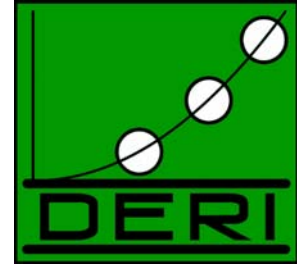


DERI – Digital Enterprise Research Institute



Real-life Rating Algorithm

Mateusz Marmolowski

DERILion Deliverable D1.6.3 - Real-life Rating Algorithm

May 22, 2008

DERI Galway
University Road
Galway
Ireland
www.deri.ie



National University of Ireland, Galway
Ollscoil na bÉireann, Gaillimh



sfi
science foundation ireland
fondúireacht eolaíochta éireann

DERI – Digital Enterprise Research Institute

Real-life Rating Algorithm

Mateusz Marmolowski¹

Abstract:

The number of resources on the Internet increases every day. There are lots of solutions to deal with that amount of data. Rating systems are very useful for ordering resources in terms of importance and usefulness. Resources are being rated by users and the results are presented in the form of ranking.

The majority of rating systems is based on mathematical calculations of rates given by users. The information about users' reliability and social connections is not taken into account. There is a strong need to construct a rating system considering more information about users. Their friendship relations and level of knowledge in various domains can be priceless for every rating system. It's time to propose an idea for the real "human rating" of resources.

We propose the rating algorithm based on social network of domain experts. Our approach takes into account the information about users' friendship relations and their expertise levels in various domains. The process of rating is quite similar to real life situations, because it decides whether a user is reliable or not. Our system also provides an automated mechanism for user's expertise calculation.

Keywords: social network, domain experts, rating algorithm, expertise, ranking

¹ Digital Enterprise Research Institute, National University of Ireland, Galway, University Road, Galway, Ireland, E-mail: mateusz.marmolowski @deri.org

Acknowledgements: In cooperation with John Breslin.

© May, 2008 by the author

Table of Contents

1	INTRODUCTION	4
1.1	Motivation.....	5
1.2	Our goals	6
2	RATING ALGORITHM.....	6
3	RATING CALCULATION	8
3.1	Friendship influence	9
3.2	Domain knowledge.....	10
3.3	Inaccurate domain adjustment	10
3.4	Multiple domains	11
3.5	User Weight	12
3.6	Weighted Rate	12
3.7	Bayesian Weighted Rate.....	13
4	EXPERTISE CALCULATION	15
4.1	Multiple Domains.....	16
4.2	Weighted Expertise Value	16
4.3	Activity factor.....	17
4.4	Final expertise calculation	18
4.5	Expertise propagation process.....	19
5	CONCLUSIONS AND FUTURE WORK	21
6	REFERENCES	22

1 INTRODUCTION

There is a wide variety of approaches to resource rating being used on the Internet. Some of them use simple formulas to calculate the rating for a resource. There are also algorithms that analyze incoming data and search for some dependencies. Those algorithms take into consideration number of rates per user, time of a rate or user's attendance within a system. They are able to distinguish an experienced user from a new one. They are even able to make the rating dependent on resource's lifetime.

The most primitive and popular type is based only on simple mathematical calculations. These algorithms just take all incoming rates and compute the overall rating out of that data. The most popular approach is the Arithmetic Mean. This solution is both the most simple and unreliable. The rating is being calculated out of all rates and an average is presented. However there is a serious, well-known problem with that approach. It is much easier for new resources to get a high overall rating. The extreme situation is the highest rate given by only one user. In that case, a resource will be thought as the best by the rating system. However, for resources that have lots of rates assigned, it's almost impossible to get the best overall rating.

Actually all simple approaches to rating have some advantages. They are efficient, so even a large scale system will have no performance problems to calculate ratings. Additionally, the way they work is clear and comprehensible for users. That means each user knows the rules of the rating. Furthermore the implementation of those simple algorithms is not a big deal, even for a beginner.

On the other hand we have the second category of rating algorithms. They are more complicated to both understand and implement. What is more their efficiency is also lower than the previous ones. Those systems provide multiple mechanisms that calculate various dependencies between users or resources. For instance there is a project [1] concerning users' previous actions and their activity within a system. This approach is based on user's reliability during rating process. The general idea of the algorithm is that good reviewers are those whose reviews predict the ultimate consensus review of a resource. The authors claim that a reviewer who consistently ranks resources near their ultimate average can be considered as a reliable one. What is more the process is completely automated, so there is no need for a user to take any additional action during rating.

According to a user study made by Muzaffer Ozakca and Youn-Kyung Lim [2], users do not treat rating as a major factor during making decisions. It is caused by the lack of trust to users giving those rates. When user does not know anything about rates, he will not be keen to regard any rating as reliable. However, users also claim that they would highly consider the opinion of a friend, especially if they knew his expertise. The research also gives us information about the reviews. The evaluated users knew that every resource,

they were rating, had a review written by a professional. That information was crucial for them, because they could simply trust those reviews whenever they were not experts on themselves. They could be also inspired by reviews given by people having some practical experience with a specific domain, not only the professional experts.

Furthermore, there is also a research made by Don Turnbull [3], where he presents the idea of OpenChoice system. That project is created to build a community of rating users. The system asks every user to rate resources that are not rated sufficiently yet. The author also quotes some popular communities, well-known on the Internet. Almost every Web 2.0 webpage uses at least one type of rating. For example, YouTube [6] allows for interactive rating (1-5 stars), views or comments. Another popular community site Digg.com [7] focuses on simple “thumbs-up” or “thumbs-down” ratings for sites. It also shows the quantity of comments and proposed category for each website.

