MIS Quarterly

RESEARCH ARTICLE

MANAGING THE CROWDS: THE EFFECT OF PRIZE GUARANTEES AND IN-PROCESS FEEDBACK ON PARTICIPATION IN CROWDSOURCING CONTESTS

Lian Jian
Annenberg School of Communication, University of Southern California,
Los Angeles, CA 90089 U.S.A. {ljian@usc.edu}

Sha Yang
Marshall School of Business, University of Southern California,
Los Angeles, CA 90089 U.S.A. {shayang@marshall.usc.edu}

Sulin Ba
School of Business, University of Connecticut,
Storrs, CT 06268 U.S.A. {sulin.ba@uconn.edu}

Li Lu
College of Business & Public Management, West Chester University,
West Chester, PA 19383 U.S.A. {llu@wcupa.edu}

Li Crystal Jiang
Department of Media and Communication, City University of Hong Kong,
Hong Kong, CHINA {crystal.jiang@cityu.edu.hk}

Appendix A

Study Site Description

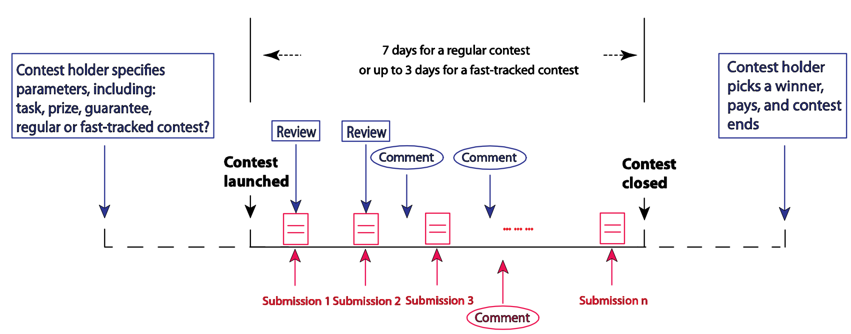

Figure A1. Timeline of a Typical Contest

For each contest, the "Contest Title" column displays the title of the contest (masked out here for anonymity), a short description of the business (masked out), and an image of the business. The "Contest Holder" column shows the screen name (masked out) of the contest host. The "Ends" column shows the time left until the contest closes, the "Entries" column shows the total number of entries the contest has received so far, and the "Package" column shows the prize package, including the amount and whether the prize is guaranteed. Note: Contest # 3's prize is guaranteed. "bronze" is a tag for the contest's prize level. Higher prizes can be marked as silver, gold, and platinum.

Figure A2. Screenshot of Page Displaying All Contests

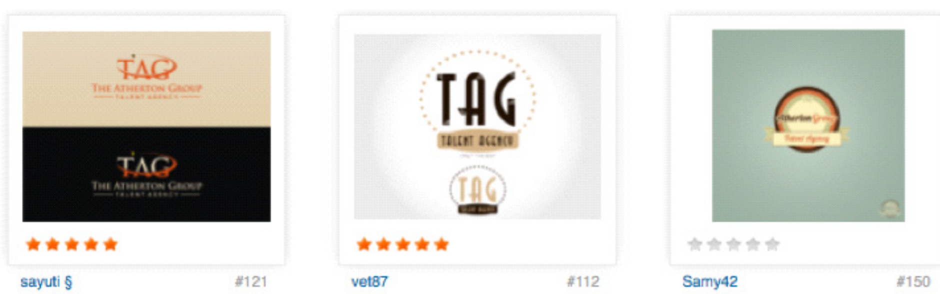

Figure A3. Screenshot of Three Entries Received and Rated in a Contest

As of May 2016, the site we study has hosted more than 350,000 contests and paid more than $100 million to participants. At the time of our data collection in June 2011, 125,527 users were registered on the platform, and more than 6 million designs had been submitted. On this site, a typical contest goes through three stages (see Figure A1): before, during, and after the contest. As soon as a contest is launched, any visitor to the site can discover it by clicking the "browse projects" link on the front page. All projects at various stages are shown on this page (see Figure A2), including the title of the task, the name of the contest host, the time left, the number of entries received, and the prize amount. If a contest's prize is guaranteed, the word "guaranteed" appears under the prize amount. A visitor can sort the contests by the last three columns (i.e., the time left on the contest, the number of entries received, and the prize amount).

By default, the contests are sorted such that those closing the soonest appear on the top. During a contest, all the existing entries are displayed in descending order of their ratings (see Figure A3). Underneath the design, the entry number (indicating the order in which it was submitted), the screen name of the contestant, and the rating it received (if any) are displayed.

Appendix B

Robustness Test of Using the Cumulative Number of Entries by Period *t-1* as the Control Variable for the Current Level of Participation

The results of replacing the control variable $Log(Contestants_{i,t-1})$ with $Log(Entries_{i,t-1})$ reported in Table B1. All the estimated coefficients are similar to those reported in Table 7 in their signs, magnitude, and statistical significance, except that the coefficient on $Log(NegativeReview_{i,t-1}) \times Gua$ has become statistically significant while it was not in the main model in Table 7, thus providing support for H7a (effect of negative reviews moderated).

Table B1. Unconditional NB Model with FE as Dummies Predicting $NewEntries_{i,t}$

	DV = $NewEntries_{i,t}$			
	Unconditional Negative Binomial with Dummy FE			
	Model (1)		Model (2)	
$Log(NegativeReview_{i,t-1}) \times Gua$			0.07*	(0.03)
$Log(ReviewVolume_{i,t-1}) \times Gua$			-0.13**	(0.04)
$Log(HighReview_{i,t-1}) \times Gua$			0.10*	(0.04)
$Log(NegativeComments_{i,t-1}) \times Gua$			-1.20	(0.76)
$Log(CommentVolume_{i,t-1}) \times Gua$			-0.29***	(0.08)
$Log(HighComments_{i,t-1}) \times Gua$			0.31*	(0.14)
$Log(Entries_{i,t-1}) \times Gua$			0.18***	(0.04)
$Log(NegativeReview_{i,t-1})$	-0.05***	(0.01)	-0.08***	(0.02)
$Log(ReviewVolume_{i,t-1})$	0.21***	(0.02)	0.25***	(0.03)
$Log(HighReview_{i,t-1})$	-0.30***	(0.02)	-0.36***	(0.03)
$Log(NegativeComments_{i,t-1})$	-0.94*	(0.43)	-0.08	(0.79)
$Log(CommentVolume_{i,t-1})$	0.27***	(0.04)	0.36***	(0.06)
$Log(HighComments_{i,t-1})$	-0.20**	(0.07)	-0.34***	(0.09)
$Log(Entries_{i,t-1})$	-0.16***	(0.03)	-0.24***	(0.04)
$Log(MedianSubmn_{i,t-1})$	0.05	(0.04)	0.08	(0.04)
$Log(NewContests_{i,t})$	0.02	(0.01)	0.01	(0.01)
Contest-level fixed effects	Yes		Yes	
Period and weekend dummies	Yes		Yes	
Observations	13,665		13,665	
Number of contests	1,031		1,031	
Alpha	0.68		0.67	
LL	-32,602.40		-32,568.30	
AIC	65,250.81		65,182.61	
BIC	65,423.82		65,356.63	

