,body] then
            printf('msg %s is a duplicate of
                   %s; removed
',
                   msgs[j,id], msgs[i,id]);
            msgs := [ op(msgs[1..j-1]), op(msgs[j+1..]) ];
        fi
    od od;

msg From peterjohn2004@go.com Wed Feb 26 12:10 MET 2003 is a duplicate of
    From peterjohn@ny.com Wed Feb 26 10:43 MET 2003; removed
msg From benwills9@netzero.com Sat Apr  5 12:38 MET 2003 is a duplicate of
    From benwills11@netzero.com Sat Apr  5 12:28 MET 2003; removed

As expected the exact duplicates come from (nearly) the same sender and at about the
same time. This is expected from spam.

Building the Dayhoff matrices for this particular data

Running the all-against-all of 348 messages requires too much time for a normal
interactive session (60378 long alignments). We will set a flag, named PartialRun,
under which we can run just a few entries to show the process while the complete results
are read from a file. Conversely, if PartialRun is not set, the complete computation will
be run, and the files of results will be stored.

PartialRun := true;

PartialRun := true

The Dayhoff matrices used for protein alignments are completely useless in this case.
The symbols do not correspond to amino acids, and their mutation probabilities will
certainly differ. This is a situation that may be encountered quite often in biology too
when we are faced with sequences which are not average proteins. So the following
analysis is also very relevant for biology, in particular it is the standard procedure to
derive Dayhoff matrices from new data.

The procedure for finding the similarity/scoring matrices is iterative. It repeats the
following steps:

1. (first time only) Select an initial Dayhoff matrix which is sufficiently diagonally
dominant. That is, identical matches will be positive and mismatches will be
   negative. We start with a matrix at PAM 30 which has all mutations equally likely.
   We will call this matrix DM1.

2. Align all the sequences against each other with this similarity matrix.

3. Select a subset of the alignments which ensures that there is no relative over-
   representation of some evolutionary event. This can be achieved by using a circular
   tour of the sequences with accumulated maximal similarity. It can also be achieved
   with the methods described for the Unbiased selection of sample alignments

4. From each of the selected alignments, count the matches and mismatches per
   symbol pair. Mismatches are counted 1/2 for each pair, as we do not know which
   one mutated into which.
(5) Compute the frequencies, mutation matrix and Dayhoff matrices from the Counts matrix. (Darwin's ComputeDayMatrices command.)

(6) Repeat from step 2 with these new matrices until convergence. Convergence is normally very fast, and in this example we will run a single iteration.

For the initial matrix, it is easier to use the darwin function CreateDayMatrices which requires a matrix of counts for each pair of symbols. We will produce a synthetic counts matrix with all 1's except for the diagonal (10's).

```plaintext
SymCounts := CreateArray(1..Dim,1..Dim,1);
for i to Dim do SymCounts[i,i] := 10 od:
CreateDayMatrices( SymCounts, map, type=Alphabetic );
DM1 := SearchDayMatrix(30,dms);
```

now we run the complete or restricted all-against-all matching.

```plaintext
n := If( PartialRun, 6, DM[ TotEntries] );
AllAll := CreateArray(1..n,1..n):
Dist := CreateArray(1..n,1..n):
for i to n do
   for j from i+1 to n do
      AllAll[i,j] := AllAll[j,i] := Align(msgs[i,body],msgs[j,body],DM1);
      Dist[i,j] := Dist[j,i] := Score;
   od od:
maxdist := max(Dist);
```

```plaintext
n := 6
maxdist := 14539.1689
```

Find a circular tour of the leaves to collect mutation information

The set of all messages, assuming that they are all related, will form a phylogenetic tree. This tree is at this point unknown, (is what we want to construct). An excellent way to inspect all the pairs of leaves of the tree is by doing a circular tour of the tree in any of its planar representations. A circular tour that considers each pair of leaves in the tour and uses the information of the path that connects both leaves will consider each of the tree branches exactly twice. This is very important when we do not want to over-represent information. Notice that an all-against-all, would use some edges much more than others.
Circular tour of an unrooted tree. The pairs of nodes are 
(A,B), (B,C), (C,D), (D,E), (E,F), (F,G) and (G,A)

