Testing the Efficiency of the Elo ranking algorithm and Reddits Hot Rank algorithm

Priscilla Coronado
pcoron15@earlham.edu
Earlham College
Richmond, Indiana

ABSTRACT

This research looks at the efficiency of two ranking algorithms, the Elo algorithm and Hot Rank algorithm. With a database of a thousand recipe inputs, each recipe will receive a random number of points, or "upvotes," for the two algorithms to create their quality numbers in order to sort the list.This simulates a real user base. Both are ranking algorithms, but have different equations to determine which input ranks higher than the other. Therefore, the final ordering may not be identical. Each algorithm will be timed as they sort all of the one thousand inputs by comparing two inputs at a time. The experiment shows that the Elo algorithm is substantially faster than the Hot Rank algorithm.

CCS CONCEPTS

• **Algorithms Reddit; Hot Rank algorithm; Elo algorithm; Databases**;

KEYWORDS

Datasets, Databases, Algorithms, Reddit Hot Rank, Elo algorithm

ACM Reference Format:
Priscilla Coronado. 2018. Testing the Efficiency of the Elo ranking algorithm and Reddits Hot Rank algorithm. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Algorithms are the foundation of any program; they are the step-by-step process for developers to create the programs that modern society uses. Efficiency is key in building an algorithm, especially when it will be used by millions. People of today expect fast results and an application that takes too much time doing its processes will deter potential consumers.

Hot Rank is one algorithm that fall within the expectation of delivering fast results. It is an algorithm that is used by the 18th most popular website, Reddit. Reddit, being a forum based website, requires the Hot Rank algorithm to determine which user made post is more popular than other posts. The algorithm determines this by taking into account when the post was posted, what is the current date, the number of upvotes, and the number of downvotes of the post. These ideas are discussed further in section 2.2.

The Elo algorithm is an algorithm that normally does not fall into the expectation of delivering fast results. Originally it was created as a means to rank different chess players based on their wins and losses. This is discussed further in section 2.1.

For the purpose of this experiment, the two algorithms have been modified in order to sort a list and to simulate how the algorithms function in a realistic state. In other words, the algorithms are implemented in a way in order to simulate a real user base and how the user inputs generate the quality numbers that the algorithms produce.This is to observe and answer the main problem this experiment addresses: Which of the two modified sorting algorithms is able to efficiently sort an unsorted list of ids and their associated number of points through pairwise comparisons. In other words, the purpose of this experiment is to observe the modified algorithms sort the recipe ids by comparing the ids in pairs. This differs from regular sorting algorithms because the quality numbers that simulate user preference and are used by this experiments Elo and Hot Rank algorithm are not typically observable. The numbers generated within this experiment are created in order to simulate those quality numbers in order for the algorithms to sort them through pairwise comparisons.

From this experiment it has been concluded that the Elo and the Hot Rank implementations of having an exponential time complexity of $O(n^2)$ and $O(n^2(logn))$ respectively, the Elo algorithm is able to sort quicker than the Hot Rank algorithm. Therefore this experiments implementation of the Elo algorithm is better suited for sorting recipe id's within a list. This is discussed further in the section 6. From these results, this experiment has contributed to the question of which of these algorithms are better suited to make pairwise comparisons.

2 BACKGROUND

As noted in the introduction, the purpose of this research is to see if either the Elo algorithm or the Hot Rank algorithm is better suited for sorting recipes id's within a list. To conduct a preliminary evaluation of the algorithms, we have generated synthetic time and quality values. The purpose of this section is to describe how the two original systems function.

2.1 Elo Algorithm

Arpad Elo, the creator of the Elo rating system, originally designed the system for competitive chess, where there are two players. Within Elo's rating system, a players rank represents their ability within the game. This rank is dependent on how many wins and losses the player has. When two chess players compete against

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

each other, the algorithm predicts the outcome of the match based on its probability formula. For example, the formula used in this implementation is:

$P = 1/1 + 10^{R_1 - R_2/400}$

where P is the probability, R_1 is one of the players rank and R_2 is the second players rank. For example, player one has a ranking of 1200 and player two has a ranking of 1000. To calculate the probability of player ones rating versus player two the equation is represented as:

$P = 1/1 + 10^{1200-1000/400}$

The result is .76. This means that player one has a 76 percent chance of winning against player two. The algorithm also calculates the probability of player two winning against player one. This is represented as:

$P = 1/1 + 10^{1000-1200/400}$.