Last but not least for us is to concern time-varying data. There is a framework called EventRank [4]. It contains proven approaches to analyze time-varying networks. There are some efficient solutions for making snapshots and history analysis. Another, let's say the most popular rating system, is Google PageRank [5]. Almost each website on the Internet is being rated by this algorithm and then ordered into a ranking. This ranking enables Google Search Engine to arrange websites into a sorted list of results. This is the best example of a system with time-varying network. The relations (hyperlink) between websites change every second. PageRank is designed to recalculate values for the whole network basing on actual data. The history of relations is forgotten and not taken into account.

Finally there are also some solutions based on trust between users connected in a social network. One of those systems is FilmTrust [8]. It merges users into a social network. This project uses FOAF ontology [9] for each user's profile representation. As a user you are able to make anybody a friend, but you have to manually define how strong you trust him in films. Although this additional requirement is very useful for rating calculation, it is quite inconvenient for users. If we want that system to be reliable, we need lots of trust relations, what is not easy to achieve nowadays.

1.1 Motivation

Our motivation is based on human real-life situations. When we have to choose between multiple possibilities, we usually ask somebody reliable for his opinion. For instance, if we want to watch a film, we will ask a person who is interested in films or maybe have already watched this film. Every time we aim to rely on people being experts or having high experience in a particular domain.

On the other hand if we want somebody to rate our own work, we know that the most precious opinions are always given by people being experts in a specific domain. What is

more, we should know that our friend will not be the most reliable person, because he is usually subjective towards us. Sometimes he could increase or even decrease his rate on purpose. That is why the ideal situation is to ask an expert that we do not know at all. His opinion will be both objective and reliable.

From the examples, presented above, we can deduce that expertise and friendship are the crucial factors during every rating process. This conclusion motivates us to develop the rating algorithm that takes into consideration those factors.

1.2 Our goals

According to our motivation we want to develop the new, real-life rating algorithm. Our goal is to make this algorithm fully automated and efficient. We do not want to convince users to define additional relations or fill in huge forms step by step. Nowadays nobody likes to be forced to waste his time on unnecessary work. The system must be as friendly as it is possible and its efficiency is also very important.

There are two major challenges that we should cope with. Firstly we have to provide information about users' friendship relations. There are lots of projects that merge people into communities. Our goal is to take advantage of those social networks to be able to gather friendship relations. Secondly we should somehow gather people's expertise levels in various domains. We can ask users about their knowledge, but it is not consistent with our main goal – process automation. What is more some users could be untruthful. So we have to create a reliable, efficient and fully automated solution for expertise collection.

Finally there is also an issue of time-varying ratings. Concerning users' expertise causes some problems with development of their knowledge. Each expertise value changes dynamically, which means it stabilizes due to user's evolution. We should also take this factor into consideration, because it's not obvious whether we should recalculate user's previous rates in terms of his expertise increase or leave the historical data unchanged.

2 RATING ALGORITHM

We propose a reliable and efficient solution for expertise collection. It is quite obvious that people publishing high quality resources in a particular domain can be thought as experts in that domain. The measure of quality can be achieved by other users' rates. For instance, a person having well-rated resources in Biology can be thought as an expert in that field of science. The whole idea is presented in section "Expertise Calculation".

Besides expertise collection we also provide a solution for rating calculation. All disadvantages mentioned before can be forgotten. Our approach is fully automated and reliable as well. Our solution uses FOAFRealm [10] that is an authorization system. FOAFRealm is based on FOAF Ontology and provides information about its users and their friendship relations. Those relations are called “trust levels of friendship” and are defined by users. FOAFRealm users constitute a social network of friendship. There are two important factors: friendship level and distance. Those two parameters describe each pair of users in the social network. The main advantage of FOAFRealm is its connection with FOAF Ontology. The system is used in multiple projects, so every relation should be defined only once. In that way we omit the obligation for users to manually define their friendship relations in every service separately.

The problem of dynamic change of expertise should be also resolved. We claim that user expertise’s variation in the future should not influence on his previous actions. His rates, given in the past, were granted with his previous expertise (lower or higher). Hence, there is no need to recalculate those rates in terms of his evolution, because for instance he is currently more experienced and only his new rates will be more reliable.

Let’s assume that we have a small social network (5 people) where every node is a FOAF Person (profile). The graph of friendship relations is dense. As it is provided by FOAFRealm, the graph is directed and weighted. Every relation has a percentage value informing about the strength of friendship (*Trust level of the friendship*).