If shorter edges mean higher similarity, a planar tour has minimum total sum of edges 
and maximum total similarity. A circular tour with maximal similarity can be found by 
using the Traveling Salesman Problem solution (TSP). The TSP function minimizes a 
tour, so we will complement the similarities so that by finding a minimum of the 
complements, we find the desired maximum. For convenience, we complement against 
the maximum so that all distances end up positive.

```plaintext
for i to n do for j from i+1 to n do
    Dist[i,j] := Dist[j,i] := maxdist - Dist[i,j] od od;

for t := [1, 3, 5, 4, 2, 6] do
    for j := [op(t), t[1]] do
        for i to n do
            DPS := DynProgStrings( AllAll[ t[i], t[i+1] ] );
            for j to length(DPS[2]) do
                i1 := map(DPS[2][j]);
                i2 := map(DPS[3][j]);
                Counts[ i1, i2 ] := Counts[ i1, i2 ] + 1/2;
                Counts[ i2, i1 ] := Counts[ i2, i1 ] + 1/2;
                od;
            od;
        od;
    od;
```

For programming convenience we have added the first sequence of the tour to the end. 
We are now ready to follow the tour and for each alignment count all the matches in a 
Count matrix.

```plaintext
Counts := CreateArray(1..Dim, 1..Dim, if(PartialRun, 1, 0));
if PartialRun then for i to Dim do Counts[ i, i ] := 10 od fi;
for i to n do
    DPs := DynProgStrings( AllAll[ t[i], t[i+1] ] );
    for j to length(DPs[2]) do
        i1 := map(DPS[2][j]);
        i2 := map(DPS[3][j]);
        Counts[ i1, i2 ] := Counts[ i1, i2 ] + 1/2;
        Counts[ i2, i1 ] := Counts[ i2, i1 ] + 1/2;
    od;
od;
```

and now we compute the new Dayhoff matrices.
CreateDayMatrices(Counts,map,type=Alphabetical);

For the partial runs we usually have too few sequences, which leave some counts empty. When there are empty counts, it becomes impossible to compute the scoring matrix. So for the partial run we start with ones on each count and 10 on each diagonal to prevent empty counts.

**Compute the all-against-all distance matrix**

Now we are ready (by having a better scoring matrix) to align the messages and estimate their distances. This is done with the function Align when it receives an array of Dayhoff matrices instead of a single one (e.g. DMS). Align will compute the estimated distance and its variance for every pair of messages. The distance in this case, corresponds to a Markovian model of random independent mutations (which is not exactly what happens to the messages, but an approximation). A final technical problem is caused by alignments which are spurious, very good, and very short. This could happen by a very short phrase which appears in two unrelated messages. E.g. "thousand united states dollars". These short, good matches falsify the distance computations. To prevent this from happening, we use a mode of Align which forces an alignment of at least 400 amino acids of each message (parameter MinLength(400)).

for i to n do
  for j from i+1 to n do
    AllAll[i,j] := AllAll[j,i] :=
    Align(msgs[i, body], msgs[j, body], DMS, MinLength(400));
  od;
end;

bytes alloc=39200000, time=255.120

If we are running a partial run, we will now read the full results from a pre-prepared file. We will also reset the number of messages and recompute the Dayhoff matrices. If this was a complete run, then we will create the file of results for future partial runs.