The result is .24 or 24 percent. If the first player wins the amount of points awarded are taken from player two. The difference between player ones rating and player twos rating determines how many points are taken from the loser. In the case of the example, because player one has a higher chance of winning and therefore is expected to win, only a few points would be taken from player two. However if player two wins, which is not expected, then many points would be transferred.

If there is a draw between player one and player two, the Elo algorithm awards the player with the lowest ranking with points and the player with the highest ranking loses points. The amount of points is dependent on the difference between the two rankings of the player. In the example used before where player one has a rating of 1200 and player two has a rating of 1000, player two receives more points than if their rating is 1100. This is because 1100 is closer to 1200 than 1000.

2.2 Reddit Hot Rank Algorithm

The Hot Rank algorithm is most notably used by Reddit, a forum-based website that receives millions of views per day. The Hot Rank algorithm is one the algorithms used to sort the posts within the website. It does this by looking at the number of upvotes as well as the number of downvotes and the submission time of the post. Within Reddit, a user has the ability to give a post single point. This point can either be contributed to the number of upvotes a post has or to the number of downvotes. This is what calculates the submission score. The submission score is a number that represents how popular a post is on Reddit. For example, if there is a post with 200 upvotes and 100 downvotes, then its score is 100 points. This is one parameter that helps the algorithm determine how popular or "hot" a post is. The second parameter that the Hot Rank algorithm takes into consideration is the submission time. This is used to calculate how much time has transpired from when it has posted til the current date. As time goes by, the posts submission score will not decrease but instead newer posts are ranked higher than the first. This is to ensure that more recent posts are viewed by users.

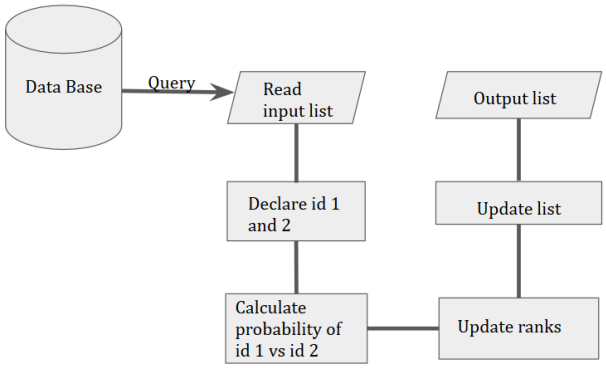

Figure 1: A basic flow chart depicting the relationship between the database and the Elo algorithm and how the Elo algorithm functions on a high level.

3 METHODOLOGY

This section will describe the design of each algorithm, how each is built, the tools used to build them, and the database that holds of the recipes.

3.1 PostgreSQL Database

The purpose of the PostgreSQL database is to host the data list for the algorithms to use.This list is formatted with four columns: [id, cuisine, ingredients, rank] where the id is the primary key, the cuisine and ingredients are simple text data types and the rank is an integer. In a fielded system, the ranks are generated through user input. In other words, the recipe ids associated number of points simulate real user input in order to create the observable quality numbers that the algorithms will produce. Therefore, each recipe is assigned a randomly generated rank (or number of points) that never changes. This database then gets connected to a server that hosts the algorithm scripts, and feeds the algorithms the necessary data for them to begin sorting the list.

3.2 Implementation of the Elo Algorithm

As stated in section 2.1, the Elo rating system ranks players by taking the probability of player one winning against player two and the probability of player two winning against player one. In the case of this experiment instead of players the competitors are recipe ids that are queried from the PostgreSQL database. The recipe ids are treated as players where each id is get compete against each other and get updated Elo ratings. Like determining the probabilities of winning between player one and two, the algorithm determines the probability of recipe id winning against recipe id 2 based on their original ranking. Their original ranking is the original amount of points associated with the recipe id.