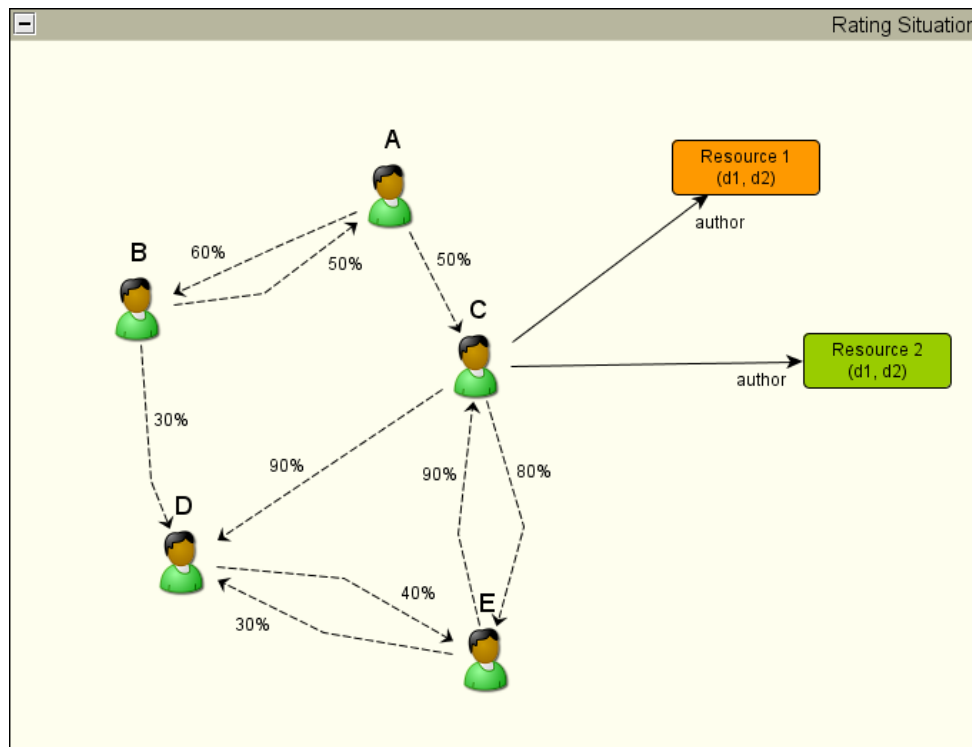


Figure 1: Rating Situation

Our goal is to provide additional information about users' level of expertise in various domains. With that kind of data we are able to make rating much more reliable. Although the usage of reliable expertise data is not a big deal, we introduce a mechanism for gathering users' knowledge. What is more we can also consider friendship influence on user's decisions. Users rate their friends subjectively, basing on their close relations and current attitude to them in the real world.

Let's assume that each resource is tagged within taxonomy, which describes its domain. Figure 1 shows two resources, each being tagged with two categories (d1 and d2). We are able to analyze the expertise of people giving rates according to their knowledge in a specific domain. It means that people being considered as experts in Biology are much more reliable in rating biology resources than users without any expertise in that field of science. Every FOAF Person has its own level of expertise in various domains. The scale of expertise is percentage and it ranges between a predefined value α and 100%. Although the maximum expertise level is always 100%, the minimum value is a parameter (α) and means a level of complete amateur.

There are two major stages in the rating process. Firstly the algorithm has to "weigh" each rate considering user's level of expertise in the rated resource's domain. The calculation of a rate is described in details in the section "Rating calculation". The second stage is to recalculate the "expertise value" for the author of a rated resource. Our goal is to compute user's expertise level out of average ratings of his resources. The recalculation process analyzes all his resources related to each domain. Basically, a user having well rated Biology resources is considered as an expert in that domain. The exact value of expertise depends on average ratings of his Biological resources. Each "expertise value" is being stored in a percentage scale. The details of this process can be found in the section "Expertise calculation".

At the beginning all users have the same level of expertise in each domain by default. Actually they are described as complete amateurs. Each expertise value is set to a minimum value (α). This assumption enables us to provide both, the same rules of weighting rates for every novice user and the encouragement for publishing resources. Every publication is the ability for a user to increase his expertise level in a specific domain. Simply, the activity is strongly recommended.

3 RATING CALCULATION

The main goal of this step is to recalculate a rate, given by user, according to his knowledge and level of friendship with the author of a rated resource. To provide that kind of calculation we need a social network with friendship relations and some additional information about user's level of expertise in various domains. Although the social network is provided by FOAFRealm system, we need to somehow gather "expertise

values” of FOAF users. However in this section we only use those expertise values for calculation. The whole process of expertise gathering is described in details in section “Expertise calculation”.

As it was said before, in our case study we have two resources and five FOAF users. Those users are connected in a social network of friendship.