if PartialRun then
  ReadProgram( 'NigerianPrince/AllAll.drw' );
else OpenWriting( 'NigerianPrince/AllAll.drw' );
  printf( 'Counts := %d\n', Counts );
  printf( 'CreateDayMatrices(Counts,map,type=Alphabetical);\n' );
  printf( 'AllAll := CreateArray(1..n,1..n);\n' );
  printf( 'AllAll:=%A\n', AllAll );
  OpenWriting( terminal );
fi;

bytes alloc=43200000, time=294.830

It is interesting to see some alignments at different PAM distances. To make this easier we write a function which selects the pair of messages which are at a distance as close as possible to a given one and prints their alignment. This will allow us to test the limit of quality of the alignments.
PrintAlignmentAt := proc( d::positive )
best := AllAll[1,2];
for i to n do for j from i+1 to n do
if |best[ PamDistance]−d| > |AllAll[ i,j,PamDistance]−d| then
    best := AllAll[ i,j] fi od od;
print( best )
end:

We now print alignments close to 30, 50 and 70 to see where the boundary is.

PrintAlignmentAt( 30 );

lengths=427,456 simil=1025.1, Pam_dist=30.2968, identity=45.4%
assist us, to transfer the sum of Twenty Million Five

Hundred Thousand United States Dollars 20,500,000)into his her a

count. This fund resulted from an over-invoiced bill from contrac

ted awarded by us under the budget allocation to my Minist

t_for and executed for on behalf of my Minist

ry this bill has been approved for payment by the concerned authorities. The

contract has since been executed, commissioned and the

contract was supposed to be awarded to two foreign

contractors to the tune of US$180,000,000.00 One hundred and Eighty Million

paid the actual cost of the contract. W

nited States Dollars) But in the course of negotiation, the contract w

PrintAlignmentAt( 50 );

lengths=531,576 simil=416.0, Pam_dist=50, identity=36.5%
of Contract Award Committee with the Federal Ministry of Agricul

OF CONTRACT AWARD AND MONITORING COMMITTEE OF THE MINISTRY OF

ture. By the virtue of our position and the power best

owed us by the government, we carefully and deliberately over invoice

ED BY THE MAURITIUS GOVERNMENT IS TO PROVIDE IDE
d the value of some contract |

THE BASIC AMENITIES, SOCIAL RECREATIONAL ACTIVITIES IN URBAN AND
RURAL AREAS, THIS PROGRAM INCLUDES ASSISTANCE TO DEPRIVED LOCAL COMMUNITIES AND

(5 output lines skipped)

FURTHERMORE FROM THIS PROJECTS WE HAVE BEEN ABLE TO SECURED SOME REASONABLE AMOUNT

Twenty-Two Million five Hundred Thousand United State Dollars US$ 21,800,000.00) Now the contracts have been fully executed and commission and pay

ment are about to be made to all the contractors who have successfully executed

their contracts, the over invoicing

PrintAlignmentAt(70);

lengths=434,594 simil=459.0, PAM_dist=70, identity=39.7%

account will be compensated with twenty percent 20% of the total sum if you

accounted in a ny part of the world, which you prove

will stand as the beneficiary.

ide, will then facilitate the transfer of this money to you as the beneficiary.

of the fun d Eighty x of kin of Mr. Barry Kelly. The money will be paid into your account for us to s

- percent 80% for we the officials involved, N

hare in the ratio of 60% for me and 40% for you. There is no risk at all.

ote that we have done our homework very well in our country, this transaction is

the paper work for this transaction will

safe and guaranteed. If you are interested please send down

successful execution of this transaction. If you are interested, please rep

to us account information, so that we will immediately seek an approval o
ly immediately via the private email a

f the fund on your be half from the

address below: Upon your response, I shall then provide you with more details

relevant

and relevant

We can see that the boundary is fuzzy, but at 50 there are very large portions of text which have been duplicated whereas at 70 the relation is dubious. Consequently we will choose 50 as the limiting distance for a trustworthy relation. It is interesting to notice that in Biology we are used to consider and study homologies at much larger distances, up to 250 in a reliable way. Clearly the techniques that have been developed are much more sensitive than the human eye.

Now that we have computed all the distances, we will produce the full tree.

