Detecting Phony Reviewers by Analyzing Trust in Amazon User Reviews

Presentation by Robert Yates

Joint Research by Dylan Shinzaki, Kate Stuckman, and Robert Yates

Problem Statement and Background

Fraud

- Amazon has over 34 million reviews
- Most are helpful, but there are also fake reviews written to skew product ratings
  - book authors may have fake reviews say how great their book is
  - manufacturers may write to fake reviews trashing a competitor’s product
- Users vote reviews as either helpful or unhelpful, but prior ratings on a product can influence their decision
- Problem Statement: How do we tell which reviews are authentic and identify fraudulent reviewers?

Outline

- Background
  - Problem Statement
  - Prior Work
  - Review Graph Paper
- Our Contribution
  - Analysis of Amazon Dataset
  - Solution to Detection of Phony Users
  - Two Methods of Computing User Helpfulness from Votes
  - Algorithm to Compute User Trust
  - Compare Helpfulness and Trust
- Additions
  - Algorithm Improvement (K Threshold Approximation)
  - Evaluation Using Bots

Prior work

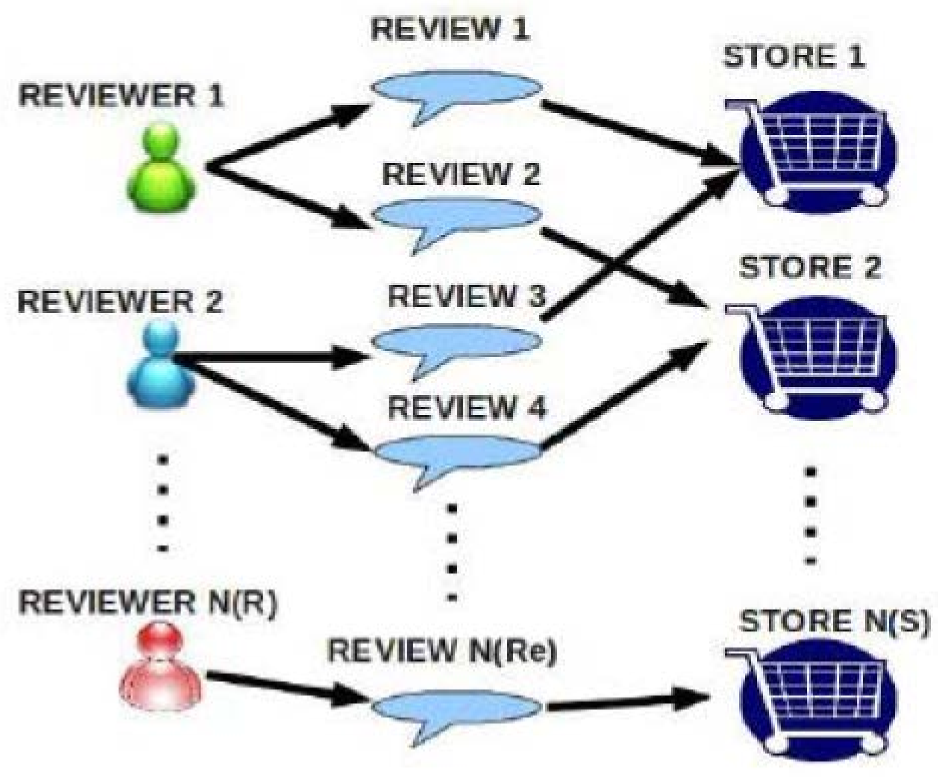
- Previous ratings can influence users when they vote on review helpfulness
- If high variance in review star ratings (reviewers disagree)
  - Then polarizing reviews (1 or 5 stars) are voted as most helpful
- If low variance in review star ratings (reviewers agree)
  - Then average reviews (near product mean) are voted as most helpful
- C. Danescu-Niculescu-Mizil ets al. “How opinions are received by online communities”[1]

Review Graph based Online Store Review Spammer Detection

Review Graph based Online Store Review Spammer Detection

- First time using review graph to find spammers
- Purely graph analysis, complementary to textual analysis spam detection
- Stores tend to provide the same basic services (i.e. sell products), but
  - High quality stores tend to get higher product reviews and fewer fraudulent users
  - Lower quality stores get lower product reviews from genuine users and high product reviews from fraudulent users promoting the store
- G. Wang et al. “Review graph based online store review spammer detection”[2]

