Paper 151-2008

Generalizing Google's PageRank to Rank National Football League Teams

Anjela Y. Govan* Carl D. Meyer† Russell Albright‡

ABSTRACT

The ranking of sports teams is of significant importance to those who are involved with or interested in the various professional and amateur leagues that exist around the world. We present a ranking algorithm that is simple to implement in SAS code and which gives results that are consistent with some of the best and most well-known computer methods for ranking sports teams. Whether you are trying to get an edge up on the office pool or to rank the teams for your daughter's soccer league tournament, the techniques presented here will allow you to accomplish this easily. Our method, inspired by the Google's PageRank algorithm, uses ideas from the PageRank algorithm to rank teams based on their schedule results. We include SAS/IML code implementation of each of the algorithms at the end of the relevant section.

INTRODUCTION

The ranking of teams, whether it is applied to teams in the NFL, the NCAA, or any other league for that matter, is a familiar occurrence to almost everyone. Polls and computer rankings are updated weekly for most major sports leagues. A *ranking* of n teams in a league is an assignment of an ordinal number (rank) to each team which allows us to order these teams according to some standard of quality. Typically the rank is based on some real-valued rating score that is computed from season statistics. The ratings are then sorted so that the ranks can be assigned to each team. The team with the largest rating is assigned a rank of 1, the team with the second largest rating is ranked 2, and so on. Obviously, establishing the rating score is the key component for generating this ranking.

In some cases the rating is derived from votes as is done when coaches or sport's writers are polled. In this case, individual voters assign their votes to the teams and the votes are tallied to obtain a rating. Typically voters will use a team's win-loss record, other statistics, and their own feeling or intuition to vote for the teams. Computer ranking algorithms cannot rely on feeling or intuition to rank teams but only on the concrete statistics derived from the season. Many computer ranking techniques exist for obtaining the rating and subsequent ranking of sport's teams (4; 6; 9; 13) among others.

In this paper we will compare two important ranking algorithms developed by W.N. Colley and J.P. Keener to the one we have devised, Generalized Markov chains Method (GeM). GeM is derived from Google's PageRank algorithm for computing the ratings for each indexed page on the Web.

The remainder of this paper will proceed as follows. In the next section we will discuss the challenges that computer ranking techniques must overcome in order to be effective. In Section we introduce the IML framework used to program the algorithms. Section cover the approaches by Colley and Keener. In Section we provide a brief background of the PageRank algorithm. Section introduces our GeM method by showing how the results of competition between athletic teams can be formulated in a way that generalizes the PageRank solution. All three algorithms are implemented and presented using SAS/IML. We conclude the paper with a discussion of our results.

THE COMPUTER RANKING CHALLENGE

Obtaining a ranking of teams in a league can be very difficult and controversial. It is sometimes problematic even to obtain a consensus on which single team should receive the highest rank, let alone a consensus on the ranking of all of the teams in a league. Simply considering the win-loss record of a team does not provide enough information to establish a ranking because each team does not compete against the same competition. In other words, the "strength of schedule" needs to be taken into account.

In the rankings we discuss in the paper, teams that win against stiff competition will receive a larger boost in the rankings than those that win against mediocre competition. And similarly, teams that lose against stiff competition will not be as adversely affected as teams that lose to poor competition.

‡Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (aygovan@math.ncsu.edu)
‡Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (meyer@math.ncsu.edu)
‡SAS Institute Inc., Cary , NC (Russell.Albright@sas.com)

Ranking teams with a computer algorithm also presents a problem if a team can deliberately manipulate its rank. When a ranking method uses a statistic, such as the margin of victory, strong teams can sometimes improve their rank by manipulating that statistic. In the 90's, for instance, it was not uncommon for teams to run up the score against a weak opponent in order to improve their rank.