Labels := [ seq( m[shortid], m=-msgs )];
Seqs := [ seq( m[body], m=-msgs )];
tree := PhylogeneticTree( Seqs, Labels, Distance, AllAll );
DrawTree( tree, Unrooted, CrossReference );

The full tree is quite large and impossible to read. To improve this we will do several things. First we will not print branch lengths nor nodes. In these cases, the Phylogram style is the most readable format. We will use the reordering that makes the left side the largest one so that the top of the tree appears more organized. Because the tree is so large
we will trim it down by removing the entries which are 80% or more identical. To do this we will use the fact that we have constructed a tree based on distances, and hence very similar sequences will have a very short distance and hence will be neighbour leaves in the tree. Using this idea we write a function that returns a trimmed tree:

```plaintext
TrimTree := proc (t:Tree, allall:matrix )
if type(t,Leaf) then t
else l := TrimTree( t[Left], allall );
    r := TrimTree( t[Right], allall );
    if type(l,Leaf) and type(r,Leaf) and
    allall[l[1],r[1]],identity > 0.8 then
        printf( '%s and %s have identity %.2f%%, %s ignored
',
            t[Label], r[Label], allall[l[1],r[1]],identity*100,
            r[Label] );
    else Tree(l,t[2],r);
fi
end:
```

With all these changes the tree is now:

```plaintext
DrawTree( TrimTree(tree,AllAll), Phylogram, Legend,
    OrderLeaves=LeftHeavy, LengthFormat=' ' );
```

solomongarba22 and bulawa2000 have identity 96.81%, bulawa2000 ignored osmansary and osmansaryl have identity 99.89%, osmansaryl ignored ericjons2 and ericjons01 have identity 97.90%, ericjons01 ignored ericjons2 and ericm001 have identity 97.90%, ericm001 ignored harenma and harenma have identity 92.50%, harenma ignored kambirelana and kambirelana have identity 99.95%, kambirelana ignored ekidion45 and ekidion45 have identity 99.92%, ekidion45 ignored omoek166 and ekidinn have identity 97.43%, ekidinn ignored ekidion45 and omoek166 have identity 95.67%, omoek166 ignored ekidion45 and gloriaeze2003 have identity 89.48%, gloriaeze2003 ignored farsan1 and farsan1 have identity 97.97%, farsan1 ignored stevemaloko and ohnsondoyle have identity 96.63%, ohnsondoyle ignored dukatka_s22 and mpeti_ka01 have identity 92.02%, mpeti_ka01 ignored luisabiTa75 and dukatka_s22 have identity 90.53%, dukatka_s22 ignored challenge_11 and challenge_secl have identity 97.72%, challenge_secl ignored wallmark and petersigho have identity 94.01%, petersigho ignored o_attah and drenakiga have identity 82.36%, drenakiga ignored hassan210k and mk_hassan002 have identity 98.32%, mk_hassan002 ignored ifekmark and markinjoa have identity 96.14%, markinjoa ignored johndisi1973 and bmakelele1 have identity 99.97%, bmakelele1 ignored . . . . (30 output lines skipped) . . .
mrs_mariumses13 and mrs_mariumses13 have identity 98.85%, mrs_mariumses13 ignored sesekom and mrs_mariumses13 have identity 95.41%, mrs_mariumses13 ignored ladawamobu3000 and sesekom have identity 94.44%, sesekom ignored sesekom and ladawamobu3000 have identity 88.02%, ladawamobu3000 ignored sebekom and sesekom have identity 93.31%, sesekom ignored simonmuzenda and simonmuzenda have identity 98.60%, simonmuzenda have identity 99.50%, simonmuzenda and muzendaasimon have identity 96.50%, muzendaasimon ignored dsmnd23 and danstevens101 have identity 85.45%, danstevens101 ignored rubutuwen99 and rubutuwen99 have identity 99.72%, rubutuwen99 ignored donaldste10 and jamesufon5050 have identity 96.79%, jamesufon5050 ignored emmastevens100 and donaldste10 have identity 99.02%, donaldste10 ignored emmastevens100 and emmastevens100 have identity 98.13%, emmastevens100 ignored rubutuwen99 and emmastevens100 have identity 92.87%, emmastevens100 ignored
franksalihu23 and rubutuwena99 have identity 95.29%, rubutuwena99 ignored
musa10smith and franksalihu23 have identity 94.01%, franksalihu23 ignored
mahabile50 and mahabile10 have identity 99.54%, mahabile10 ignored
sodindo2001 and sodindo2001 have identity 99.94%, sodindo2001 ignored
satu25 and satu25 have identity 96.35%, satu25 ignored
luckygarbi and simonmuzendda have identity 95.75%, simonmuzendda ignored