The Review Graph



Algorithm and Terminology

- Analyze average rating of reviews for a single store
- A store is more reliable if it has more positive reviews from trustworthy reviewers
- A reviewer is trustworthy if he/she wrote many honest reviews
- A review is honest if it is supported by many other honest reviews
- Use iterative method to compute the three concepts using the review graph

Results

- Agreement of human judges to determine if spammer candidates are really spammers
  - 100 spammer candidates identified by algorithm
  - Evaluator 1 identified 49 suspicious reviewers, out of which 33 were recognized by Evaluator 2 and 37 were caught by Evaluator 3.
  - Agreement among evaluators is 60%

	Evaluator 1	Evaluator 2	Evaluator 3
Evaluator 1	**49**	33	37
Evaluator 2		**35**	23
Evaluator 3			**40**

Conclusion

- Even though humans can’t confirm reviewers were fraudulent, they still could be spammers
- Existing methods focus on identifying duplicate results in text content
- This algorithm finds spammer candidates previous methods have overlooked
- We detected spamming behavior and confirmed with human agreement

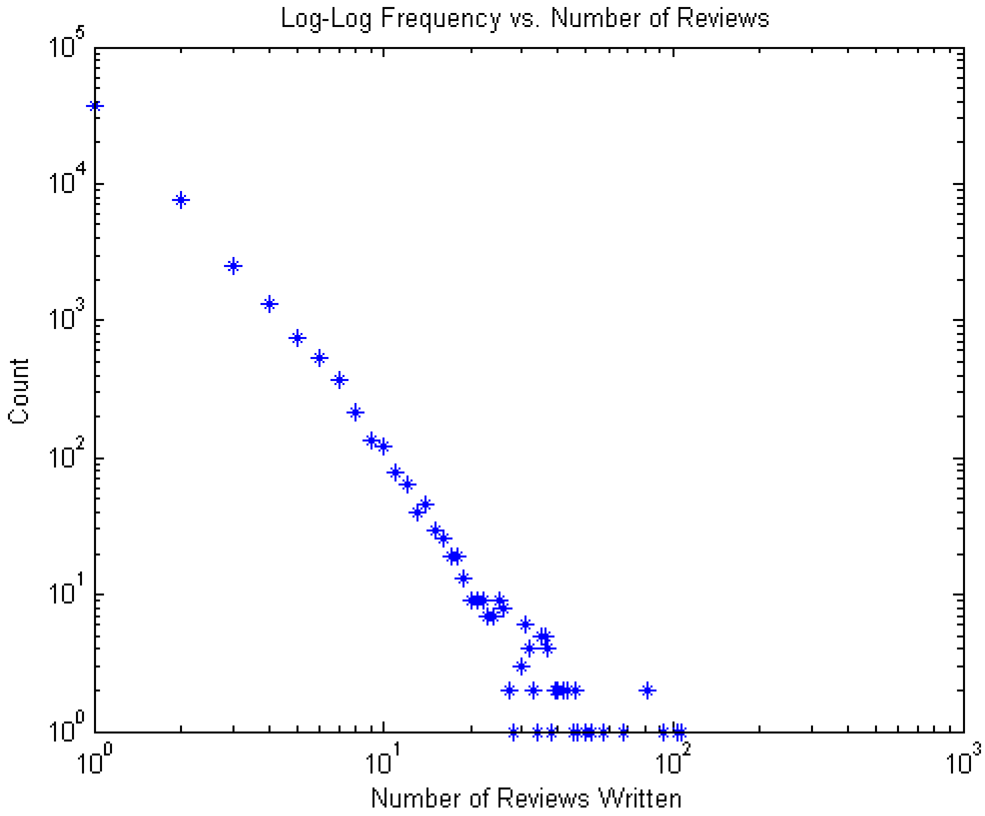
Our Contribution

Analysis of Amazon Dataset

Dataset

- We studied the Foods subset of an Amazon review dataset
  - The full dataset contains 34,686,770 reviews
  - The Foods subset contains over 500,000 reviews
  - The number of reviews per user follows a power law distribution