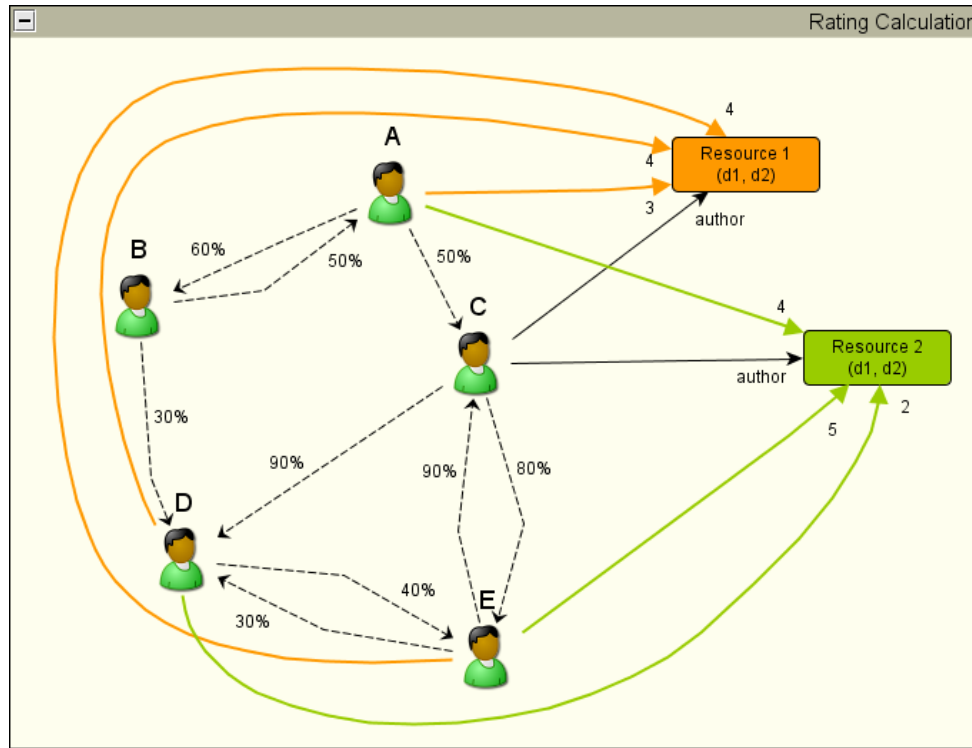


Figure 2: Rating Calculation

The situation presented on Figure 2, shows the resources being rated by users. Let’s analyze the Resource 1 (R_1). User C is the author and users A, D, E have rated his resource (orange lines) within 3, 4, 4 rates respectively. Our rating algorithm takes into account some information about friendship with the author (C) and users’ (A, D, E) expertise in d1 and d2 domains.

3.1 Friendship influence

It’s quite obvious that people who know the author will rate him subjectively. Their rates can be higher or lower than the others’. They do not rate only the quality of a resource but also their attitude to its author. That is why we can simply decrease the weight of their rates and define FW as a Friendship Weight or more properly – objectivity factor:

$$FW(U_x, U_y) = \frac{200\% - \frac{FRIENDSHIP(U_x, U_y)}{DISTANCE(U_x, U_y)}}{2}$$

High value of FW means that user U_x is more objective (reliable) in his rating of the resources of user U_y . In the formula above FRIENDSHIP is a percentage friendship level of the rating user (U_x) with the author of a resource (U_y). DISTANCE means the shortest path from U_x to U_y in the graph of friendship relations. It simply means that users connected indirectly are seemed to be more reliable while having the same friendship value with those connected directly. That is why a higher DISTANCE value increases FW factor, which ranges between 50% and 100%.

3.2 Domain knowledge

The major advantage of the Rating Algorithm is that we take into account each user's domain knowledge. User being said as an expert in a specific domain is much more reliable in his rating decisions. Expertise level of a user in each domain ranges between α and 100%. As it was mentioned before parameter α can be predefined and means a complete amateur. In Rating Calculation process we only get available expertise levels and use them for calculation. However the second stage of the Rating Algorithm calculates expertise values for each user in various domains (taxonomy categories). There is also an Expertise Propagation process that is responsible for expertise estimation for some nodes in a hierarchy of domains. Expertise Values describe user's knowledge and are used in User Weight calculation. This measure (User Weight) informs the algorithm about user's reliability and knowledge for a specific resource during rating process.

3.3 Inaccurate domain adjustment

Every user is able to get an expertise level in a specific domain only after publishing a resource in that domain. There will be lots of categories for every user having no value. In case of any difference between a resource's domain and available user expertise values we provide a mechanism to propagate knowledge level of a user through the taxonomy tree. For example a particular resource regards the domain of Flowers and we know that a user is 90% expert in Biology. If user rates this Flower resource, we should slightly decrease his level of expertise, because Biology is more general than Flowers. It simply means that he will be treated as a medium expert in Flowers, even though he had

expertise value only in Biology. This mechanism should be applied every time if the matching of domains is not similar.

However in Rating Calculation stage we only check if user has any Expertise Value in a specific domain. That value is simply used for calculation. The whole domain adjustment process is being launched during Expertise Calculation stage. The mechanism is described in details in the “Expertise propagation” subsection.

3.4 Multiple domains

The next issue, that should be analyzed, is tagging resources with multiple domains. Each user is able to tag every resource within a tree of categories. What is more he is not forced to use only one category for a resource. He is able to assign multiple domains and choose them from different levels of hierarchy. That opportunity forces us to provide a mechanism for dealing with multiple domain adjustment.

For that purpose we introduce the Multiple Domain factor (MD). The formula is based on geometric series. This is a series with a constant ratio between successive terms. The original formula for the element with index n of geometric series looks as following:

$$S_n = a_1 \frac{1 - q^n}{1 - q}$$

We have adapted that theory to our purpose. The Rating Calculation process needs a factor that could decrease the expertise value in case of many domains being assigned to a resource. Although tagging systems encourage users to use multiple categories for resource descriptions, in our case the expertise value is being calculated out of those categories. We should provide a formula that takes into account all the domains assigned by the author, but their influence on expertise should be decreased. The higher is a number of domains the lower should be the expertise value in each of those categories. Our proposition is the following formula:

$$MD(d) = \frac{1 - q^d}{d}$$

$$q = 0.5$$

d	MD(d)
1	100%
2	75%
3	58%
4	47%
5	39%

Here MD is the Multiple Domain factor and parameter d is the number of domains assigned to a specific resource. The proposition for q parameter is 0.5 and it means that the geometric series approaches 2 in the limit. According to our assumption the MD

value is decreasing due to an increase of d parameter (number of domains being assigned to a resource). An exemplary calculation can be found in the table above. In the first columns we have a possible number of domains assigned to a resource and the second column contains a calculated MD value for that number. This MD value means percentage participation of Expertise Value for one domain. Let's assume that a specific resource has 3 domains assigned and the Expertise Value based on this resource is 70%. Due to the MD factor we should assign 58% of that 80% to each of those 3 domains. Hence, each domain obtains the Expertise Value, which equals 40.6%