As stated in section 2.1, to calculate the probability the formula below is used:

$P = 1/1 + 10^{R_1 - R_2/400}$

where P is the Probability and R_1 and R_2 are the rankings of the first id and the second id. The outputted value, which is a decimal, is stored within a variable. Instead of taking the probability again

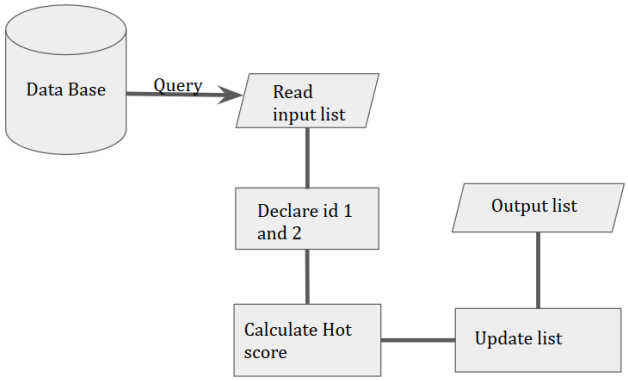

Figure 2: A basic flow chart depicting the relationship between the database and the Hot Rank algorithm and how the Hot Rank algorithm functions on a high level.

for R_2 versus R_1 the probability of R_1 versus R_2 is subtracted from 1.00. This decision was made because taking the probability of R_2 versus R_1 is the same as subtracting the probability of R_1 from 1.00. This result is also stored within another variable. This calculation is used by the next function, EloRating.

The EloRating function does as follows: EloRating(id_1, id_2,R_1,R_1,K,d) takes the the first id and second id, rating of both ids, a variable d and a constant K in order to update the recipe ids rating.

: $id_1$is the first recipe id it encounters within the list.
id_2 is the second recipe id it encounters within the list after recipe id 1.
: R_1 is the number of points associated with id_1
: R_2 is the number of points associated with id_2
: Variable d determines which id has one. It is represented with either a 0 or a 1 for either recipe id. The 0 means that the recipe id has lost and the 1 represents that the recipe has won.
: K determines the maximum number of points an id is able to win or lose. In the case of this experiment the K value is set to 30 because it is the number recommended by other Elo rating systems that have been implemented in the past.

For example to update the recipe ids:

: id_1 is A.
id_2 is B.
: R_1 is 1200.
: R_2 is 1000.
: The max amount of points a recipe can receive will be 30.

EloRating calls Probability to calculate the probability of A winning against B. The result is .76. This gets assigned to a variable in order to take the difference between 1.00 and .76. That result is .24. This is the probability of B winning against A. If A wins then variable d will be 1. This is so the algorithm knows which equation to use to update the Elo ratings for the ids. That equation is the following:

$R_1 = R_1 + K(1 - P_1)$
$R_2 = R_2 + K(0 - P_2)$

where R_1 is the first id's rating, K is the points, the d value is 1 to represent a win, and P_1 is the probability of id 1 winning. The variable R_2 is the second id's rating, K is the points, the d value is 0 to represent a loss, and P_2 is the probability of id 2 winning. In the case of the example the equation is be:

$1207.2 = 1200 + 30(1 - 0.76)$
$992.8 = 1000 + 30(0 - 0.24)$

The new ranking for R_1 is 1207.2 and for R_2 it is 992.8.

If id 2 wins, then both id rankings will be changed using these equations:

$R_1 = R_1 + K(0 - P_1)$
$R_2 = R_2 + K(1 - P_2)$

where the variables represent the same as the previous two equations but the difference is R_2 has its d set to 1 to represent id 2 winning and R_1 has its d set to 0 to represent it losing. The rankings will be updated to the following:

$1177.2 = 1200 + 30(0 - 0.76)$
$1022.8 = 1000 + 30(1 - 0.24)$

These values are then taken to put into a list as such:
$[(A, 1177.2), (B, 1022.8)]$
In order to be used by the next function, Sort.