Dataset (Powerlaw)



Solution to Detection of Phony Users

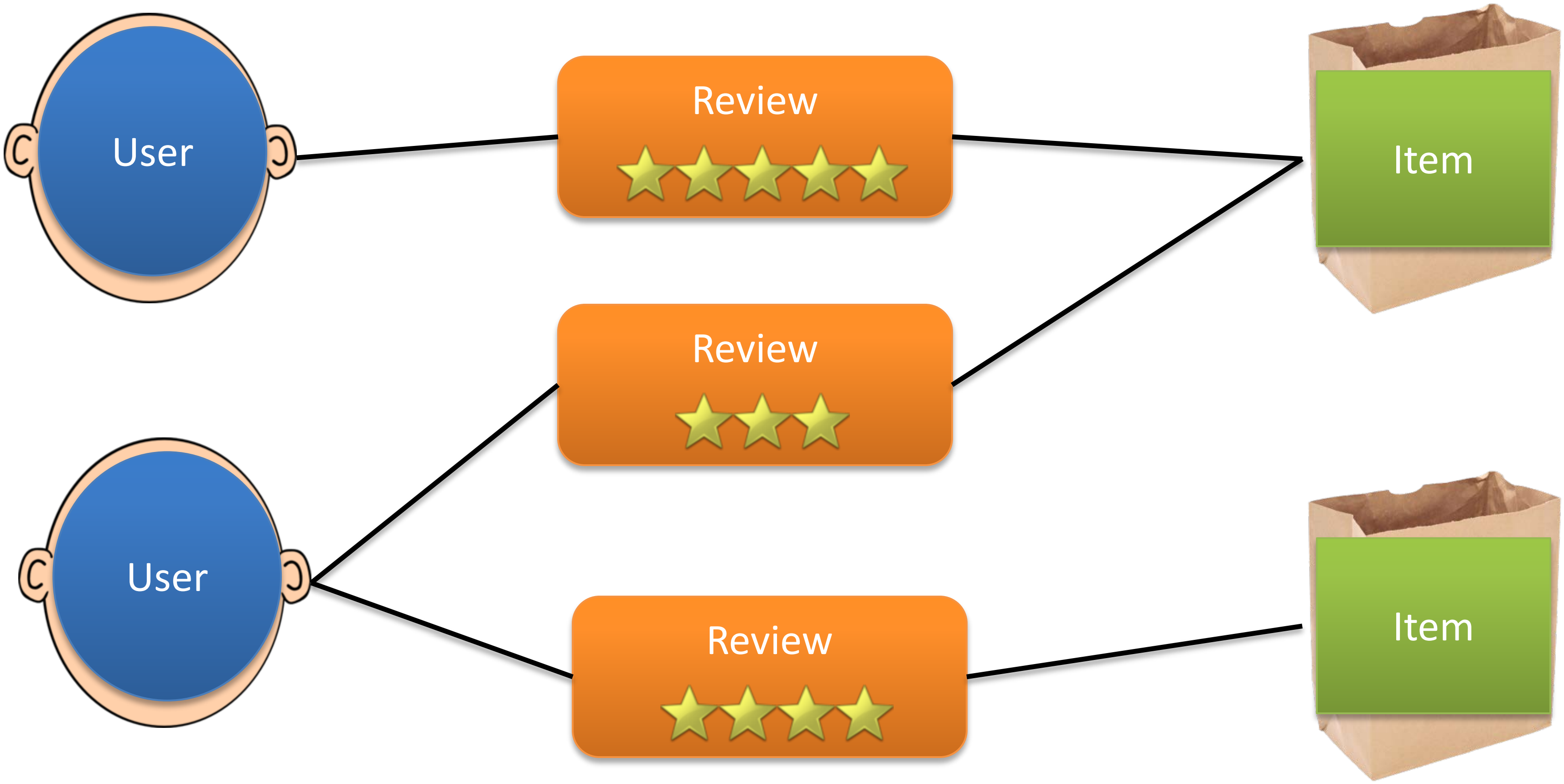
Solution

- We create two methods to compute user helpfulness
- We implement the algorithm proposed in “Review graph based online store review spammer detection” to better determine trustworthiness of reviewers
- Modify algorithm for our purposes to use a single store
- Run algorithm and compare user trust with user helpfulness votes