3.5 User Weight

Every rate given by a user to a resource is weighted by the algorithm. The process decides whether a user is reliable or not. This weight is called User Weight and is constructed of friendship factor (FW), expertise value (EXP) and multiple domain factor (MD). The User Weight value means the priority of user's rate. The highest value is achieved when user doesn't know the author and additionally he is an expert in all domains assigned to the resource. On the other hand, the lowest UW value can be obtained when he is a close friend of the author (FRIENDSHIP = 100%) and his knowledge in resource's domains equals the possible minimum. The formula for User Weight calculation for a specific resource (R_j) looks as following:

$$UW(U_x, U_y, R_j) = FW(U_x, U_y) \frac{\sum_{i=1}^d EXP(U_x, D_i) * MD(d)}{d * MD(d)}$$

In the formula above U_x means the user giving a rate and U_y is the author of a resource (R_j). Firstly we use Friendship Weight to adjust user's reliability in terms of friendship relations with the author. Then we have EXP function which gets expertise level of user U_x in D_i domain. D is a set of domains assigned to the considered resource (R_j). Finally, we use Multiple Domain factor (MD) to decrease expertise influence in case of high amount of categories being assigned to the resource.

3.6 Weighted Rate

We have already presented all the factors needed in a Weighted Rate calculation process for resource (R_j). The Weighted Rate of a resource is its overall rating in the

system. The WR calculation is based on rates and User Weight values. For this purpose we will use a weighted average. Each rate given by a user is weighted by his User Weight value calculated out of friendship relations and expertise levels in resource's domains. The formula for a specific resource looks as following:

$$WR(R_j) = \frac{\sum_{i=1}^{RN(R_j)} UR(U_i, R_j) * UW(U_i, U_{author}, R_j)}{\sum_{i=1}^{RN(R_j)} UW(U_i, U_j, R_j)}$$

- $WR(R_j)$ – Weighted Rate for resource R_j
- $UR(U_i, R_j)$ – Rate given by user U_i for resource R_j
- $UW(U_i, U_{author}, R_j)$ – User U_i Weight for resource R_j
- U_{author} – The author of R_j
- $RN(R_j)$ – The number of rates for R_j

3.7 Bayesian Weighted Rate

Finally the formula for Weighted Rate calculation should be remodelled regarding the number of rates given by users. The most common problem in lots of rating systems is inadequate rating for new resources. For the most popular approach (arithmetic mean) it is much easier for new resources to get a high overall rating. The extreme situation is the highest rate given by only one user. In that case, a resource will be thought as the best by the rating system. However, for resources that have lots of rates given by users, it's almost impossible to get the best possible overall rating. That is why the activity of rating should be awarded by our Rating Algorithm.

There is an efficient solution for that problem. We could use the Bayesian Rating that decreases influence on the overall average until the number of resources reaches a specified amount. Bayesian Rating is using the Bayesian Average. This is a mathematical term that calculates a rating of an item based on the "believability" of the votes. When there are very few rates, the Bayesian Rating of an item will be closer to the average rating of all items. The formula for a specific resource looks as following:

$$BR = \frac{avg_num_votes * avg_rating + this_num_votes * this_rating}{avg_num_votes + this_num_votes}$$

- avg _ num _ votes* - The average number of votes of all resources
- avg _ rating* - The average rating of all resources' ratings
- this _ num _ votes* - The number of votes for this resource
- this _ rating* - The rating of this resource

Hence, we are able to adjust the Bayesian Rating theory for our Rating Algorithm. Let's introduce the final average rating for a resource and call it Bayesian Weighted Rate. BWR is based on Weighted Rate of a resource and some assumptions that will be explained below.

Apart from the WR value and total number of rates for a resource, the Bayesian Rating also requires average rating and average number of votes out of all resources in the system. In our Rating Algorithm we have a predefined scale of rating. For instance, the rating scale can have 5 grades. It means that users are able to rate resources within 5 different rates (from 1 to 5). The middle value in this scale is 3 and it is a "neutral value" in our case. If a user is able to use 5 grade scale he automatically treats the first two values as a negative vote and the highest two values as positive votes. What is more the middle value (rate 3) is then considered as a neutral value. It means that users use that rate, every time they do not consider a resource as either low or high quality – just neutral.

Although for the 5 grade scale the neutral value is 3, for 10 grade scale it equals 5.5. In even scales there are always two middle rates, so the neutral value should be calculated out of them. That kind of scale is also unfriendly for users, because they always have to decide between positive and negative votes. In odd scales there is always a middle value that user is able to choose in case of neutrality.

In our Bayesian Weighted Rate the "neutral value" is used in place of "avg_rating" from the previous formula. However, we should also provide "avg_num_votes" value. This variable means after how many votes the resource's rating becomes reliable. The "avg_num_votes" value for a specific resource should be based on average number of votes for its domains. The mechanism should check all resources assigned to the same domain as the considered resource. This value should be calculated as following:

$$avg_num = \frac{\sum_{i=1}^d \sum_{x \in D_i} RN(x)}{\sum_{i=1}^d \sum_{x \in D_i} 1}$$

Finally we are able to provide a formula for Bayesian Weighted Rate calculation. This value is being thought in our algorithm as a result rating for a resource. What is more the

Expertise Calculation process is based on BWR values for author's resources. Although "avg_num" changes every time a user rates a resource, we recalculate BWR for a specific resource only when it is being rated directly. Then the rating is being adjusted to the whole evolving system. Hence, the Bayesian Weighted Rate can be considered as stable within our Rating Algorithm. The final formula for BWR looks as following:

$$BWR(R_j) = \frac{avg_num * neutral_value + RN(R_j) * WR(R_j)}{avg_num + RN(R_j)}$$

4 EXPERTISE CALCULATION

After Rating Calculation we are able to start the Expertise Calculation process. This process recalculates user's expertise value in a specific domain. Figure 3 shows a visualization of this algorithm. There is a great efficiency advantage of that approach. We compute the new value only for the author of a resource that has just been rated. That is why the whole Rating Process, described in this document, is constructed from two separate steps. The first step is responsible for BWR value calculation for a specific resource. Then the second step calculates the Expertise Value for this resource's author.