Sort(list) accepts the list of tuples that contain the id of the recipes and their associated number of points. The number of points is the current ranking for each recipe. Sort calls the function Probability to calculate the probability of recipe id 1 winning against id 2. The result is stored within a variable.

In order to replicate chance the Python library random is used to pick a number between 0 and 1.00. If the probability of id 1 is greater than the randomized number then that means id 1 won, otherwise id 2 has won. It then calls EloRating to calculate the updated ranking. It takes the results from EloRating, which is the updated tuple, to append it onto the updated list to be sorted. The list is then sorted by comparing the new rating of the first id and the new rating of the second id. If the first id has a larger rating then it is placed above the second rating. If the second rating is larger than the first then there are no changes.

For example, the current list is:
[("A", 1200), ("B", 1000), ("C", 1200), ("D", 900)].

The function first compares "A" and "B" by calling the Probability function on 1200 and 1000. The result is below:

$P = 1/1 + 10^{1200-1000/400}$

"A" has a .76 chance of winning against "B". This is stored into a variable. Next, the function randomizes a number from 0 to 1.00. In this example, the randomized number comes out to .50. If .50 is less than .76 then that means "A" has won. In this case "A" does win because .50 is less than .76. Then the EloRating function is called in order to update the function. Since "A" won, its d value is set to 1. This tells the algorithm to use the following equation:

$R_1 = R_1 + K(1 - P_1)$
$R_2 = R_2 + K(0 - P_2)$

where R_1 is "A" and R_2 is "B", K is 30, P_1 is .76 and P_2 is .24. This is shown below:

$1207.2 = 1200 + 30(1 - 0.76)$
$992.8 = 1000 + 30(0 - 0.24)$

The results are then appended into a new list. The current list is:

[("A", 1207.2), ("B", 992.8), ("C", 1200), ("D", 900)].

Since id "A" has a higher rating than id "B" "A" is placed higher than "B". The new result is:

[("B", 992.8), ("A", 1207.2), ("C", 1200), ("D", 900)].

The same methods is used on id "A" and id "C" until the whole list has their updated ratings and is sorted.

In the case of a tie, meaning that the probability of id 1 and id 2 are .50 and the randomized number is also .50, the d value is set to 2 and the K value is set to 10. The algorithm is then uses these equations:

$R_1 = R_1 + K(1 - P_1)$

$R_2 = R_2 + K(1 - P_2)$

The reason why the algorithm considers a tie as both players winning and given a K value of 10 is because in the original Elo algorithm draws result in both players gaining a number points. The difference between the original algorithms method and this implementation in this situation is the amount of points received do not depend on the original rankings.

This implementation of the Elo algorithm is a simulation of how the recipe ids will be ranked through pairwise comparisons. Within a realistic environment the associated number of points given to each recipe id and allows for the Elo algorithm to calculate its values, will be given through user input as opposed to being randomly generated. Therefore the numbers that are shown from this experiment are not real numbers but a simulation to show the hidden quality numbers that the Elo algorithm produces.

3.3 Implementation of the Hot Rank Algorithm

The Hot Rank algorithm takes the submission time and the number of upvotes and downvotes into consideration when determining which recipe id is popular. For experimental purposes, we simulate some aspects of the environment in which the algorithm operates.

The functions of the algorithm are described below:

Seconds(date) determines how many seconds has transpired from when the post was posted to the date. The date posted is generated randomly to simulate a user posting a post at some certain time of day. For example:

The list to be sorted is: [("A", 94), ("B", 50)]. Within this experiment the current date *date* is: $2019 - 12 - 1$.