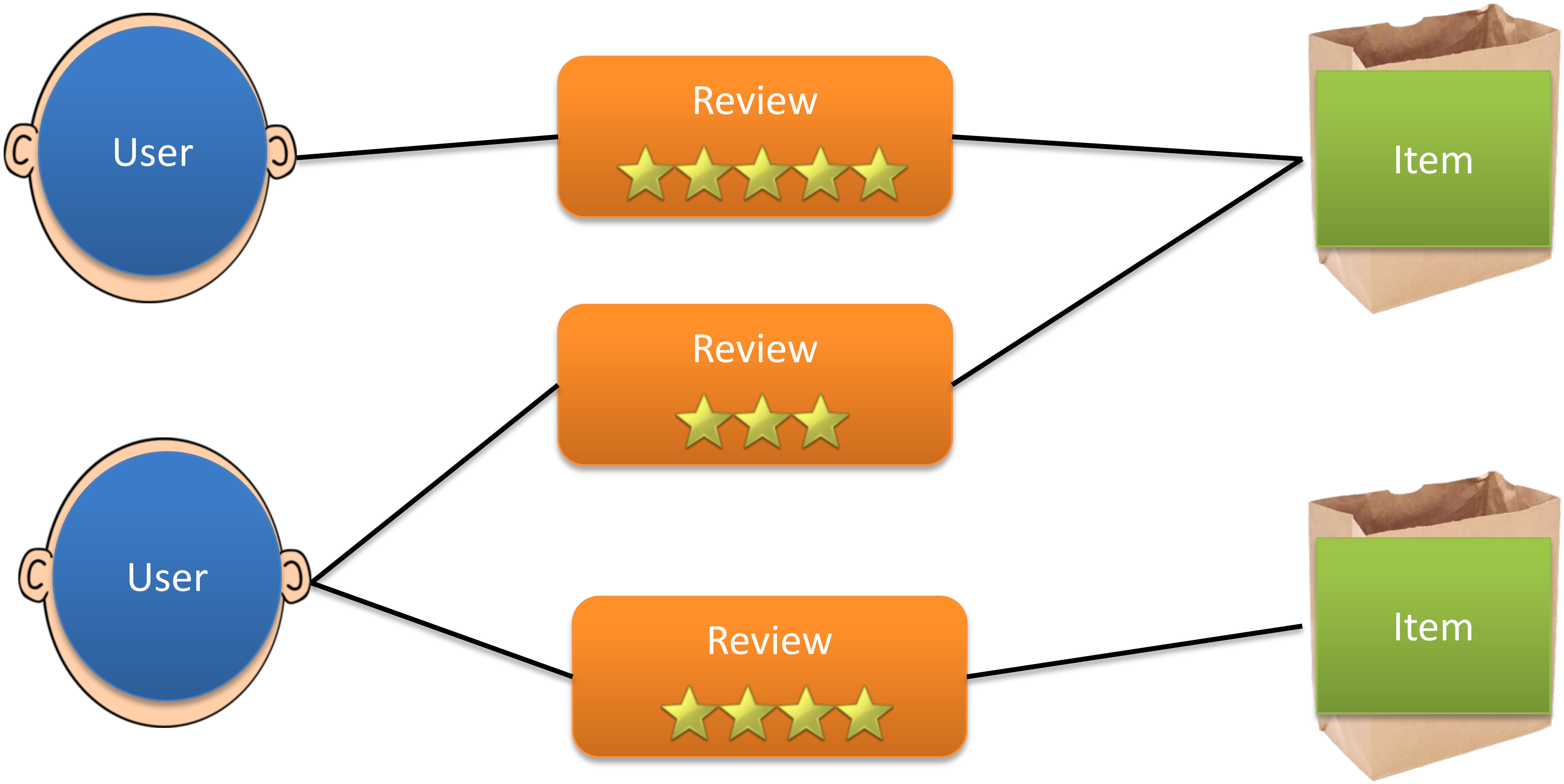
Solution

- We also improve the efficiency of the algorithm by removing users who have written few reviews (K Threshold Approximation)
- