

Jiminy – A Scalable Incentive-based Architecture for Improving Rating Quality

Evangelos Kotsovinos, Petros Zerfos, Nischal M. Piratla, Walid Jerbi
 Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
 {firstname.lastname}@telekom.de



Abstract

Jiminy: a framework for explicitly rewarding users who participate in reputation management systems by submitting ratings. To defend against participants who submit random or malicious ratings in order to accumulate rewards, Jiminy facilitates a probabilistic mechanism to detect dishonesty and halt rewards accordingly.

Jiminy's reward model and honesty detection algorithm, and its cluster-based implementation are highlighted. The proposed framework is evaluated using a large sample of real world user ratings in order to demonstrate its effectiveness. Jiminy's performance and scalability are analysed through experimental evaluation. The system is shown to scale linearly with the on-demand addition of slave machines to the Jiminy cluster, allowing it to successfully process large problem spaces.

User $\rightarrow u$
 Subject $\rightarrow s$
 Rating for subject $s \rightarrow Q_s$
 Log-Likelihood of Q_s based on pdf $\rightarrow L_s$
 $L_s = \ln(\Pr(Q_s))$
 Let B be the subset of subjects rated by u
 Sum of likelihoods over all subjects T_u

$$= \sum_{s \in B} \ln(\Pr(Q_s))$$

 T_u is a random variable and its deviation from mean \bar{T} considering all users and standard deviation \bar{s} :
 $Z = (T_u - \bar{T}) / \bar{s}$
 $|Z|$ is a honesty estimator, also called nose length

Concept of Jiminy - Honesty Metric & Thresholds

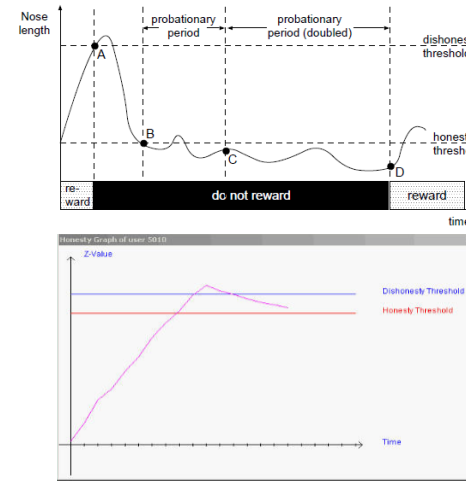


Fig. 1. Jiminy reward model

- When user nose-length rises above a certain threshold (*dishonesty threshold*), user is deemed dishonest
- When user is deemed dishonest rewards for further ratings that she submits to the RMS are halted.
- For user rewards to be resumed, user honesty metric has to fall below the *honesty threshold*, and remain so for a specific period of time (*probationary period*).
- An exponential back-off scheme is followed for the value of the probationary period, which is doubled each time a participant is considered as being dishonest - ensures that there is a substantial penalty for recurring dishonest behaviour, and, at the same time avoid being too strict to first-time cheaters. To achieve this,

Algorithm

1. Update T_u Values()
 - Require:** New ratings added to RMS since last update
 - 1: *AffectedUsers*? Find users that have rated subjects that appear in new ratings
 - 2: **for** (each user u in *AffectedUsers*) **do**
 - 3: *UserSubjects*? Find subjects rated by user u
 - 4: *NewUserRatings*? Find new ratings about *UserSubjects*
 - 5: **for** (each new user rating i in *NewUserRatings*) **do**
 - 6: *Subject*? Subject rated by i
 - 7: *UserRating*? Rating about *Subject* by user u
 - 8: *NumberSame*? Number of ratings about *Subject* equal to *UserRating*
 - 9: *TotalSubjectRatings*? Number of ratings that rate *Subject*
 - 10: T_u ? $T_u - (\log(\text{NumberSame}) - \log(\text{TotalSubjectRatings}))$
 - 11: **if** (*UserRating* = rating of rating i) **then**
 - 12: T_u ? $T_u + (\log(\text{NumberSame} + 1) - \log(\text{TotalSubjectRatings} + 1))$
 - 13: **else**
 - 14: T_u ? $T_u + (\log(\text{NumberSame}) - \log(\text{TotalSubjectRatings} + 1))$
 - 15: **end if**
 - 16: **end for**
 - 17: **end for**