Bootstrapped standard errors are in parentheses. Alpha is the overdispersion parameter. "Gua" is short for Guarantee. ***p < 0.001, **p < 0.01, *p < 0.05.

Appendix C

Robustness Test of Dropping the Last Period

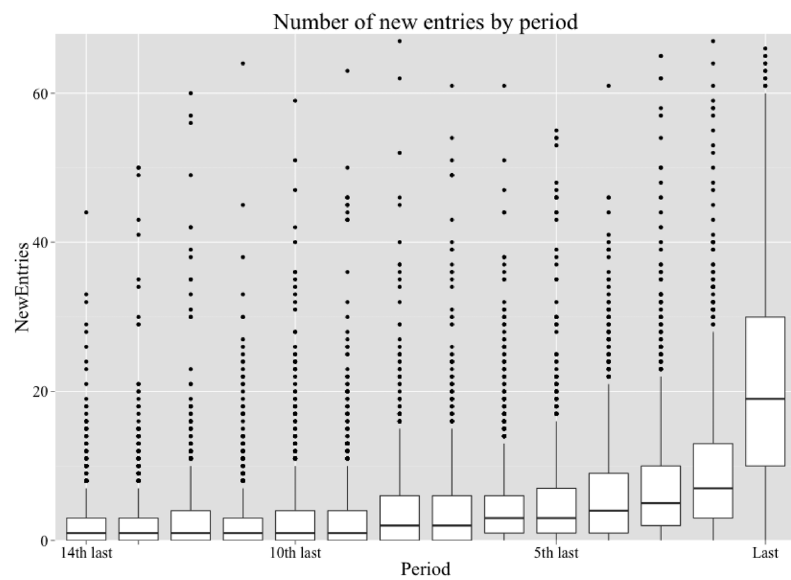

Figure C1. The Number of New Entries Submitted During Each Period

Figure C1 plots the mean number of new submissions by period. Because a spike was observed in the last period, a robustness check was conducted by dropping the last period in the panel. Our analyses show that dropping the last period does not change our results qualitatively. The estimated coefficients as reported in Table C1 are similar to those produced by the main model (in Table 7) in terms of their signs, magnitude, and statistical significance, except that the interaction effect $Log(HighReview_{i,t\text{-}1}) \times Gua$ has become marginally significant (p = 0.06).

Table C1. Unconditional Negative Binomial Model with FE as Dummies Predicting the Number of New Entries During Period *t* in Contest *i*

	DV = $NewEntries_{i,t}$			
	Unconditional Negative Binomial with Dummy FE			
	Model (1)		Model (2)	
$Log(NegativeReview_{i,t-1}) \times Gua$			0.06	(0.04)
$Log(ReviewVolume_{i,t-1}) \times Gua$			-0.10*	(0.04)
$Log(HighReview_{i,t-1}) \times Gua$			0.09+	(0.05)
$Log(NegativeComments_{i,t-1}) \times Gua$			-1.53	(0.99)
$Log(CommentVolume_{i,t-1}) \times Gua$			-0.30**	(0.11)
$Log(HighComments_{i,t-1}) \times Gua$			0.38*	(0.16)
$Log(Entries_{i,t-1}) \times Gua$			0.24***	(0.05)
$Log(NegativeReview_{i,t-1})$	-0.05*	(0.02)	-0.08**	(0.03)
Log($ReviewVolume_{i,t-1}$)	0.12***	(0.03)	0.15***	(0.03)
$Log(HighReview_{i,t-1})$	-0.20***	(0.02)	-0.25***	(0.03)
$Log(NegativeComments_{i,t-1})$	-1.07**	(0.36)	0.01	(0.83)
$Log(CommentVolume_{i,t-1})$	0.26***	(0.05)	0.37***	(0.08)
$Log(HighComments_{i,t-1})$	-0.18*	(0.08)	-0.37**	(0.12)
$Log(Entries_{i,t-1})$	0.09	(0.06)	-0.03	(0.07)
$Log(MedianSubmn_{i,t-1})$	-0.12*	(0.05)	-0.09	(0.06)
$Log(NewContests_{i,t})$	0.04**	(0.01)	0.03*	(0.01)
Contest-level fixed effects	Yes		Yes	
Period and weekend dummies	Yes		Yes	
Observations	12,634		12,634	
Number of contests	1,031		1,031	
Alpha	0.68		0.68	
LL	-28,355.70		-28,322.07	
AIC	56,771.40		56,704.13	
BIC	56,994.73		56,927.46	

Bootstrapped standard errors are in parentheses. Alpha is the over-dispersion parameter. “Gua” is short for Guarantee. *** p<0.001, ** p<0.01, * p<0.05, + p<0.1

Appendix D

Propensity Score Matching Diagnosis

Figure D1 demonstrates that the distributions of propensity scores among contests with and without guarantees exhibit substantial overlap, indicating sufficient matching.

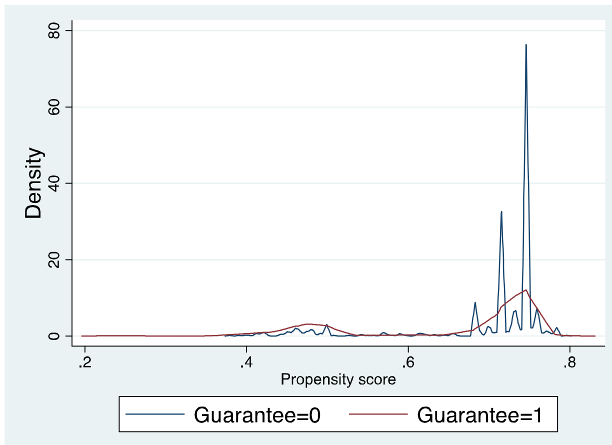

Figure D1. Distribution of Propensity Scores Among Contests With and Without Prize Guarantees

Appendix E

Instrument Variable Analyses

Descriptive statistics of the instrument variables are reported in Table E1 and the first stage regression results are reported in Table E2.

Table E1. Descriptive Statistics of Instrumental Variables

Variable	N	Mean	Std. Dev.	Min	Max
$SimNegativeReview_{i,t\text{-}1}$	13,665	4.25	12.89	0	479.14
$SimReviewVolume_{i,t\text{-}1}$	13,665	11.85	13.99	0	94.17
$SimHighReview_{i,t\text{-}1}$	13,665	2.13	6.31	0	277.11
$SimNegativeComments_{i,t\text{-}1}$	13,665	0.0001	0.002	0	0.07
$SimCommentVolume_{i,t\text{-}1}$	13,665	0.58	0.82	0	3.43
$SimHighComments_{i,t\text{-}1}$	13,665	0.11	0.34	0	2.2

Table E2. First Stage Regression Results												
DV	Log(*NegReview*$_{i,t\text{-}1}$)		Log(*ReviewVol*$_{i,t\text{-}1}$)		Log(*HighReview*$_{i,t\text{-}1}$)		Log(*NegComm*$_{i,t\text{-}1}$)		Log(*CommVol*$_{i,t\text{-}1}$)		Log(*HighComm*$_{i,t\text{-}1}$)	
Log(*SimNegReview*$_{i,t\text{-}1}$)	0.35***	(0.02)	0.63***	(.01)	0.45***	(0.02)	0.00	(.00)	0.05**	(0.02)	0.00	(.01)
Log(*SimReviewVol*$_{i,t\text{-}1}$)	-0.06***	(0.01)	0.26***	(.00)	-0.04***	(0.01)	0.00*	(.00)	-0.04***	(0.01)	0.00	(.00)
Log(*SimHighReview*$_{i,t\text{-}1}$)	0.39***	(0.02)	0.15***	(.01)	0.31***	(0.02)	0.00	(.00)	0.10***	(0.02)	0.01*	(.01)
Log(*SimNegComm*$_{i,t\text{-}1}$)	-2.80	(2.48)	-2.10*	(.87)	1.78	(2.45)	1.93***	(.10)	12.02***	(2.00)	-0.50	(.59)
Log(*SimCommVol*$_{i,t\text{-}1}$)	-0.02*	(0.01)	-0.03***	(.00)	0.13***	(0.01)	0.00*	(.00)	0.34***	(0.01)	0.00	(.00)
Log(*SimHighComm*$_{i,t\text{-}1}$)	-0.01	(0.03)	-0.02	(.01)	0.21***	(0.03)	0.00	(.00)	1.81***	(0.03)	0.93***	(.01)
Log(*Contestants*$_{i,t\text{-}1}$)	0.12***	(0.01)	0.11***	(.00)	-0.13***	(0.01)	0.00	(.00)	0.06***	(0.01)	-0.01	(.00)
Log(*MedianSubmn*$_{i,t\text{-}1}$)	-0.04***	(0.01)	0.03***	(.00)	0.06***	(0.01)	0.00	(.00)	0.06***	(0.01)	0.01***	(.00)
Log(*NewContests*$_{i,t}$)	-0.01	(0.01)	-0.01*	(.00)	-0.02**	(0.01)	0.00	(.00)	-0.02**	(0.01)	0.00	(.00)
Contest-level FE	Yes		Yes		Yes		Yes		Yes		Yes	
Period dummies	Yes		Yes		Yes		Yes		Yes		Yes	
Weekend dummies	Yes		Yes		Yes		Yes		Yes		Yes	
Observations	13,665		13,665		13,665		13,665		13,665		13,665	
Number of contests	1,031		1,031		1,031		1,031		1,031		1,031	
R^2	0.49		0.96		0.55		0.07		0.58		0.65	

Standard errors in parentheses. Due to space limitations, abbreviations are used, including “Neg” = Negative; “Comm” = Comment or Comments; “Vol” = Volume.
***p < 0.001, **p < 0.01, *p < 0.05

Appendix F

Robustness Test of Using Per-Period (i.e., Noncumulative) Independent Variables

We have added analyses by an alternative model in which the cumulative independent variables were replaced with per-period measures. In Table F1, we report results from an unconditional negative binomial model with fixed effects modeled as dummies. Overall, the results are qualitatively similar to our main results. All the main effects of in-process feedback have retained their signs, magnitude, and statistical significance except for the coefficient on *Log*(*NewNegativeComments*$_{i,t\text{-}1}$), which has retained the correct sign. The interaction effects have retained the expected signs but their statistical significance has changed. In particular, among the four moderating effects identified in the main model (Table 7), only the coefficient on *Log*(*NewReviewVolume*$_{i,t\text{-}1}$) X Gua is still statistically significant. However, two other coefficients that were not statistically significant in the main model have become statistically significant: *Log*(*NewNegativeReview*$_{i,t\text{-}1}$) × *Gua* and *Log*(*NewNegativeComments*$_{i,t\text{-}1}$) × *Gua*. In summary, the results using per-period measures of independent variables are largely consistent with our main results.

Table F1. Results of Robustness Tests Using Per-Period Independent Variables in an Unconditional Negative Binomial Model with FE as Dummies

	DV = $NewEntries_{i,t}$			
	Unconditional Negative Binomial with Dummy FE			
	Model (1)		Model (2)	
$Log(NewNegativeReview_{i,t-1}) \times Gua$			0.09***	(0.03)
$Log(NewReviewVolume_{i,t-1}) \times Gua$			-0.09***	(0.03)
$Log(NewHighReview_{i,t-1}) \times Gua$			0.06	(0.05)
$Log(NewNegativeComments_{i,t-1}) \times Gua$			29.00***	(1.72)
$Log(NewCommentVolume_{i,t-1}) \times Gua$			-0.10	(0.12)
$Log(NewHighComments_{i,t-1}) \times Gua$			0.22	(0.17)
$Log(Contestants_{i,t-1}) \times Gua$			0.06*	(0.03)
$Log(NewNegativeReview_{i,t-1})$	-0.07***	(0.02)	-0.12***	(0.02)
$Log(NewReviewVolume_{i,t-1})$	0.22***	(0.02)	0.26***	(0.02)
$Log(NewHighReview_{i,t-1})$	-0.16***	(0.02)	-0.19***	(0.04)
$Log(NewNegativeComments_{i,t-1})$	-1.14	(0.85)	-29.90***	(2.02)
$Log(NewCommentVolume_{i,t-1})$	0.33***	(0.05)	0.39***	(0.06)
$Log(NewHighComments_{i,t-1})$	-0.20*	(0.10)	-0.31**	(0.12)
$Log(Contestants_{i,t-1})$	0.10***	(0.03)	0.06	(0.09)
$Log(MedianSubmn_{i,t-1})$	-0.14***	(0.03)	-0.13	(0.07)
$Log(NewContests_{i,t})$	0.02**	(0.01)	0.02	(0.01)
Contest-level fixed effects	Yes		Yes	
Period and weekend dummies	Yes		Yes	
Observations	13,665		13,665	
Number of contests	1,031		1,031	
Alpha	0.66		0.66	
LL	-32,545.46		-32,536.65	
AIC	65,220.92		65,095.30	
BIC	65,709.89		65,178.05	

Bootstrapped standard errors are in parentheses. Alpha is the over-dispersion parameter. "Gua" is short for Guarantee. ***$p < 0.001$, **$p < 0.01$, *$p < 0.05$

Appendix G

Robustness Test of Using Arellano-Bond Dynamic Panel-Data Estimator

The results of Arellano-Bond dynamic panel-data estimator with a lagged DV are reported in Table G1. All the coefficients reported in the main model (Table 7) have retained their signs, magnitude, and statistical significance, except the coefficient on $Log(NegativeComments_{i,t-1})$ which has the expected sign but has lost its statistical significance.

Table G1. Results of Arellano-Bond Dynamic Panel-Data Estimator

	DV = $Log(NewEntries_{i,t})$			
	Arellano-Bond Dynamic Panel-Data Estimator			
	Model (1)		Model (2)	
$Log(NegativeReview_{i,t-1}) \times Gua$			0.12	(0.05)
$Log(ReviewVolume_{i,t-1}) \times Gua$			-0.19*	(0.04)
$Log(HighReview_{i,t-1}) \times Gua$			0.08*	(0.06)
$Log(NegativeComments_{i,t-1}) \times Gua$			-3.20	(2.30)
$Log(CommentVolume_{i,t-1}) \times Gua$			-0.15**	(0.11)
$Log(HighComments_{i,t-1}) \times Gua$			0.49*	(0.20)
$Log(Contestants_{i,t-1}) \times Gua$			0.04**	(0.07)
$Log(NegativeReview_{i,t-1})$	-0.22***	(0.03)	-0.26***	(0.03)
$Log(ReviewVolume_{i,t-1})$	0.17***	(0.02)	0.24***	(0.03)
$Log(HighReview_{i,t-1})$	-0.50***	(0.03)	-0.52***	(0.04)
$Log(NegativeComments_{i,t-1})$	-3.81	(1.47)	-1.35	(0.58)
$Log(CommentVolume_{i,t-1})$	0.18***	(0.05)	0.27***	(0.08)
$Log(HighComments_{i,t-1})$	-0.31*	(0.10)	-0.54**	(0.14)
$Log(Contestants_{i,t-1})$	-0.61	(0.04)	-0.62*	(0.05)
$Log(MedianSubmn_{i,t-1})$	-0.21***	(0.03)	-0.22***	(0.03)
$Log(NewContests_{i,t})$	-0.01	(0.01)	-0.01	(0.01)
$Log(NewEntries_{i,t-1})$	0.09***	(0.02)	0.09***	(0.02)
Contest-level fixed effects	Yes		Yes	
Period, day/night and weekend dummies	Yes		Yes	
Observations	13,665		13,665	
Number of contests	1,031		1,031	
Wald χ^2	3810.55(24)		3806.09(31)	

Robust standard errors are in parentheses. "Gua" is short for Guarantee. For Wald χ^2 tests, the degrees of freedom are reported in parentheses. ***p < 0.001, **p < 0.01, *p < 0.05.

Appendix H

Contestant-Level Analysis

We conducted individual-level analysis to verify our main results obtained at the contest level. In this contest-contestant-period dataset, each observation focuses on the outcome variable, $n_{i,j,t}$, the number of submissions contestant j submits to contest i in period t. The dataset contains 1,031 contests and 5,545 contestants. An observation of contest i-contestant j-period t is included in the dataset only if contestant j has submitted at least an entry to contest i before or during period t. In-process feedback variables are added into the model together with three sets of control variables, $ContestantControls_{i,j,t-1}$, $ContestControls_{i,t-1}$, and $PeriodControls_{i,t}$, as well as two fixed effects at the contest and contestant level respectively, C_i and C_j.

$$\begin{aligned}\log\left(n_{i,j,t}\right) = {} & \alpha_0 + \alpha_1 \log\left(NegativeReview_{i,t-1}\right) + \alpha_2 \log\left(ReviewVolume_{i,t-1}\right) \\ & + \alpha_3 \log\left(HighReview_{i,t-1}\right) + \alpha_4 \log\left(NegativeComment_{i,t-1}\right) \\ & + \alpha_5 \log\left(CommentReview_{i,t-1}\right) + \alpha_6 \log\left(HighComment_{i,t-1}\right) \\ & + \alpha_7 ContestControls_{i,t-1} + \alpha_8 ContestantControls_{i,t-1} \\ & + \alpha_9 PeriodControls_{i,t} + \delta_i C_i + \delta_j C_j + \varepsilon_{i,j,t}\end{aligned} \tag{3}$$

The $ContestControls_{i,t\text{-}1}$ is a vector of contest-period specific variables, which includes only $Entries_{i,t\text{-}1}$ (cumulative number of entries contest *i* receives by period *t*-1). $ContestantControls_{i,j,t\text{-}1}$ includes three variables describing the cumulative reviews contestant *j* has received from contest *i* as of period *t*-1: $SelfNegativeReview_{i,j,t\text{-}1}$, $SelfReviewVolume_{i,j,t\text{-}1}$, and $SelfHighReview_{i,j,t\text{-}1}$. These three variables were introduced because prior research has shown that direct feedback received by participants to their own submissions have strong effects on their subsequent submissions (Jiang et al. 2016; Wooten and Ulrich 2016; Yang et al. 2013). The $PeriodControl_{i,t}$ is a vector of contest-period specific variables, which includes $Weekend_{i,t}$, $Period_t$, and $NewContests_{i,t}$. To examine the interaction effects, we add

$$\alpha_k = \alpha_{k0} + \alpha_{k1} Guarantee_i \tag{4}$$

where k = {1, 2, 3, 4, 5, 6, 7, 8}. That is, the first six independent variables in equation (3) are interacted with the variable $Guarantee_i$.

In Table H1, we report the results of contestant level analyses using two models: a linear model with both contest and contestant level fixed effects and an unconditional negative binomial model with contest-level fixed effects modeled as dummies. The results show that these two models yield results highly consistent in the sign and statistical significance of the coefficients. The results show that all our main results about in-process feedback are born out at the individual contestant level, except the main effect of negative comments (H3b), the effects of comment volume moderated (H5b) and high comments moderated (H7b).

Table H1. Results of Individual Contestant-Level Analyses								
	DV = *Log*(*NewEntries*$_{i,j,t}$)				**DV = *NewEntries*$_{i,j,t}$**			
	Linear				**Unconditional NB with FE as Dummies**			
	Model (1)		**Model (2)**		**Model (3)**		**Model (4)**	
Log(*NegativeReview*$_{i,t-1}$) × *Gua*			0.00	(0.01)			0.01	(0.02)
Log(*ReviewVolume*$_{i,t-1}$) × *Gua*			-0.06***	(0.01)			-0.13***	(0.03)
Log(*HighReview*$_{i,t-1}$) × *Gua*			0.03	(0.01)			0.12***	(0.02)
Log(*NegativeComments*$_{i,t-1}$) × *Gua*			0.27	(0.27)			0.14	(0.56)
Log(*CommentVolume*$_{i,t-1}$) × *Gua*			-0.09	(0.07)			0.01	(0.11)
Log(*HighComments*$_{i,t-1}$) × *Gua*			-0.03	(0.06)			-0.06	(0.10)
Log(*NegativeReview*$_{i,t-1}$)	-0.02**	(0.01)	-0.02*	(0.01)	-0.05***	(0.01)	-0.06***	(0.01)
Log(*ReviewVolume*$_{i,t-1}$)	0.12***	(0.01)	0.15***	(0.01)	0.22***	(0.01)	0.28***	(0.02)
Log(*HighReview*$_{i,t-1}$)	-0.07***	(0.01)	-0.08***	(0.01)	-0.19***	(0.01)	-0.24***	(0.02)
Log(*NegativeComments*$_{i,t-1}$)	-0.09	(0.09)	-0.35	(0.26)	-0.03	(0.18)	-0.19	(0.53)
Log(*CommentVolume*$_{i,t-1}$)	0.13***	(0.03)	0.19***	(0.05)	0.32***	(0.05)	0.31***	(0.09)
Log(*HighComments*$_{i,t-1}$)	-0.10***	(0.03)	-0.09	(0.05)	-0.19***	(0.05)	-0.16*	(0.08)
Log(*SelfNegativeReview*$_{i,j,t-1}$) × *Gua*			0.00	(0.01)			0.01	(0.01)
Log(*SelfReviewVolume*$_{i,j,t-1}$) × *Gua*			0.01	(0.01)			0.02*	(0.01)
Log(*SelfHighVolume*$_{i,j,t-1}$) × *Gua*			-0.03	(0.01)			-0.07***	(0.01)
Log(*Entries*$_{i,t-1}$) × *Gua*			0.05***	(0.01)			0.04	(0.02)
Log(*SelfNegativeReview*$_{i,j,t-1}$)	-0.02***	(0.00)	-0.02**	(0.01)	-0.04***	(0.01)	-0.04***	(0.01)
Log(*SelfReviewVolume*$_{i,j,t-1}$)	0.02***	(0.00)	0.02***	(0.01)	0.08***	(0.00)	0.08***	(0.01)
Log(*SelfHighVolume*$_{i,j,t-1}$)	0.09***	(0.01)	0.10***	(0.01)	0.09***	(0.01)	0.13***	(0.01)
Log(*Entries*$_{i,t-1}$)	-0.47***	(0.01)	-0.49***	(0.01)	-0.63***	(0.02)	-0.65***	(0.02)
Log(*NewContests*$_{i,t}$)	-0.01*	(0.00)	-0.01*	(0.00)	-0.03***	(0.01)	-0.03***	(0.01)
Individual-level fixed effects	Yes				No			
Contest-level fixed effects	Yes				Yes			
Period and weekend dummies	Yes				Yes			
Observations	132,575				132,575			
R^2 or pseudo R^2	0.0723		0.0728		0.0400		0.0401	
AIC	334,692		334,661		270,417		270,365	
BIC	345,006.10		345,073		280,760		280,806	

Robust standard errors are in parentheses. Dataset contains 1,031 contests and 5,545 contestants. "Gua" is short for Guarantee. For Wald Chi^2 tests, the degrees of freedom are reported in parentheses. ***$p < 0.001$, **$p < 0.01$, *$p < 0.05$.

Appendix I

Alternative DV: Number of New Contestants Entering Contest *i* in Period *t*

In our main results we focused on the number of new submissions as our dependent variable. An alternative measure of participation is the number of participants. It is important to test our model with this alternative dependent variable, for at least two reasons. First, it would potentially rule out an alternative hypothesis that in-process feedback led to more submissions simply because it encouraged more repeated submissions by the current participants (perhaps the direct receivers of the feedback), but not because it attracted more new participants. Second, the number of participants is of theoretical interest because in creative-design contests more participants might lead to more innovative ideas.

We retested our hypotheses by replacing the DV with $Contestants_i$ (the total number of contestants) in the cross-sectional analysis and with $NewContestants_{i,t}$ (the number of participants who made their first submissions to contest *i* during period *t*) in the panel analysis. $Contestants_i$ has a mean of 18.25 and a standard deviation of 25.77 and $NewContestants_{i,t}$ has a mean of 1.84 and a standard deviation of 4.30.