THE SAS/IML FRAMEWORK

The algorithms presented in this paper are coded in SAS/IML. We use a small example based on a subset of ten games from the 2007 NFL regular season to help explain the algorithms. Prior to using the algorithms, the outcomes of the league games need to be gathered and stored. We use SAS data sets to store the results of the games. Since we need to transform the results into a matrix form prior to applying the algorithm, we assign indices to each team in the league. In the code below indexes 6 teams in our small example.

```
data indexTeam;
  Input Team $3. Index;
   datalines;
Car 1
Dal 2
Hou 3
NO  4
Phi 5
Was 6
;
run;
```

We store each game as an observation in the data set and variables indicate which teams played and the scores for each team.

```
data NFL2007EXAMPLE;
  Input Team_A_Index Score_A Team_B_Index Score_B;
    datalines;
1 16 4 13
2 38 5 17
2 28 6 23
3 34 1 21
3 23 4 10
4 31 1 6
5 33 6 25
5 38 4 23
6 27 2 6
6 20 5 12
;
run;
```

Now we can load the data into IML and call the respective algorithms. The code for the algorithms will be presented in the coming sections.

```
/*IML framework for Ranking Modules*/
proc iml;

/*placeholder for the Colley Module*/
/*placeholder for the Keener Module*/
/*placeholder for the GeM Module*/

use NFL2007EXAMPLE;
read all into A;
```

```
close NFL2007EXAMPLE;
dim=max(max(A[,1]),max(A[,3]));

Keenerating=keener(A,dim);  /*call function keener(data matrix, number of teams)*/
Colleyrating=colley(A,dim); /*call function colley(data matrix, number of teams)*/
GeMrating=GeM(A,0.85,dim); /*call function GeM(data matrix, number of teams)*/

create result07 var{GeMrating Colleyrating Keenerating};
append;
show contents;

quit;
```

ALTERNATIVE COMPUTER RANKING ALGORITHMS

COLLEY MATRIX METHOD

The Colley Matrix Method (4) is one of the clearest and most straightforward approaches to ranking teams. The only statistics used by this algorithm are the number of wins and losses, and the number of games played for each team, assuming no ties. This simplistic approach is very effective because it does not succumb to the bias introduced by some of the other statistics that more complex algorithms use and because it inherently accounts for the strength of schedule of each team.

The essence of the Colley method is to formulate and solve a system of linear equations to obtain the rating of each team. Each equation in the system corresponds to a team and the simultaneous solution to the system provides the rating score for each team.

The algorithm is based on an elegant result from probability called Laplace's Rule of Succession (14, p.108). The rule is used to approximate probabilities of boolean events (in our case, the probability of winning or losing a game). If we observe s successes out of n attempts, the rule states that $(s+1)/(n+2)$ is a better estimate for an upcoming event to be a success than simply calculating the winning percentage, s/n. The outline of the Colley algorithm is as follows:

1. Form Colley matrix **C**.
$$\mathbf{C}_{ij} = \begin{cases} -n_{j,i} & \text{if} \quad i \neq j \\ 2+n_i & \text{if} \quad i = j \end{cases}$$
where n_i is the total number of games played by team i and $n_{j,i}$ is the number of times team i played team j.
2. Form vector **b**.
$$b_i = 1 + (w_i - l_i)/2$$
where w_i is the number of wins by team i and l_i is the number of loses by team i.
3. Solve
$$\mathbf{C}r = \mathbf{b}$$
the vector **r** contains rating scores of each team. Use the scores to determine each team's rank.

For our small NFL example Colley matrix **C** and vector **b** are

$$\mathbf{C} = \begin{pmatrix} 5 & 0 & -1 & -2 & 0 & 0 \\ 0 & 5 & 0 & 0 & -1 & -2 \\ -1 & 0 & 4 & -1 & 0 & 0 \\ -2 & 0 & -1 & 6 & -1 & 0 \\ 0 & -1 & 0 & -1 & 6 & -2 \\ 0 & -2 & 0 & 0 & -2 & 6 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 1/2 \\ 3/2 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix}.$$

