Coach Ranking Model

Akilesh Potti, Dominick Twitty, Wenhai Yang

February 8, 2014

Abstract

Todo

Disclaimer: For legal and moral reasons, all geographical locations and company names used in this paper are entirely fictional unless otherwise stated. However, for the sake of accuracy, real world data was used to tune our model.

1 The Problem

1. Todo

2 Model assumptions

1. Todo

3 Models

3.1 Simple Heuristic Model

3.1.1 Sorting by Win/Loss Ratio

3.1.2 Sorting by Net Wins

3.1.3 Differentiate between Games

3.2 Machine Learning Model

3.3 Graphical Model

It comes very natural for us to consider the problem as a graph. The nodes of the graph are the coaches. An edge between two nodes denotes the relation between two coaches, such as the game result of them playing against each other.

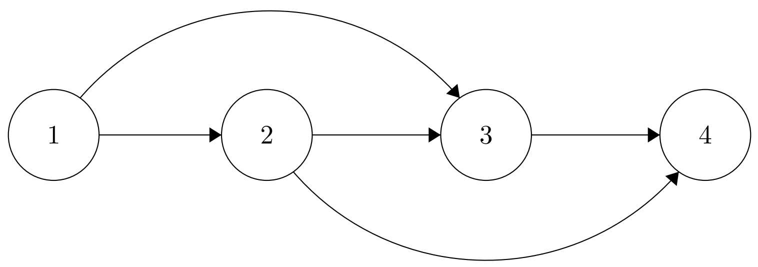
This idea of representing the problem as a graph is a very good step forward because it allow us to go beyond the simple statistics like win/loss ratio of a single coach, and gain more insight from the data.

Another advantage of this model is that it can aggregate data from different time periods. For example if coach A beats coach B and then retired in 1990, and coach B beat coach C later in 1995, then in the graph we will have a way to compare coach A and coach C because there is a path from A to C through B, even though they never played against each other.

3.3.1 Topolical Ordering

A very first idea that comes to mind is Topological Ordering. In this sub-model, there exist an edge from i to j if and only if among all the games i played against j, j beat i more often than i beat j (net-loss). Then if there exists a Topological Ordering of the graph, we have a ranking of the coaches based on their rival history.

Consider the following example. Coach 1 has been beaten by both 2 and 3, coach 2 has been beaten by coach 3 and 4, coach 3 has been beaten by coach 4, and coach 4 is never beaten by anyone. We have a topological ordering in the graph: $\{1, 2, 3, 4\}$, and it is natural to conclude that coach 4 is the best coach among them. (For now, when we talk A is beaten by B, we mean net-loss, which means A is beaten by B more often than B is beaten by A. We will introduce more sophisticated measures later).



However, one natural limitation of this model is the following theorem:

Theorem 3.1 *A graph G has a topological ordering if and only if it is a DAG (directed acyclic graph).*

Proof See Section 3.6 of Algorithms Design by Kleinberg and Tardos.

To solve this issue, we think of two algorithms. One is to reduce the originial graph into a DAG: keep removing the most "unimportant" edges untils the graph is acyclic. By most "unimportant" we then have to find a way to measure the importance of an edge. In this model, the weight (or importance) an edge by the net-loss it is associated with:

$$w(e_{i,j}) = \text{number of times j beats i} - \text{number of times j beats i}$$

The algorithm then works as following:

```
while $G$ is not DAG do
    $e_{min} = \underset{e \in E}{\operatorname{argmin}}\, w(e)$;
    Remove $e_min$ from $G$
end
```

However, this algorithm has apparent flaws: suppose even though coach A and coach B only encounter each other once, however it is in the SuperBowl, while the matchup between coach A and coach C is quite often but only in friendly games or early-season games and therefore the result is less important and less indicative of their comparative abilities. Therefore using the net-loss as the only measurement of the importance of an edge can possibly remove the important edge and keep the unimportant ones. We can of course tweak the edge-weight by introducing the importance of a game, the time in both coaches' career when the match happens, etc., but simply adding heuristics seem not like the best approach to the problem.

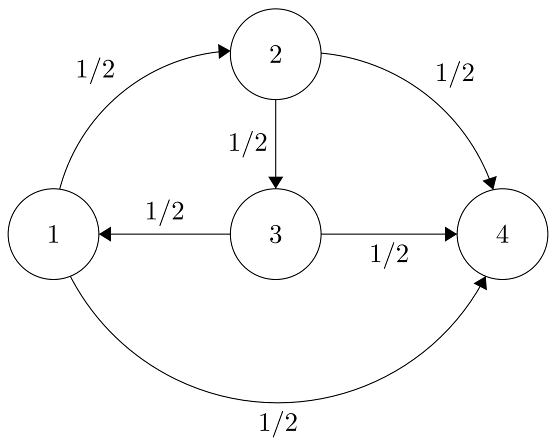
The second algorithm requires a certain "leap of faith" by introducing randomness into our model:

3.3.2 Markov Chain (Voting Model)

To solve the problem of Topogical Ordering, instead of trying to create a DAG to fit the algorithm, we decide adjust the mechanism behind ranking using Topological Ordering.