which shows some of the structure, but still virtually impossible to find the details.

Separating the sequences into clusters

Sequences which are completely unrelated to one another disrupt the phylogenetic trees. Sometimes this disruption is so great that it is difficult to recognize the true properties. This can very well happen in this case, where there may be more than a single source of messages. Like with genomic sequences, trying to associate completely different classes of objects destroys real relations. Once that we have computed the all-against-all alignments we can cluster the sequences using the Clusters function in Darwin. This function provides a distance parameter to make the clusters and hence the algorithm can be tuned to the data. In this case we know that we would like to separate the clusters around distance 40

clusters := Clusters( AllAll, AveDistance=40 );

These clusters are composed of truly related messages. We write a function to compute a tree for a given cluster. This function receives a set of Entry numbers (a cluster) and uses the AllAll matrix to select the desired alignments; no further computation of alignments is needed. The rest is identical to the previous tree construction.
ClusterTree := proc( c::set(posint) )
n2 := length(c);
allall := CreateArray(1..n2, 1..n2);
for i to n2 do for j from i+1 to n2 do
    allall[i,j] := allall[j,i] := AllAll[c[i],c[j]];
    od od;
Labels := [ seq( msgs[i,shortid], i=c )];
Seqs := [ seq( msgs[i,body], i=c )];
tree := PhylogeneticTree( Seqs, Labels, Distance, allall );
DrawTree( TrimTree(tree,allall), Unrooted, CrossReference );
end:

We now test the function over three largest clusters. To do this, we first sort the clusters by reverse size (so the three largest are the first three) and display the two largest.

clusters := sort( clusters, x -> -length(x) ):
clusters[1];
clusters[2];

(18, 42, 47, 67, 102, 146, 150, 154, 156, 183, 214, 215, 242, 261, 262, 270)

ClusterTree( clusters[1] );
ClusterTree( clusters[2] );

benwills20012002 and benwills11 have identity 96.84%, benwills11 ignored
francisdurul2 and mkll1km have identity 95.18%, mkll1km ignored

ClusterTree( clusters[3] );

malik3 and afiz have identity 99.85%, afiz ignored
aya and aya have identity 99.73%, aya ignored
pat27 and colzaq8 have identity 98.85%, colzaq8 ignored
The clusters show a pattern which is typical of sequences which are mutated again and
again and we have all the intermediate steps of their evolution (which is not the case in
biology). This is indicated by nodes which have branches so short that they lie against
the main branch. As we collect more information, it is likely that we will connect some
of the disjoint clusters.

The source data for this bio-recipe (as an electronic mailbox standard format) can be
found here.

© 2005 by Gaston Gonnet, Informatik, ETH Zurich

Please cite as:

@techreport{Gonnet-NigerianPrince,
  author = {Gaston M. Gonnet},
  title = {Phylogenetic tree of the "Nigerian Prince" email scam},
  month = {July},
  year = {2003},
  number = {415},
  howpublished = {Electronic publication},
  copyright = {code under the GNU General Public License},
  institution = {Informatik, ETH, Zurich},
  URL = {http://cburg.inf.ethz.ch/bio-recipes/NigerianPrince/code.html}
}

Index of bio-recipes

Last updated on Sun Apr 17 21:42:18 2005 by GhG