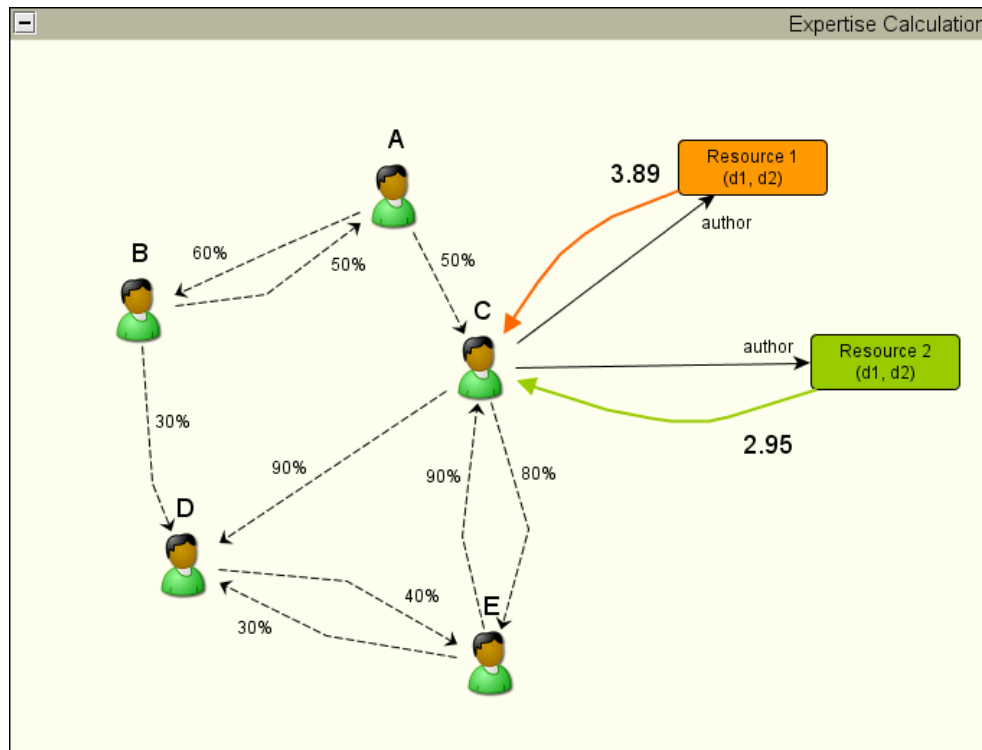


Figure 3: Expertise Calculation

The Expertise Calculation process is also quite complicated and is described in a few subsections. At first we present an approach for dealing with multiple domain assignment – very similar to the one used in Rating Calculation. Then we should compute the Weighted Expertise Value for the author, which is based on his own resources. The value is calculated separately for every domain and is constructed of average ratings of user's all resources assigned to this domain. Furthermore there is an Activity Factor included in Expertise Calculation process. Those users, who publish more, are awarded for their activity.

On the other hand users who have published only few resources are not able to get a high expertise value. This assumption is caused by the well-known problem in simple rating systems. It is much easier to get a high overall rate at the beginning – with only a few rates. However, when the quantity of rates increases, every new rate almost doesn't change the overall average rate of a resource. Finally we propose a mechanism for Expertise Propagation - based on a hierarchy of domains. That propagation estimates expertise levels for categories, where no information about user's knowledge is available. The details of this approach can be found in the subsection Expertise Propagation.

4.1 Multiple Domains

As it was presented in the Rating Calculation stage we have to take into account the number of domains assigned to a resource. The main goal of this factor is to avoid spam situations, when user uses lots of categories just to increase his expertise level in multiple domains. For this purpose we could use the same formula as in the previous stage of the algorithm. Just to remind it looks as following:

$$MD(d) = \frac{1 - q^d}{d} \quad q = 0.5$$

The Multiple Domain factor is being used during the calculation of Weighted Expertise Value. Every resource's average rate is multiplied by this factor. The q parameter is predefined and can be changed during the configuration of our Rating Algorithm.

4.2 Weighted Expertise Value

In this subsection we introduce the recalculation process of expertise values in various domains for a specific user. The algorithm is based on Bayesian Weighted Rates of the resources connected with considered expertise domain. The Weighted Expertise Value is related to a specific user and a specific domain. The algorithm uses the resources

published by a specific user that have assigned the considered domain (category). Every resource tagged within that domain participates in WEV calculation process.