To test the effect of *Guarantee*, we again report results from five estimates of treatment effects (i.e., PSM, NNM, RA, IPW, and IPWRA). Since the first stages (e.g., computing the propensity score or matching) were exactly the same as those used in the main model, their results as well as balance examinations are omitted. Treatment effects estimated across the five methods (Table I1) show that *Guarantee* had a positive effect on $Contestants_i$. The ATETs were estimated to range from 12.24 to 14.45.

Table I1. Treatment Effects of Guarantee on Total $Contestants_i$

	Estimated ATET	Observations
Propensity Score Matching (PSM)	12.62***(2.23)	644 treated and control
Nearest Neighbor Matching (NNM)	14.45***(1.94)	644 treated and control
Regression Adjustment (RA)	12.46***(1.96)	1,031
Inverse-Probability Weighting (IPW)	12.68***(1.96)	1,031
IPW Regression Adjustment (IPWRA)	12.24***(1.91)	1,031

Robust standard errors are in parentheses and for both PSM and NNM, robust Abadie-Imbens[1] standard errors are reported. For both PSM and NNM, the matching ratio was 1:1. ***p < 0.001, **p < 0.01, *p < 0.05.

Results about the main effects of in-process feedback (reported in Table I2) are largely consistent with results from our main model, with a few losing statistical significance. While our main model has yielded support for all the main effects (H2a–H4b), with $NewContestants_{i,t}$, H3a (effect of negative reviews), H3b (effect of negative comments), and H4b (effect of high comments), have lost support. For hypotheses regarding the interaction effects, the main model has yielded support for the following four hypotheses: H5a (effect of review volume moderated), H5b (effect of comment volume moderated), H7a (effect of high reviews moderated), and H7b (effect of high comments moderated). With this alternative DV (see AMEs reported in Table I3), all four coefficients have the correct signs, and H5a and H7a were supported (the coefficient supporting H7a was marginally significant).

Overall, the results predicting the number of new participants do not differ substantially with those predicting the number of new entries. The effect of *Guarantee* (H1) and the main effects of the review volume (H2a), comment volume (H2b), and high reviews (H4a) still hold. Two important interaction effects also hold, including the effect of review volume moderated (H5a) and the effect of high reviews moderated (H7a).

---

[1]The robust standard error for PSM was derived based on Abadie and Imbens (2016), accounting for the fact that the propensity score was estimated prior to matching. The robust standard error for NNM was computed based on methods derived by Abadie and Imbens (2006, 2011, 2016).

Table I2. Unconditional Negative Binomial Model with FE as Dummies, Predicting the Number of New Contestants Who Entered Contest *i* During Period *t*

	DV = $NewContestants_{i,t}$			
	Unconditional Negative Binomial with Dummy FE			
	Model (1)		Model (2)	
$Log(NegativeReview_{i,t-1}) \times Gua$			0.07*	(0.03)
$Log(ReviewVolume_{i,t-1}) \times Gua$			-0.11**	(0.04)
$Log(HighReview_{i,t-1}) \times Gua$			0.13**	(0.05)
$Log(NegativeComments_{i,t-1}) \times$ Gua			-0.42	(6.59)
$Log(CommentVolume_{i,t-1}) \times Gua$			-0.26*	(0.11)
$Log(HighComments_{i,t-1}) \times Gua$			0.33*	(0.14)
$Log(Contestants_{i,t-1}) \times Gua$			0.23***	(0.05)
$Log(NegativeReview_{i,t-1})$	0.04*	(0.02)	0.00	(0.02)
$Log(ReviewVolume_{i,t-1})$	0.04*	(0.02)	0.09***	(0.03)
$Log(HighReview_{i,t-1})$	-0.30***	(0.02)	-0.39***	(0.03)
$Log(NegativeComments_{i,t-1})$	-0.44	(0.57)	-0.25	(6.48)
$Log(CommentVolume_{i,t-1})$	0.13**	(0.05)	0.22*	(0.10)
$Log(HighComments_{i,t-1})$	-0.16	(0.08)	-0.34**	(0.10)
$Log(Contestants_{i,t-1})$	-0.22***	(0.05)	-0.34***	(0.04)
$Log(MedianSubmn_{i,t-1})$	0.02	(0.04)	0.05	(0.05)
$Log(NewContests_{i,t})$	0.02*	(0.01)	0.02	(0.01)
Contest-level fixed effects	Yes		Yes	
Period and weekend dummies	Yes		Yes	
Observations	13,665		13,665	
Number of contests	1,031		1,031	
Alpha	0.29		0.29	
LL	-20,135.14		-24,852.01	
AIC	40,332.29		40,216.32	
BIC	40,565.49		40,359.25	

Bootstrapped standard errors are in parentheses. Alpha is the overdispersion parameter. “Gua” is for Guarantee. ***p < 0.001, **p < 0.01, *p < 0.05.

Table 20. Average Marginal Effects of In-Process Feedback on $NewContestants_{i,t}$

	Average Marginal Effects of Feedback on $NewContestants_{i,t}$					
	Guarantee = 1		*Guarantee* = 0		Difference	
$Log(NegativeReview_{i,t-1})$	0.21	(0.14)	0.00	(0.06)	0.21	(0.14)
$Log(ReviewVolume_{i,t-1})$	-0.07	(0.11)	0.26*	(0.11)	-0.33*	(0.17)
$Log(HighReview_{i,t-1})$	-0.79***	(0.39)	-1.14*	(0.50)	0.36+	(0.19)
$Log(NegativeComments_{i,t-1})$	-2.00	(1.39)	-0.72	(18.96)	-1.28	(19.43)
$Log(CommentVolume_{i,t-1})$	-0.11	(0.19)	0.64	(0.47)	-0.75	(0.56)
$Log(HighComments_{i,t-1})$	-0.01	(0.20)	-0.98	(0.61)	0.97	(0.62)

Average marginal effects calculated as the mean marginal effects evaluated at the variables’ values in the sample. Standard errors derived with the delta-method are reported in parentheses. ****p < 0.001, **p < 0.01, *p < 0.05, +p < 0.10*.