A date is randomized for "A" first. This date is 2012-01-20. A total of 2872 days and 0 seconds have passed since "A"s date and the current date. This result is passed into the variable td. This equation is used to determine how many seconds are within 2872 days:

$totalseconds = td.days * 86400 + td.seconds + (float(td.microseconds)/1000000$

The current variables represent the following:

- $td.days$ gives the number days without the extra seconds. In this example td.days is be 2872 days.
- $td.seconds$ gives the amount of extra seconds. In this example td.seconds is be 0 seconds.
- $td.microseconds$ gives the number of microseconds in td. In this example td.microseconds is $2.4814e + 14$.

With these results the equation to find the amount of seconds in 2872 days is:

$248140800 = 2872 * 86400 + 0 + (float(2.4814e + 14)/1000000$

This result is used by the next function hot.

hot(ups,downs) determines how popular a post will be by taking the number of upvotes, downvotes, and when the post was submitted. The number of upvotes are the associated number of points that the id has. For example, the previous example has the list: [("A", 94), ("B", 50)]. "A"s upvotes is 94. The number of downvotes are randomly generated for each recipe id. This is to simulate users downvoting the id. For this example the number of downvotes is 10. The date submitted for "A", as stated previously, is $2012 - 01 - 20$. From these values it is able to calculate the following:

Score: Takes the difference between the number of upvotes and downvotes. For example, "A" has 94 upvotes and 10 downvotes. Therefore its score is 84.

Order: This is a logarithmic scale used to reduce the impact of additional votes. While the experiment does not allow the action of additional votes it remains in place to stay true to the Hot Rank algorithm. The equation is: $order = log(max(abs(score), 1), 10)$. For example, id "A"s weight is $1.9 = log(max(abs(84), 1), 10)$.

Sign: Determines whether or not the score is negative or positive. If the score is positive it is presented as a 1 and if the score is negative then it is represented as a 0. For example, the score for id "A" is 84. Since 84 is positive, it is represented as a 1.

DateSeconds: Calculates the amount of seconds of the current date through this equation: $Seconds(date) - 1134028003$. The number 1134028003 represents the UNIX timestamp for the date they introduced the Hot Rank algorithm to the public. For example, the current date in this implementation is 2019-12-1. Therefore DateSeconds is:

$Seconds(248140800) - 1134028003 = -885887203$

where 248140800 is the amount of seconds from the randomly generated date from the function Seconds and the current date.

To calculate how popular a recipe id is according to the Hot Rank algorithm is through the use of the function:

$hot = sign * order + DateSeconds/45000, 7$

In the case of the example used with id "A" the equation is represented as:

$5516.14 = 1 * 1.9 + 248140800/45000, 7$

This means that that the id "A" has a hot score of 5516.14. This is the score that is used to determine what rank "A" holds within the list.

sort(list) sorts the recipe id's depending on their hot score. If one recipe id is determined more hot than the other, then the list is

changed to reflect that by placing the id with the higher hot score above the other id. For example:

The list [("A", 94), ("B", 50)] has two ids, "A" and "B". The hot score for "A" is 5516.14 and the hot score for "B" is 2161.7. It can be assumed for "B" that it has received the same number of downvotes. The function compares both hot scores in order to determine which score is higher than the other. Id "A"s score is higher than id "B"s therefore, id A is placed higher than "B" as such: [("B", 50), ("A", 94)].

In the case of a draw between posts, then nothing will change because this means that both id's were submitted at the same time and received the same amount of upvotes and downvotes. While this is unlikely, it is still in place.

This implementation of the Hot Rank algorithm is a simulation of how the recipe ids will be ranked through pairwise comparisons. Values such as the hot score, the recipe ids associated number points, the date, etc are values that are simulated and therefore not observable in the Hot Rank algorithm in a realistic setting. Therefore the numbers that are shown from the Hot Rank algorithm are a simulation to show the hidden quality numbers that the algorithm produces.