Imagine instead of having us, the spectators doing the voting and the ordering, each coach will have a **vote** of a certain **size**, which she can divide up into pieces. She can then give the chunks of the vote to different coaches which she think is definitely better coach than she is. We assume the coaches to be rational: she doesn't take into account the opinions of amateurs and public media, and only give out votes to the one she truly believe is better than her (one possible way is only give out vote to the coaches who has beaten her). Then we will rank the coaches based on how many votes each coach eventually get. However since the coaches form a graph and there even might be cycles in the game, so at each timestep a coach can be both receiving and giving. What we are interested in is the distribution of the votes at some large time.

Consider the following example. There are four coaches: 1, 2, 3, 4. Coach 1 has been beaten by 1 and 4, coach 2 has been beaten by 3 and 4, coach 3 has been beaten by 4 and 1, coach 4 has never been beaten.



The Markov Chain above shows how the vote will be given out at each timestep: coach 1 will give out half of her vote to coach 2 and the other half to 3, coach 2 will give out half of her vote to coach 3 and the other half to coach 4, coach 3 will give out half of her vote to coach 1 and the other half to coach 4. Coach 4, since she never lost to anyone, will simply keep her vote to herself. We can therefore see this can be modeled as a Markov Chain, and the distribution of votes at some large time is the stationary probability distribution of this Markov Chain:

$$\pi = \phi_0 P^n \text{ for some initial distribution } \phi_0$$

where $\mathbf{P}$ is the probability transition matrix, with $\mathbf{P_{i,j}}$ equal to the proporption of her chunk coach i will give to j (can also be interpreted as the probability of going from i to j in the Markov Chain):

$$\mathbf{P} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We know that if the Markov Chain is irreducable and aperiodic, then such a stationary distribution exists, and is unique, and it is the left eigenvector of $\mathbf{P}$ for the eigenvalue 1.

Irreducable & Aperiodic

Apparently, the example we show doesn't satisfy the property. To make sure the matrix we encounter will be irreducable and aperiodic, we use a similar technique as the PageRank algorithm used by Google:

1. For dead-ends (coach 4 in the example), in each step she will split her vote into equal pieces and give one piece to every coach, (go to a random state in the Markov Chain)
2. For non-deadends (coach 1, 2, 3 in the example), in each step with he will first split her vote into two, with proportion α and $1 - \alpha$. With the α proportion she did what she did before, spliting it and give the pieces to coaches who have beaten her, the remaining $1 - \alpha$ she will split into equal pieces and give one piece to every coach.

Therefore, after adjusting for deadends in $\mathbf{P}$, the new probability transition matrix can be represented as:

$$\mathbf{D} = \alpha\mathbf{P} + (1 - \alpha)\mathbf{U}$$

where $\mathbf{U}$ with all entries equal to $\frac{1}{n}$, n being the total number of coaches (or number of states in the Markov Chain).

With the new probability transition matrix, since all entries in $\mathbf{D}$ are positive, we know by Perron-Frobenius theorem that $\mathbf{D}$ is both irreducable and aperiodic, and that the stationary distribution π exists and is unique. With π we can find the coaches with the top 5 probability and they will be the best coaches "of all time" since they get the most proportion of votes from all coaches in the long-run.

Formal Representation

$$G(V, E) \text{ represents the model}$$

$$V = \{v_1, v_2 ... v_n\} \text{ where } v_i \text{ is coach i}$$

$$(v_i, v_j) \in E: \text{ coach j beats coach i more often than coach i beats coach j}$$

$$\mathbf{D}(i,j) = \begin{cases} \alpha \frac{1}{d(v_i)} + (1-\alpha)\frac{1}{n}, & \text{if } d(v_i) \geq 1 \\ \frac{1}{n}, & \text{otherwise} \end{cases}$$

Efficient Implementation

Since the matrix $\mathbf{D}$ is huge and all entries are non zero, calculating its eigenvalues and eigenvectors directly can be very inefficient computationally. However $\mathbf{P}$ is a sparse matrix and most of its entries are zero. To make sure we will be able to produce the eigenvector that represents the stationary distribution efficiently, we use Power Method:

```
$v = (1/n, ..., 1/n)$;
while $\|\alpha v\mathbf{P} + (1-\alpha)(1,...,1) - v\| < e^{-8}$ do
    $v = \alpha v\mathbf{P} + (1-\alpha)(1,...,1)$;
end
return $v$;
```

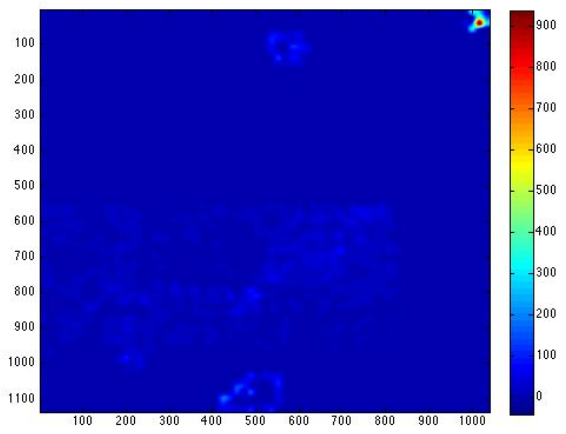
Since $v\mathbf{P}$ can be calculated using sparse matrix $\mathbf{P}$, this algorithm is much more computationally efficient than simply calculating the eigenvectors of $\mathbf{D}$. As α increases, the time it takes for v to converge is shorter, but the difference in the ranking might be less significant, therefore this is a tradeoff and we pick $\alpha = 0.85$.