The IML module to calculate the ratings is presented below. The entries for the matrix **C** and vector **b** are calculated and the IML solve function is used to calculate the rating for each team. The module for calculating the Colley rating is presented below.

```
/*************************************************************
Module colley -computes rating scores of the teams
INPUT: A (games by 4) matrix [team i score i team j score j]
       teams - number of teams
OUTPUT: vector Cvec, where Cvec(i) is the rating score of team i
*************************************************************/
start colley(A,teams);               /* begin module colley*/
/* N[i,j] = number of times team i played team j*/
/* WL[i,1] = number of wins by team i, WL[i,2]= number of losses by team i*/
N=j(teams,teams,0);
WL=j(teams,2,0);

do i=1 to nrow(A);
    N[A[i,1],A[i,3]]=N[A[i,1],A[i,3]]+1;
    N[A[i,3],A[i,1]]=N[A[i,3],A[i,1]]+1;
    if A[i,2]>A[i,4] then
      do;
          WL[A[i,1], 1]=WL[A[i,1],1]+1;
          WL[A[i,3], 2]=WL[A[i,3],2]+1;
      end;
    else
      do;
          WL[A[i,1], 2]=WL[A[i,1],2]+1;
          WL[A[i,3], 1]=WL[A[i,3],1]+1;
      end;
end;

total = N[ ,+];
C=j(teams,teams,0); /* initialize Colley matrix to zero*/
B=j(teams,1,0);  /* initialize B vector to zero*/

/* Fill in Colley matrix and B vector */
do i=1 to teams;
   B[i,1]=1+(WL[i,1]-WL[i,2])/2;
   C[i,i]=2+total[i,1];
   do j=i+1 to teams;
      C[i,j]=-N[i,j]; C[j,i]=-N[i,j];
   end;
end;

Cvec = solve(C,B);
return(Cvec);
finish;                          /* end module colley*/
```

The resulting rating scores vector for our example is

$$\mathbf{r} \approx \begin{pmatrix} 0.3597 & 0.616 & 0.6687 & 0.3149 & 0.5015 & 0.5392 \end{pmatrix}^T$$

According to these rating scores the list of ranked teams (from first to last) is

Houston Dallas Washington Philadelphia Carolina New Orleans. .

KEENER RANKING METHOD

James P. Keener proposed his ranking method based on the theory of nonnegative matrices in 1993 (6). Specifically, Keener makes use of valuable properties of existence and uniqueness of the Perron vector guaranteed by Perron-Frobenius Theorem (10, p.673).

Keener's approach forms a smoothed score matrix. Keener makes use of Laplace's rule of succession. Unlike Colley, Keener uses game scores to compute ratings. This makes his algorithm susceptible to some manipulation since teams

can affect their ranking by running up scores. To minimize the possibility of manipulation Keener created a score "smoothing" function. Although he presents several possibilities for smoothing functions, we use Keener's function, h, defined below is an intuitive example and seems to work well.

Again the substance of this ranking algorithm is in solving a linear system, although now the result is a positive eigenvector of the Keener matrix. Keener's ranking algorithm is:

1. Form Keener matrix **K**.

$$\mathbf{K}_{ij} = \begin{cases} h\left(\dfrac{S_{ij}+1}{S_{ij}+S_{ji}+2}\right) & \text{if team } i \text{ played team } j \\ 0 & \text{otherwise} \end{cases}$$

   where S_{ij} is number of points scored by team i against team j and

$$h(x) = 1/2 + (1/2)\text{sgn}(x-1/2)\sqrt{|2x-1|}$$

2. Positive vector **r** contains each team's rating scores and is the Perron vector of matrix **K**, i.e. solve

$$\mathbf{K}r = \lambda \mathbf{r}$$

   where λ is the spectral radius (dominant eigenvalue) of **K**. Use rating scores in **r** to rank teams.