Basically the idea is to construct the Weighted Expertise Value, where weights are numbers of rates given by users and Multiple Domain factors. The algorithm also uses the Bayesian Weighted Rate of each resource for the calculation. We provide the formula for the Weighted Expertise Value calculation for user U_y (author of resources) in a specific domain as follows:

$$WEV(U_y, D_k) = \frac{\sum_{j=1}^r RN(R_j) * MD(d_j) * BWR(R_j)}{\sum_{j=1}^r RN(R_j) * MD(d_j)} * \frac{100\%}{rating_scale}$$

$WEV(U_y, D_k)$ – Weighted Expertise Value for user U_y in domain D_k

$RN(R_j)$ – The number of rates given to the resource R_j

$MD(d_j)$ – Multiple Domain factor for the resource R_j

d_j – The number of domains assigned to R_j

$BWR(R_j)$ – Bayesian Weighted Rate of the resource R_j

r – The number of author's resources in domain D_k

$rating_scale$ – The number of rates (e.g. for scale 1-5 the $rating_scale = 5$)

The construction of the calculation process above strongly depends on rating of user's own resources. The whole expertise value in every domain is only based on ratings of user's resources. The only way to be a domain expert is to publish well-rated resources and categorize them properly. The whole WEV value is also normalized to range between 0 and 100% by the "rating_scale" value.

4.3 Activity factor

All Weighted Expertise Values evolve according to user's activity in publishing resources. It is quite obvious that user having lots of resources (well-rated by the community) can be strongly recommended as an expert in a specific domain.

User's activity consideration is very important, because the more active is a user, the more reliable he is. The common problem in rating systems is an inadequate rating for new users who has published only few resources. They usually have much higher overall

rating than the most active ones. There is an efficient solution for that problem and it is being used during Bayesian Weighted Rate calculation. However, Expertise Value is based on BWR values and we do not need to apply this solution again during the Expertise Calculation process.

Novice users (with low amount of rates) will be simply “punished” because of low BWR values of their own resources. That approach enables us to encourage users to be active and the community to rate lots of resources. The community encouragement is based on a simple rule – rate others and they will rate you.

4.4 Final expertise calculation

The last step of Expertise Calculation process is here described in details. The Expertise Value (EXP) is the overall knowledge level of a user in a specific domain from a hierarchy. This measure is based on Weighted Expertise Value calculated during the previous step. In Rating Algorithm EXP is considered as the final expertise level and is being stored in a database. The value is evolving due to the WEV modifications. The Expertise Value is recalculated every time the Weighted Expertise Value changes.

We should also define the parameter α that has been previously mentioned. This parameter describes a complete amateur in a specific domain. If a user has no calculated or propagated value of expertise for a domain, this initial value (α) is being assigned. In a result in a hierarchy of domains there are no zero values. Either we have a fixed value (calculated out of user’s resources) or there is a temporary value (propagated or initial). That is why the algorithm always has a non-zero EXP value for every user and domain in the system.

The formula for Expertise Value calculation contains two parts. Either a user is a complete amateur and gets the initial value α or he has already published at least one resource and his EXP value equals WEV. The proper formula looks as following:

$$EXP(U_y, D_k) = \left\{ \begin{array}{l} \alpha, \text{ for } WEV(U_y, D_k) = 0 \\ WEV(U_y, D_k), \text{ for } WEV(U_y, D_k) > 0 \end{array} \right\}$$

As it was previously mentioned, the parameter α is being defined during the algorithm configuration. Actually its value is based on “rating_scale” parameter. The complete amateur in our Rating Algorithm should always have the possibly minimal value of expertise. Our goal is to encourage users to publish and rate resources in the system. That is why the initial value of expertise should be at most equal the minimal value that could be calculated by WEV formula. The following equation fulfils the expected conditions:

$$\alpha = \frac{100\%}{rating_scale}$$

Hence, for the 5 grade scale the value of α equals 20%. The same value is achieved for a user who has all resources rated at 1.00. It is really rare to have that extreme situation in our Rating Algorithm, so the formula above can be certainly thought as the proper one.

4.5 Expertise propagation process

Our Rating Algorithm strongly depends on taxonomy. User should assign categories to every resource to define its domain. Only the author of a resource is able to gain a level of expertise in domains that he published his resources. The precision of categorising is the key problem in all systems. Some users are scrupulous and search for the most proper categories. However there are lots of users not interested in browsing the tree of categories. They usually use first two levels of hierarchy, which are more general.

Our system is build to provide expertise levels of users in multiple domains. Besides Expertise Calculation we should also manage with majority of domains for which users have no expertise calculated. The hierarchy of domains enables us to estimate some values. We are able to generalize or particularize user's knowledge known by the algorithm. The estimation for a specific domain is temporary till user publishes a resource regarding this domain and gains the proper, calculated Expertise Value. The goal of expertise propagation is to provide as much knowledge about each user as it is possible and reliable.