More on Edge Weights

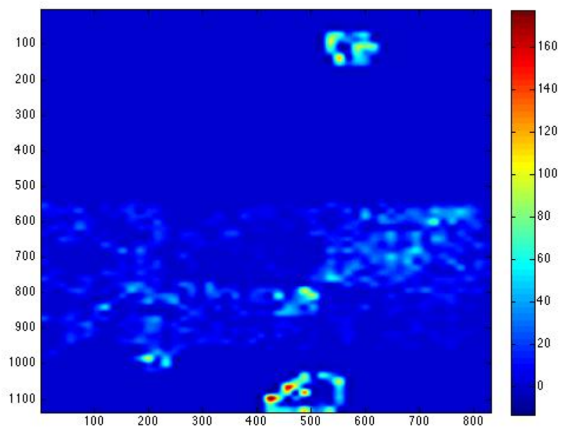
In the model we introduced above.

4 Results, Validation, and Robustness

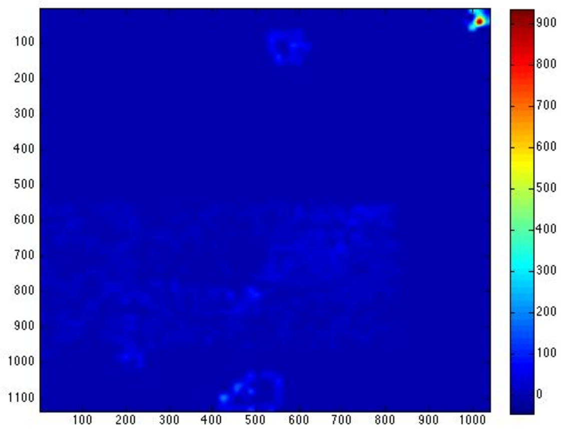
(a) Distribution of Request Source with Airport



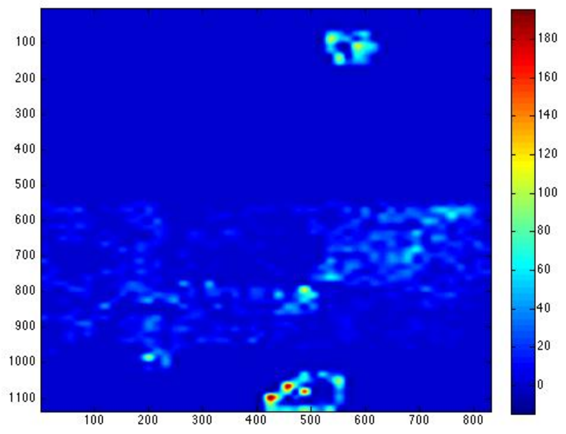
(b) Distribution of Request Source without Airport



(c) Distribution of Request Destination with Airport



(d) Distribution of Request Destination without Airport



$$p_{unreg} = k \ln(t_{reg})$$

$$\frac{p_{unreg'}}{p_{unreg}} = \frac{\ln_{t_{reg}}}{\ln t_{reg'}}$$

Number of Cabs	14	15	16	17	18	19	20	21
Uni Ind/Conglo	1.5720	1.3773	1.2880	1.2071	1.1932	1.2113	1.1201	1.0672
Prop Ind/Conglo	1.6876	1.4040	1.2459	1.1248	1.2558	1.1498	1.1033	1.1022
Disprop Ind/Conglo	2.1519	2.0706	2.0489	1.9399	1.8725	1.8174	1.6990	1.3905

5 Strengths and Weaknesses

5.1 Strengths

- Todo

5.2 Weaknesses

- Todo

6 Conclusions

Todo

7 Future work

- Todo

8 Bibliography

References

[1] OpenStreetMaps API, `http://www.openstreetmap.org/#map=14/42.4427/-76.4984`

[2] Github Gist by user aflaxman, `https://gist.github.com/aflaxman/287370/`

[3] A stochastic optimization model for real-time ambulance redeployment, `http://www.sciencedirect.com/science/article/pii/S0305054813000385`

[4] Ithaca Demographics, `http://www.ci.ithaca.ny.us/maps/index.cfm`

[5] Cornell Demographics, `http://www.cornell.edu/about/facts/stats.cfm`

[6] Ithaca College, `http://www.ithaca.edu/admission/facts/`

9 Code