Some teams in our small NFL example play each other more than once. In this case we add the corresponding game scores for each of the games between the same two teams to produce one cumulative score. For example, Dallas (Dal) and Washington (Was) played twice where Dallas won 28-23 and Washington won 27-6. The cumulative score between Dallas and Washington is therefore Washington won 60-34. This is the simplest approach for dealing with multiple games between two teams. The Keener matrix for our small example is

$$\mathbf{K} \approx \begin{pmatrix} 0 & 0 & 0.2612 & 0.2156 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8035 & 0.2843 \\ 0.7388 & 0 & 0 & 0.8047 & 0 & 0 \\ 0.7844 & 0 & 0.1953 & 0 & 0.256 & 0 \\ 0 & 0.1965 & 0 & 0.744 & 0 & 0.5 \\ 0 & 0.7157 & 0 & 0 & 0.5 & 0 \end{pmatrix}.$$

The module for the IML implementation is given below

```
/*************************************************************
Module keener -computes rating scores of the teams
INPUT: A (games by 4) matrix [team i score i team j score j]
       teams - number of teams
OUTPUT: vector Cvec, where Kvec(i) is the rating score of team i
*************************************************************/
start keener(A,teams);          /* begin module keener*/
Scores=j(teams,teams,0);

/* Form matrix Scores, s.t. Scores(i,j)=score of team i against team j*/
do i=1 to nrow(A);
    Scores[A[i,1], A[i,3]]=Scores[A[i,1], A[i,3]]+A[i,2];
Scores[A[i,3], A[i,1]]= Scores[A[i,3], A[i,1]]+A[i,4];
end;

K=j(teams,teams,0); /* initialize Keener matrix to zero*/

/* Populate K with smoothed out score ratios*/
do i=1 to teams by 1;
   do j=1 to teams by 1;
   scoreij=Scores[i,j];
   if (abs(scoreij-0))>0.00000001 then
      do;
```

```
      t=2*(Scores[i,j]+1)/(Scores[i,j]+Scores[j,i]+2)-1;
      if t>0 then
         K[i,j]=0.5+0.5*sqrt(t);
      else
         K[i,j]=0.5-0.5*sqrt(abs(t));
      end;
   end;
end;
call eigen(E,V,K);

/* Determine the location of the largest positive eigenvalue */
maxi=1;
do i=2 to teams by 1;
   if E[maxi,1]<E[i,1] then maxi=i;
end;
Kvec = V[ ,maxi];
Kvec=abs(Kvec);
return(Kvec);
finish;                          /* end module keener*/
```

The corresponding ratings vector is

$$\mathbf{r} \approx \begin{pmatrix} 0.0474 & 0.2385 & 0.1107 & 0.1079 & 0.2342 & 0.2614 \end{pmatrix}^T .$$

Sorting our six teams according to their rating scores we obtain the following ranking from first to last

Washington Dallas Philadelphia Houson New Orleans Carolina .

PAGERANK RANKING METHOD

PageRank is a ranking algorithm developed by the founders of Google, Larry Page and Sergey Brin (12), in 1998. It is designed as a part of the search engine Google. This algorithm uses basic properties of Markov chains (10, p.687) to compute the rating scores of each web page prior to the user's query. Markov chain theory is developed for the stochastic (nonnegative matrices with each row summing to 1) matrices, and is a special case of the Perron-Frobenius Theorem.

The method forms an adjacency matrix out of web page links, tweaks this matrix to guarantee that every web page can be visited from any other web page by adjusting the dangling nodes (nodes with no outlinks) and adding a special rank 1 matrix. Final step is to calculate the left eigenvector associated with the eigenvalue 1 of the resulting matrix. The outline of the PageRank algorithm is as follows:

1a. Represent web with n web page as a directed graph, with n nodes.

1b. Form an adjacency matrix $\mathbf{A}$.

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if there is a link from web page } i \text{ to web page } j \\ 0 & \text{otherwise} \end{cases}$$

2. Form hyperlink matrix $\mathbf{H}$.

$$\mathbf{H}_{ij} = \begin{cases} 1/\sum_{k=1}^{n} \mathbf{A}_{ik} & \text{if there is a link from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

3. Form Google matrix $\mathbf{G}$.

$$\mathbf{G} = \alpha[\mathbf{H} + (1/n)\mathbf{a}\mathbf{e}^T] + (1-\alpha)\mathbf{e}\mathbf{v}^T$$

where $\mathbf{a}$ is the vertical vector such that $\mathbf{a}_i$ is 1 if row i of $\mathbf{H}$ is zero and $\mathbf{a}_i$ is 0 otherwise, $\mathbf{e}$ is a vector of all 1's, $0 < \alpha < 1$, and $\mathbf{v} > 0$ is a probability distribution vector ($\mathbf{v} \geq 0$ and sums to 1).

4. The rating vector $\boldsymbol{\pi}$ is a positive left eigenvector of $\mathbf{G}$ corresponding to eigenvalue 1, i.e. solve

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T \mathbf{G}$$

Use rating scores in $\boldsymbol{\pi}$ to rank web pages.

Note that the vector **v**, appearing in the step 3 of the algorithm, is referred to as a personalization vector. A basic personalization vector is $(1/n)\mathbf{e}$. More sophisticated personalization vector choices and their significance to the PageRank is discussed at length by Langville and Meyer in (7). The value of α effects the convergence of the power method, which is a simple algorithm for computing eigenvalues of a matrix. It is reported that Google originally used $\alpha = 0.85$ (12).

GeM RANKING METHOD

In this section we introduce a new type of ranking method, called GeM. This ranking method is motivated by the PageRank algorithm. While we apply the technique to ranking teams, it can be used on any situation which could be represented as a weighted directed graph.

The GeM method represents the sports season as a weighted, directed graph and adapts the matrix in much the same fashion as the PageRank algorithm. Following is the outline of the GeM algorithm:

1a. Represent a sport season as a weighted directed graph with n nodes ($n = 32$ in case of NFL). Each team represents a node and each game a directed edge from loser to winner with weight equal to the positive difference of the game scores.

1b. Form an adjacency matrix **A**.

$$\mathbf{A}_{ij} = \begin{cases} w_{ij} & \text{if team } i \text{ lost to team } j \\ 0 & \text{otherwise} \end{cases}$$

where w_{ij} is the positive score difference of the game team i lost to j. In case team i lost to team j more than once during a given season w_{ij} is the sum of the positive score difference of the games team i lost to j.

2. Form matrix **H**.

$$\mathbf{H}_{ij} = \begin{cases} 1/\sum_{k=1}^{n} \mathbf{A}_{ik} & \text{if there is a link from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

3. Form GeM matrix **G**.

   (a) Basic version

   $$\mathbf{G} = \alpha[\mathbf{H} + \mathbf{a}\mathbf{u}^T] + (1-\alpha)\mathbf{e}\mathbf{v}^T$$

   where $0 < \alpha < 1$, **v** is a positive probability distribution vector and **a** is the vertical vector such that $\mathbf{a}_i$ is 1 if row i of **H** is zero and $\mathbf{a}_i$ is 0 otherwise. We will allow vector **u** to be any $n \times 1$ probability distribution vector (our adjustment for undefeated teams).

   (b) Generalized version

   $$\mathbf{G} = \alpha_0\mathbf{S}_0 + \alpha_1\mathbf{S}_1 + ... + \alpha_p\mathbf{S}_p$$

   where $\mathbf{S}_i = [\mathbf{H}_i + \mathbf{a}_i\mathbf{u}_i^T]$ is a stochastic matrix $0 \le i \le p$, we will call it an ith feature matrix corresponding to the ith statistic, and $0 \le \alpha \le 1$, $\sum_{i=0}^{n} \alpha_i = 1$.

4. The vector containing the rating scores of each team is $\boldsymbol{\pi}$ such that

   $$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{G}$$

   Use rating scores in $\boldsymbol{\pi}$ to rank teams.

NFL WEIGHTED DIRECTED GRAPH

We form the NFL weighted directed graph by representing each team as a node. Each game between two teams, i and j, results in a directed edge from loser to winner, e.g. if i lost to j then there is an edge (i, j), with weight w_{ij} being the positive score difference. PageRank allows for weight 1 on each of the directed edges. Allowing any positive weights on the edges of the directed graph is our first major generalization of PageRank. The change is illustrated in Figure .

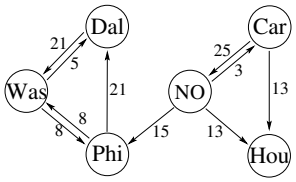

Figure 1 Small NFL example from the 2007 regular season.

ADJACENCY MATRIX A

The small weighted NFL digraph has a corresponding adjacency matrix

$$\mathbf{A} = \begin{array}{c} \\ \text{Car} \\ \text{Dal} \\ \text{Hou} \\ \text{NO} \\ \text{Phi} \\ \text{Was} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Car} & \text{Dal} & \text{Hou} & \text{NO} & \text{Phi} & \text{Was} \end{array} \\ \begin{pmatrix} 0 & 0 & 13 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 21 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 13 & 0 & 15 & 0 \\ 0 & 21 & 0 & 0 & 0 & 8 \\ 0 & 5 & 0 & 0 & 8 & 0 \end{pmatrix} \end{array}.$$

MATRIX H

Matrix **H** is formed by scaling each nonzero row of matrix **A** by the sum of its entries. In our example

$$\mathbf{H} = \begin{array}{c} \\ \text{Car} \\ \text{Dal} \\ \text{Hou} \\ \text{NO} \\ \text{Phi} \\ \text{Was} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Car} & \text{Dal} & \text{Hou} & \text{NO} & \text{Phi} & \text{Was} \end{array} \\ \begin{pmatrix} 0 & 0 & 13/38 & 25/38 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3/31 & 0 & 13/31 & 0 & 15/31 & 0 \\ 0 & 21/29 & 0 & 0 & 0 & 8/29 \\ 0 & 5/13 & 0 & 0 & 8/13 & 0 \end{pmatrix} \end{array}$$

is the resulting matrix.

GeM MATRIX G

Before we form matrix **G** we need to address the issue of the dangling nodes. The dangling nodes in the context of sports correspond to undefeated teams and show up as zero rows in **H**. The matrix that will produce the ratings vector has to be irreducible (corresponds to strongly connected digraph) and stochastic (rows add to 1). We transform matrix **H** into a stochastic matrix by making a rank 1 update, $\mathbf{H} + \mathbf{a}\mathbf{u}^T$. Since nonzero entries in the vector **a** correspond to the zero row in matrix **H** this update replaces zero rows of **H** with a probability distribution vector. Brin and Page equated the ith row entries of **H** with portions of a "vote" (via hyperlinks) of web page i for other web pages. In the context of competing sports teams, this "vote" is cast by the losing team for the winning team. The entries in the ith row of **H** can also be interpreted as the probabilities that team i will lose to the other teams. For the undefeated team the adjustment we propose can be thought of as changing the probability of an undefeated team to lose to others (including itself) to be nonzero. A simple choice is to set $\mathbf{u} = (1/n)\mathbf{e}$, the uniform probability distribution.

In the case of the small NFL example the only undefeated team present is Houston, hence $\mathbf{a} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T$ Setting $\mathbf{u} = (1/6)\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}^T$

$$\mathbf{H} + \mathbf{a}\mathbf{u}^T = \begin{array}{c} \\ \text{Car} \\ \text{Dal} \\ \text{Hou} \\ \text{NO} \\ \text{Phi} \\ \text{Was} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Car} & \text{Dal} & \text{Hou} & \text{NO} & \text{Phi} & \text{Was} \end{array} \\ \begin{pmatrix} 0 & 0 & 13/38 & 25/38 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 3/31 & 0 & 13/31 & 0 & 15/31 & 0 \\ 0 & 21/29 & 0 & 0 & 0 & 8/29 \\ 0 & 5/13 & 0 & 0 & 8/13 & 0 \end{pmatrix} \end{array}.$$

BASIC VERSION OF THE GeM MODEL

The basic model uses only one parameter α just as the original PageRank. The GeM matrix is $\mathbf{G} = \alpha(\mathbf{H} + \mathbf{au}^T) + (1-\alpha)\mathbf{ev}^T$ where $0 < \alpha < 1$, and $\mathbf{v} > 0$ is a probability distribution vector, we can continue to call it personalization vector. Again a simple personalization vector is $\mathbf{v} = (1/n)\mathbf{e}$. The personalization vector allows for great flexibility of our algorithm. We could form $\mathbf{v}$ using any or all statistical data from the games already played. If we set $\alpha = 0.85$ and use a personalization vector $\mathbf{v} = (1/6)\mathbf{e}$ then the matrix $\mathbf{G}$ for our small example is

$$\mathbf{G} = 0.85(\mathbf{H} + \mathbf{au}^T) + 0.15(1/6)\mathbf{ee}^T =$$

	Car	Dal	Hou	NO	Phi	Was
Car	1/40	1/40	6/19	111/190	1/40	1/40
Dal	1/40	1/40	1/40	1/40	1/40	7/8
Hou	1/6	1/6	1/6	1/6	1/6	1/6
NO	133/1240	1/40	473/1240	1/40	541/1240	1/40
Phi	1/40	743/1160	1/40	1/40	1/40	301/1160
Was	1/40	183/520	1/40	1/40	57/104	1/40

The SAS/IML code for computing GeM is listed below.

```
/*****************************************************************
Module GeM -computes rating scores of the teams
INPUT: A (games by 4) matrix [team i score i team j score j]
       alpha - contribution of scores to the ratings
       teams - number of teams
OUTPUT: vector Gvec, Gvec(i)=rating score of team i
*****************************************************************/
start GeM(A,alpha,teams);               /* begin module GeM*/
M=j(teams,teams,0);
do i=1 to nrow(A);
    if A[i,2]>A[i,4] then M[A[i,3], A[i,1]]=A[i,2]-A[i,4];
    else M[A[i,1], A[i,3]]=A[i,4]-A[i,2];
end;
rowsuma = M[ ,+];

/* Initialize the adjusting matrices a and b; matrix a will rescale
the nonzero rows of M, and matrix b will replace the zero rows.*/
ae=rowsuma[1];
   if ae^=0 then
      do; ae=1/ae; a=j(1,teams,ae); b=j(1,teams,0); end;
   else
      do; a=j(1,teams,1); b=j(1,teams,1/teams); end;
/* Finish constructing the adjustment matrices.*/
x=1;
do while(x<teams);
   x=x+1; ae=rowsuma[x];
   if ae^=0 then
      do;
         ae=1/ae; atemp=j(1,teams,ae); a=a//atemp;
         btemp=j(1,teams,0); b=b//btemp;
      end;
   else
      do;
         atemp=j(1,teams,1); a=a//atemp;
         btemp=j(1,teams,1/teams); b=b//btemp;
      end;
end;
/* Stochastic matrix S is formed as S=M#A, element-wise multiplication
of the given matrix. This makes non zero rows add to 1. Then add matrix
b to S to replace the zero rows.*/
```

```
S=M#a;
S=S+b;
/* Form matrix G */
G=alpha*S+(1-alpha)*j(teams,1,1/teams)*j(1,teams,1);
call eigen(E,V,t(G));

/*Extract the eigenvector (column of V) corresponding to the
eigenvalue 1 (1+0i) in the matrix E. */
t=0;
do i=1 to teams;
        if abs(E[i,1]-1)<0.0000001 then t=i;
end;
Gvec = V[ ,t];
Gvec=abs(Gvec); Gvec=(1/sum(Gvec))*Gvec;
return(Gvec);
finish;                              /* end module GeM*/
```

When the code is run on the NFL example the results are

$$\boldsymbol{\pi}^T \approx \begin{pmatrix} 0.038 & 0.2825 & 0.0656 & 0.056 & 0.2289 & 0.3281 \end{pmatrix},$$

and the ordered ranking of teams (from first to last) is

Washington Dallas Philadelphia Houston New Orleans Carolina .