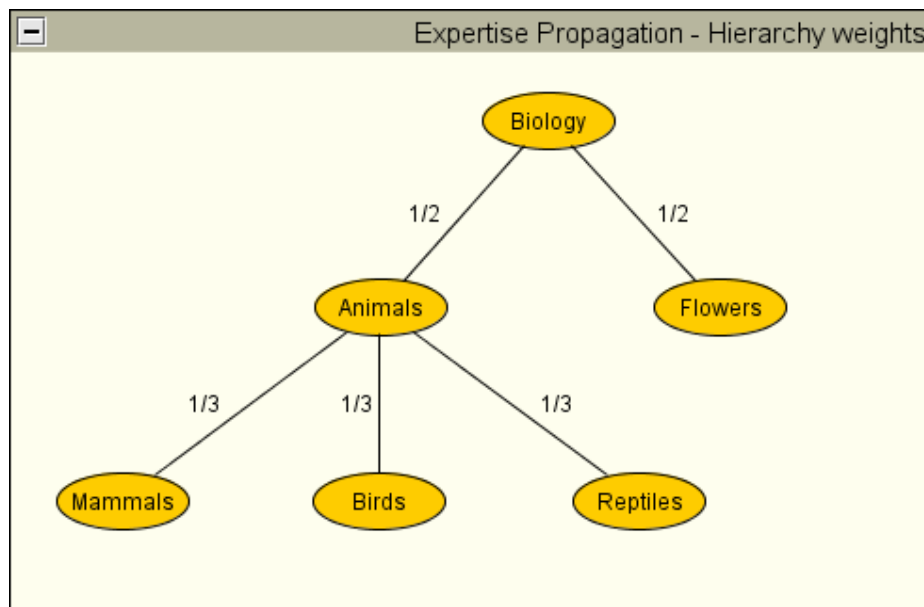


Figure 4: Expertise Propagation – Hierarchy weights

The generalization and particularization are symmetric processes, so the rules of estimation are the same. The idea is to weigh each edge in a hierarchy within the number of direct children. The value of each edge means an influence level of one category to another. For instance, if a category has two direct children, each of two edges leading to them has a weight 0.5. An exemplary hierarchy with weights on edges is shown on Figure 4.

The calculation of weights for a hierarchy is made only once during integration process of hierarchy with the Rating Algorithm. As it was shown on Figure 4 the weighting of taxonomy is very simple. At first, Biology has two children, so the weight of each edge leading to its children is 1/2. Then Animals is divided on three subcategories and each edge gets value 1/3.

As it was mentioned before the propagation mechanism fills up only the nodes without any expertise value based on published resources. What is more the process is executed for the author every time his level of expertise changes. The Expertise Calculation process assigns the new level of expertise to appropriate domain and propagates it upwards and downwards within a hierarchy.

Before giving specific formulas for that expertise propagation, we should introduce some terms to simplify further considerations. There are two types of Expertise Values. The first one is called "fixed value" and is calculated during Expertise Calculation process, so it is based on the resources published by a user. The fixed value has the highest priority and it can't be changed during Expertise Propagation process. The second type of value we call "temporary" and the propagation mechanism is able to change it. The temporary value is either an initial expertise (parameter α) or a level of expertise calculated during prior propagation. The propagation process is always launched for a specific node in the domain hierarchy. What is essential, this node is a start point for the algorithm.

The upwards propagation, also called generalization, is responsible for estimating expertise values for nodes being higher in the hierarchy than a start point. The process checks the parent of a node that has already been changed. If the parent node has no fixed value of expertise, it means the propagation mechanism is able to make a modification. The formula for temporary expertise value calculation (Propagated Expertise) looks as following:

$$PE(parent) = \sum_{C \in children} EXP(U_y, C) * edge_weight$$

$PE(parent)$ - Propagated Expertise for the parent node

$C \in children$ - Each parent's child node

$EXP(U_y, C)$ - Expertise Value for domain C for considered user U_y

$edge_weight$ - Weight of the edge connecting the parent and its child

After generalization, we should introduce the formula for downwards propagation, also called particularization. This process works the other way round – for nodes being lower in the hierarchy of domains. The algorithm should check all children of the start point node. If there was found any temporary value, it should be modified to a new one – estimated by current Expertise Propagation process. The formula for downwards propagation looks as following:

$$PE(child) = EXP(U_y, parent) * edge_weight$$

$PE(child)$ - Propagated Expertise for every child node

$EXP(U_y, parent)$ - Expertise Value for domain C for considered user U_y

$edge_weight$ - Weight of the edge connecting the parent and its child

5 CONCLUSIONS AND FUTURE WORK

If we compare our algorithm to any simple rating based on arithmetical mean, we can observe that our mechanism is much more complicated and closer to real life situations. The major cause of its difference is the distinction between expertise levels of users and their friendship with the author. Users having high level of expertise are more reliable in rating. The reliability is also higher for users having no friendship relations in a social network with the author than users being his close friends.

Our algorithm is now in the implementation stage and the whole system will be then integrated with some websites for evaluation. The system will have a REST SOA, which makes an integration very friendly and easy. There will be also an authorization mechanism for security reasons.

In conclusion we can say that the rating, concerning user's expertise in particular domain and user's friendship relations, is significantly more effective than the one based only on simple mathematical calculations. This algorithm is much more similar to the real life than any other approach. In majority of situations during our life we rather rely on domain experts than people being amateurs in a specific domain.

6 REFERENCES

- [1] Tracy Riggs, Robert Wilensky: An Algorithm for automated Rating of Reviewers. *JCDL'01, June 24-28, 2001, Roanoke, Virginia, USA*
- [2] Muzaffer Ozakca, Youn-Kyung Lim: A study of reviews and ratings on the internet. *CHI '06 extended abstracts on Human factors in computing systems*
- [3] Don Turnbull: Rating, voting & ranking: designing for collaboration & consensus. *CHI '07 extended abstracts on Human factors in computing systems*
- [4] Joshua O'Madadhain, Padhraic Smyth: EventRank: a framework for ranking time-varying networks. *Proceedings of the 3rd international workshop on Link discovery*
- [5] Sergey Brin, Lawrence Page: The anatomy of a large-scale hypertextual Web search engine. *Proceedings of the seventh international conference on World Wide Web 7*
- [6] YouTube: <http://www.youtube.com>
- [7] Digg.com: <http://www.digg.com>
- [8] FilmTrust project. <http://trust.mindswap.org/FilmTrust/>
- [9] FOAF Project. <http://www.foaf-project.org/>
- [10] FOAFRealm: <http://www.foafrealm.org/>