Appendix J

Code for Matching and Computing the Instrument Variables

```
# coding: utf-8
import os
from datetime import datetime
import pandas as pd
import numpy as np
import csv
from operator import itemgetter

N_NEIGHBOR = 30

def compute_similarity_rated(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))

    simscore = gua_diff+prize_diff + weekend_diff  + contestsheld_diff+averagefb_diff +daytime_diff
    return simscore


def compute_similarity_high(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))

    simscore = gua_diff+prize_diff + weekend_diff + daytime_diff + contestsheld_diff+averagefb_diff
    return simscore

def compute_similarity_elim(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
```

```
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))

    simscore = gua_diff+prize_diff + weekend_diff + daytime_diff + contestsheld_diff+averagefb_diff
    return simscore

def compute_similarity_comm(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))
    comm_diff = (abs(contestday_dict[contestid1]['evercomm'] - contestday_dict[contestid2]['evercomm']))*100

    simscore = gua_diff+prize_diff + weekend_diff + daytime_diff + contestsheld_diff+averagefb_diff+comm_diff
    return simscore

def compute_similarity_negcomm(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))
    comm_diff =  (abs(contestday_dict[contestid1]['everncomm'] - contestday_dict[contestid2]['everncomm']))*100

    simscore = gua_diff+prize_diff + weekend_diff + daytime_diff + contestsheld_diff+averagefb_diff+comm_diff
    return simscore

def compute_similarity_highcomm(contestid1, contestid2, day):

    contest1_exp = 0
    if (contestday_dict[contestid1]['contestsHeld']>0): contest1_exp=1
    contest2_exp = 0
    if (contestday_dict[contestid2]['contestsHeld']>0): contest2_exp=1

    gua_diff = (abs(contestday_dict[contestid1]['guaranteed'] - contestday_dict[contestid2]['guaranteed']))*100
```

```
    prize_diff = (abs(contestday_dict[contestid1]['prize'] - contestday_dict[contestid2]['prize']))/10
    weekend_diff = (abs(contestday_dict[contestid1][day]['weekend'] - contestday_dict[contestid2][day]['weekend']))*100
    daytime_diff = (abs(contestday_dict[contestid1][day]['contest_daytime'] - contestday_dict[contestid2][day]['contest_daytime']))*100
    contestsheld_diff = (abs(contest1_exp - contest2_exp))*100
    averagefb_diff = (abs(contestday_dict[contestid1]['averageFeedback'] - contestday_dict[contestid2]['averageFeedback']))
    comm_diff =  (abs(contestday_dict[contestid1]['everhcomm'] - contestday_dict[contestid2]['everhcomm']))*100

    simscore = gua_diff+prize_diff + weekend_diff + daytime_diff + contestsheld_diff+averagefb_diff+comm_diff
    return simscore

def match_two_rows_rated(contestid1, contestid2, day):

    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1

    if abs(row['delay'] - x['delay']) == 0:
        simscore = compute_similarity_rated(contestid1, contestid2, day)

    return simscore


def match_two_rows_high(contestid1, contestid2, day):

    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1
    if contestday_dict[contestid2][day]['cumRated2Ystd'] == 0: return -1

    if abs(row['delay'] - x['delay']) == 0:
        simscore = compute_similarity_high(contestid1, contestid2, day)
    return simscore

def match_two_rows_elim(contestid1, contestid2, day):

    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1
    if contestday_dict[contestid2][day]['cumRated2Ystd'] == 0: return -1

    if abs(row['delay'] - x['delay']) == 0:
        simscore = compute_similarity_elim(contestid1, contestid2, day)

    return simscore

def match_two_rows_comm(contestid1, contestid2, day):
```

```
    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1

    if row['evercomm'] == x['evercomm']:
        simscore = compute_similarity_comm(contestid1, contestid2, day)

    return simscore

def match_two_rows_negcomm(contestid1, contestid2, day):

    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1

    if row['evercomm'] == x['evercomm']:
        simscore = compute_similarity_negcomm(contestid1, contestid2, day)

    return simscore

def match_two_rows_highcomm(contestid1, contestid2, day):

    simscore = -1

    row=contestday_dict[contestid1]
    x=contestday_dict[contestid2]

    if (day not in contestday_dict[contestid2]): return -1

    if row['evercomm'] == x['evercomm']:
        simscore = compute_similarity_highcomm(contestid1, contestid2, day)

    return simscore

def find_sim_cases_rated(contestid, day):

    similarcases={'cumSimRatedYstd':0,'simcount':0}

    cases_matched =[]

    for key in contestday_dict:

        contestid1 = contestid
        contestid2 = key

        if contestid1 == contestid2: continue

        simscore = match_two_rows_rated(contestid1, contestid2, day)
```

```
        if simscore >= 0:
          cumSimRatedYstd= contestday_dict[contestid2][day]['cumRated2Ystd']
          cases_matched.append([simscore, cumSimRatedYstd])

    # Pick the N nearest neighbors
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))

    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
      index = num_cases
      total  = 0
      while index > 0:
        index -= 1
        total += cases_matched_sorted[index][1]

      similarcases['cumSimRatedYstd'] = total/num_cases
      similarcases['simcount']=len(cases_matched_sorted)
    return similarcases

def find_sim_cases_high(contestid, day):

    similarcases={'cumSimHighYstd_prop':0,'simcount':0}

    cases_matched =[]


    for key in contestday_dict:

        contestid1 = contestid
        contestid2 = key

        if contestid1 == contestid2: continue

        simscore = -1
        simscore = match_two_rows_high(contestid1, contestid2, day)

        if simscore >= 0:
          cumSimHighYstd_prop=
contestday_dict[contestid2][day]['cumHigh2Ystd']/contestday_dict[contestid2][day]['cumRated2Ystd']
          cases_matched.append([simscore, cumSimHighYstd_prop])

    # Pick the N nearest neighbors
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))

    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
      index = num_cases
      total  = 0
      while index > 0:
        index -= 1
        total += cases_matched_sorted[index][1]

      similarcases['cumSimHighYstd_prop'] = total/num_cases
      similarcases['simcount']=len(cases_matched_sorted)

    return similarcases
```

```
def find_sim_cases_elim(contestid, day):

    similarcases={'cumSimElimYstd_prop':0,'simcount':0}

    cases_matched =[]

    for key in contestday_dict:

            contestid1 = contestid
            contestid2 = key

            if contestid1 == contestid2: continue

            simscore = match_two_rows_elim(contestid1, contestid2, day)

            if simscore >= 0:
                cumSimElimYstd_prop=