GENERALIZED VERSION OF THE GeM MODEL

On the last note, let us mention the most general version of the GeM algorithm.

$$\mathbf{G} = \alpha_0 \mathbf{S}_0 + \alpha_1 \mathbf{S}_1 + ... + \alpha_p \mathbf{S}_p$$

We form each stochastic matrices $\mathbf{S}_i$ using team's ith statistic (e.g. game scores, total yards, passing yards, etc.). This is done by setting up a weighted directed graph of the teams and games (winner of each is the team with largest value of the statistic we are using) as before except the weights will be the positive difference between the chosen statistic. Then each feature matrix is $\mathbf{S}_i = \mathbf{H}_i + \mathbf{a}_i \mathbf{u}_i^T$ for $0 \leq i \leq p$.

Additional issues arise from generalization of the PageRank algorithm. We have to make sure that stochastic matrices $\mathbf{S}_i$ carry enough information so that their linear combination is at the end irreducible. What set of statistical data should we include in our ranking algorithm? Should it be all we can get our hands on or there is a set of statistics which are the best predictors of the ranks? Another question is what should values of each α_i should be. The only restriction on these parameters is that $0 \leq \alpha_i \leq 1$ for each i and $\sum_{i=0}^{k} \alpha_i = 1$. We can interpret value of α_i as the amount (or percentage) ith statistic contributes to the overall rating score of each team. These issues are a part of our ongoing research.

CONCLUSION

As with many data mining techniques there is no perfect algorithm for ranking sports teams. Different ranking algorithms have their strengths and weaknesses. The Colley algorithm is bias-free, but rigid and given the win-loss and game numbers it computes the rating scores without a regard to any other statistics. Keener's algorithm allows more information to be considered to differentiate between the teams, but this makes it susceptible to manipulation. Keener's algorithm is also more or less rigid, "smoothed" game scores determine the matrix and hence the rating scores. GeM exhibits an abundant flexibility but that creates its own set of interesting problems. We can include any number of statistical data in our matrix **G**, but what data is the best for predicting the quality of a team? If we do decide on the statistic set how much should each contribute to the final rating scores?

Regardless of methods, SAS/IML provides an easy and productive way to code out and test these various algorithms. The code presented should be beneficial in ranking teams from any number of amateur or professional leagues that you may be interested in.

REFERENCES

[1] A. Arasu, J. Novak, and J. Tomkins, A. and Tomlin, *PageRank computation and the structure of the web: Experiments and algorithms*, in Proc. Eleventh International World Wide Web Conference (WWW2002), ACM Press, 2002.

[2] C. Brezinski and M. Redivo-Zaglia, *The PageRank vector: Properties, computation, approximation and acceleration*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 551-575.

[3] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen, *Efficient PageRank approximation via graph aggregation*, in Proc. Thirteenth International World Wide Web Conference (WWW2004), ACM Press, 2004, pp. 484-485.

[4] W. N. Colley, *Colley's Bias Free College Football Ranking Method.*, 2002.

[5] A. Gulli and A. Signorini, *The indexable web is more than 11.5 billion pages*, in Proc. Fourteenth International World Wide Web Conference (WWW2005), ACM Press, 2005, pp. 902-903.

[6] James P. Keener, *The Perron-Frobenius Theorem and the Ranking of Football Teams*, SIAM Review, Vol. 35, No. 1 (Mar.,1993), pp.80-93.

[7] Amy N. Langville, Carl D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings.*, Princeton University Press, 2006.

[8] Amy N. Langville, Carl D. Meyer, *A reordering for the PageRank problem*, SIAM J. Sci. Comput., 27 (2006), pp. 2112-2120.

[9] Kenneth Massey, *Statistical Models Applied to the Rating of Sports Teams*, Bluefield College, 1997.

[10] Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2005.

[11] Cleve Moler *The world's largest matrix computation*, Matlab News and Notes, pp.12-13, 2002.

[12] Sergey Brin, Lawrence Page, *The anatomy of a large-scale hypertextual Web search engine*, Computer Networks and ISDN Systems, 33: 107-17, 1998.

[13] Charles Redmond, *A Natural Generalization of the Win-Loss Rating System*, Mathematical Magazine, Vol. 76, No. 2, (Apr., 2003) p.119-126.

[14] Sheldon Ross, *A First Course in Probability*, Pearson Prentice Hall, Upper Saddle River, NJ, 2006.