4 RESULTS

For this experiment, big-O notation was used to describe how long the Elo and Hot Rank implementations takes to sort. As described in section 3.2, the implementation of the Elo algorithm contains a total of three functions. These functions are the Probability function, the EloRating function, and the Sort function. Each of these functions has their own time complexity that will be measured. Each of the functions calls the other. This adds onto the overall time complexity of the algorithm. To begin, each of the Elo algorithms functions are listed below:

- Probability: Since this function only returns a result its run time is O(1).
- EloRating: Its purpose to do simple computations and appends therefore the runtime is O(1).
- Sort: This function has two tasks: updating the Elo ratings of all of the id's as well and sorts the final updated list. Both task has a time complexity of O(n). Therefore Sorts overall time complexity is $O(n^2)$. While Sort does call EloRating and Probability, their run times are minuscule enough to not count it towards the overall time complexity.

This implementation of the Elo algorithms time comlexity is therefore: $O(n^2)$

The Hot Rank implementation within this experiment has four functions: DateSeconds, hot, score, and sort. Each of these functions time complexity is listed below:

- DateSeconds: Determines the date the ids were "posted" and how much time has transpired between that day and the current day. Due to the required randomization for each id, the time complexity of DataSeconds is O(n).
- hot: This function calculates how popular a post is. Its calculations give it a time complexity of log_n.
- sort: The purpose of this function is to sort the list of ids based on each ids hot score. This task has a time complexity of $O(n^2)$.

Since each function calls each other their time complexities get added together. Therefore the run time for the Hot Rank algorithm is $O(n^2(log_2))$.

In order to test how long each algorithm practically takes, they were given different sized lists to sort. Below are the results:

Items	Hot Rank	Elo
50	58.90	0.00037
100	118.53	0.00072
150	179.89	0.0012
200	239.27	0.0023
1000	1940.66	0.039

The numbers within the table are approximations of how long the implemented versions of the Elo and Hot Rank algorithm will take. The time varies each run. The difference in time between the Hot Rank algorithm and the Elo algorithm is large. While Hot Rank takes minutes to sort the lists, the Elo algorithm does not even take a second.

5 FUTURE WORK

With the data collected from the tests done within this experiment a possible next step is to build a fully functional recipe app for Android and iOS. This recipe app would function similarly to Tinder, a dating application, where users would be able to swipe through recipes, have the ability to rate them, see which recipes are this weeks most popular, etc. The application is meant to be simplistic in design in order to give users a more streamlined experience. In other words, the application would not have many functionalities since it would be aimed towards users who do not have a lot of time in their day to cook. The application must be able to respond to the users input and have the ability to sort the recipes according to their rank. Therefore the recipe application requires and algorithm that meets this necessity. Another step for this experiment is to run the implemented algorithms on bigger data sets. This is to study how the algorithms practically function with a real data.

6 CONCLUSION

Despite the results given by each algorithm, it is not clear whether or not the original implementations of Elo and Hot Rank are better for pairwise comparisons. Furthermore, from the final sorted list that each algorithm produced the Hot Rank algorithms list made the most logical sense. This is because the ids were sorted based on popularity rather than probability. Therefore the most popular recipe id is ranked the highest rather than the one who happen to win. Nonetheless, the simulated implementations for this experiments show that the Elo algorithm is better suited in sorting the recipe id's due to each of its functions time complexity. This can be contributed to Hot Ranks reliance to more time complex functions as described in section 3.3.

REFERENCES

[1] Bar-Ilan, Mat-Hassan, and Levene M. Methods for comparing ranks of search engine results.
[2] Michel Billard. An introduction to ranking algorithms seen on social news aggregators.
[3] Clark, Connor, Kalita, and Jugal. Comparison of algorithms for the pairwise alignment of biological networks.
[4] R Compton. Elo outside of the competitive gaming realm.
[5] Jin Huang and Charles X. Ling. Rank measures for ordering.

[6] HugoDarwood. Epicurious - recipes with rating and nutrition.
[7] Arpit Mishra. Elo rating system: Common link between facemash and chess!
[8] Amir Salihefendic. How reddit ranking algorithms work.
[9] Swipehelper. How the 'elo score' is calculated and what you can do to improve yours.

[1] [2] [3] [4] [8] [5] [6] [7] [9]