contestday_dict[contestid2][day]['cumElim2Ystd']/contestday_dict[contestid2][day]['cumRated2Ystd']
                cases_matched.append([simscore, cumSimElimYstd_prop])

    # Pick the N nearest neighbors
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))

    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
        index = num_cases
        total  = 0
        while index > 0:
            index -= 1
            total += cases_matched_sorted[index][1]

        similarcases['cumSimElimYstd_prop'] = total/num_cases
        similarcases['simcount']=len(cases_matched_sorted)

    return similarcases

def find_sim_cases_comm(contestid, day):

    similarcases={'cumSimCommentsYstd':0,'simcount':0}

    cases_matched =[]

    for key in contestday_dict:

            contestid1 = contestid
            contestid2 = key

            if contestid1 == contestid2: continue

            simscore = match_two_rows_comm(contestid1, contestid2, day)

            if simscore >= 0:
                cumSimCommYstd = contestday_dict[contestid2][day]['cumHolderCommentsYstd']
                cases_matched.append([simscore, cumSimCommYstd])

    # Pick the N nearest neighbors
```

```
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))

    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
        index = num_cases
        total  = 0
        while index > 0:
            index -= 1
            total += cases_matched_sorted[index][1]

        similarcases['cumSimCommentsYstd'] = total/num_cases
        similarcases['simcount']=len(cases_matched_sorted)

    return similarcases

def find_sim_cases_negcomm(contestid, day):

    similarcases={'cumSimNegCommYstd':0,'simcount':0}

    cases_matched =[]

    for key in contestday_dict:

            contestid1 = contestid
            contestid2 = key

            if contestid1 == contestid2: continue

            simscore = match_two_rows_negcomm(contestid1, contestid2, day)

            if simscore >= 0:
                cumSimNegCommYstd = contestday_dict[contestid2][day]['cumNegCommYstd']
                cases_matched.append([simscore, cumSimNegCommYstd])

    # Pick the N nearest neighbors
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))


    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
        index = num_cases
        total  = 0
        while index > 0:
            index -= 1
            total += cases_matched_sorted[index][1]

        similarcases['cumSimNegCommYstd'] = total/num_cases
        similarcases['simcount']=len(cases_matched_sorted)

    return similarcases

def find_sim_cases_highcomm(contestid, day):

    similarcases={'cumSimHighCommYstd':0,'simcount':0}

    cases_matched =[]
```

```
    for key in contestday_dict:

        contestid1 = contestid
        contestid2 = key

        if contestid1 == contestid2: continue

        simscore = match_two_rows_highcomm(contestid1, contestid2, day)

        if simscore >= 0:
            cumSimHighCommYstd = contestday_dict[contestid2][day]['cumHighCommYstd']
            cases_matched.append([simscore, cumSimHighCommYstd])

    # Pick the N nearest neighbors
    cases_matched_sorted = sorted(cases_matched, key=itemgetter(0))


    num_cases = min([N_NEIGHBOR, len(cases_matched_sorted)])
    if num_cases > 0:
        index = num_cases
        total  = 0
        while index > 0:
            index -= 1
            total += cases_matched_sorted[index][1]

        similarcases['cumSimHighCommYstd'] = total/num_cases
        similarcases['simcount']=len(cases_matched_sorted)

    return similarcases


def match_rated(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimRatedYstd'] = 0
    row['simcount']  = 0

    similarcases=find_sim_cases_rated(contestid, day)

    if similarcases['simcount'] > 0:
        row['cumSimRatedYstd']      = similarcases['cumSimRatedYstd']
        row['simcount'] = similarcases['simcount']
    return row

def match_high(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimHighYstd'] = 0
```

```
    row['simcount']  = 0

    similarcases=find_sim_cases_high(contestid, day)

    if similarcases['simcount'] > 0:
        row['cumSimHighYstd']      = similarcases['cumSimHighYstd_prop'] * row['cumrated2ystd']
        row['simcount'] = similarcases['simcount']

    return row


def match_elim(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimElimYstd'] = 0
    row['simcount']  = 0

    similarcases=find_sim_cases_elim(contestid, day)

    if similarcases['simcount'] > 0:
        row['cumSimElimYstd']      = similarcases['cumSimElimYstd_prop'] * row['cumrated2ystd']
        row['simcount'] = similarcases['simcount']

    return row

def match_comm(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimCommentsYstd'] = 0
    row['simcount']  = 0

    similarcases=find_sim_cases_comm(contestid, day)

    if similarcases['simcount'] > 0:
        row['cumSimCommentsYstd']      = similarcases['cumSimCommentsYstd']
        row['simcount'] = similarcases['simcount']

    return row

def match_negcomm(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimNegCommYstd'] = 0
    row['simcount']  = 0
```

```
    if row['cumholdercommentsystd'] == 0:
        return row

    similarcases=find_sim_cases_negcomm(contestid, day)


    if similarcases['simcount'] > 0:
        row['cumSimNegCommYstd']      = similarcases['cumSimNegCommYstd']
        row['simcount'] = similarcases['simcount']

    return row

def match_highcomm(row):

    contestid = row['contestID']
    day = row['day']

    similarcases={}

    row['cumSimHighCommYstd'] = 0
    row['simcount']  = 0

    if row['cumholdercommentsystd'] == 0:
        return row

    similarcases=find_sim_cases_highcomm(contestid, day)

    if similarcases['simcount'] > 0:
        row['cumSimHighCommYstd']      = similarcases['cumSimHighCommYstd']
        row['simcount'] = similarcases['simcount']

    return row
```

References

Abadie, A.,and Imbens, G. W. 2006. "Large Sample Properties of Matching Estimators for Average Treatment Effects," *Econometrica* (74:1), pp. 235-267.

Abadie, A.,and Imbens, G. W. 2011. "Bias-Corrected Matching Estimators for Average Treatment Effects," *Journal of Business and Economic Statistics* (29:1), pp. 1-11.

Abadie, A.,and Imbens, G. W. 2016. "Matching on the Estimated Propensity Score," *Econometrica* (84:2), pp. 781-807.

Jiang, Z. Z., Huang, Y., and Beil, D. R. 2016. "The Role of Feedback in Dynamic Crowdsourcing Contests: A Structural Empirical Analysis," Ross School of Business Paper No. 1334, University of Michigan.

Wooten, J. O., and Ulrich, K. T. 2016. "Idea Generation and the Role of Feedback: Evidence from Field Experiments with Innovation Tournaments," *Production and Operations Management* (26:1), pp. 80-99.

Yang, Y., Chen, P.-Y., and Pavlou, P. A. 2013. "Managing Open Innovation Contests in Online Market," Working Paper, Temple University, Philadelphia, PA.