Design and Implementation

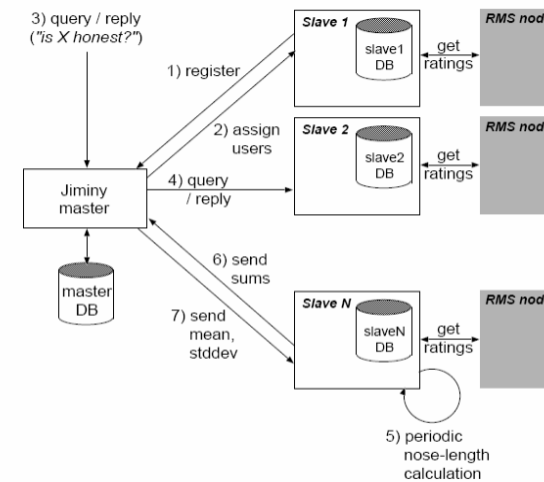
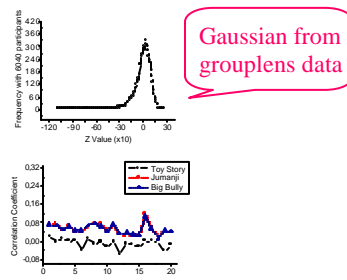


Fig. 2. An overview of the Jiminy clustered architecture

- Multi-process, multi-threaded application in Java
- MySQL backend for storage and retrieval of user data/ratings
- Distributed processing for faster computation of nose lengths
- Master/Slave architecture: Slave registers with Master and gets assigned a subset of users in the systems - subsets are disjoint
- User trustworthiness requests are dispatched to appropriate slave by the Master, when queried
- Master provides a GUI for front-end queries and also to set the threshold values for honesty and dishonesty
- Jiminy system makes use of persistent storage for storing intermediate results and general statistics on the participants and the subjects that are rated
- Failure of slave machines is handled by the master using the Java exception handling facility
- Computation of \bar{T} and standard deviation are performed after aggregation of sum of log-likelihoods of each user at the Master

Honesty Metric - Data Set

- Nose Length distribution is assumed Gaussian with small number of users being dishonest away from the main-lobe
- Above honesty metric is valid, if user ratings are independent



From the correlogram of ratings of various users (in time) using group lens data shows no dependence

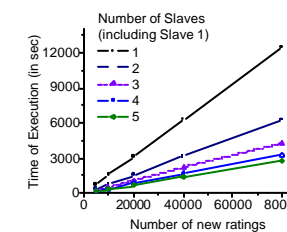
Algorithm Evaluation

- *Mr. Average*. This user periodically queries the RMS to obtain the average rating for each movie he wishes to rate, and subsequently submits the integer rating closest in value to the average rating reported for the same movie. This average rating reported is unlikely to continually be the most popular rating because of the nature of the ratings' distributions.
- *Ms. Popular*. This user periodically queries the RMS to establish the most popular rating for each movie she wishes to rate, which she then submits for the same movie.
- *Mr. Disagree*. This user periodically queries the RMS to obtain the average rating for each movie he wishes to rate, and then reports a rating that is as far from the average value as possible. For instance, he would report 1 if the average rating was 5 and vice versa, and he would report 1 or 5 (at random) if the average rating was 3.
- *Ms. Random*. This user periodically submits a random rating for each movie she wishes to rate.

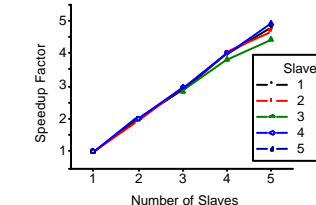
For 43 from the RMS, for above users, the results are as follows

Mr. Average	$T_u = -8.95$	$Z = 19.28$
Ms. Popular	$T_u = -7.84$	$Z = 24.78$
Mr. Disagree	$T_u = -39.35$	$Z = -132.85$
Ms. Random	$T_u = -20.31$	$Z = -37.64$

Scalability



Time taken by the periodic nose-length calculation on slave 1 for different numbers of ratings and slaves present, and



Speedup observed by increasing the number of slaves

For further details:

E. Kotsovinos, P. Zerfos, N. M. Piratla, N. Cameron and S. Agarwal, "Jiminy: A Scalable Incentive-Based Architecture for Improving Rating Quality," *Proceedings 4th International Conference on Trust Management (iTRUST 2006)*, Pisa, Tuscany, Italy, May 2006, pp: 221-235.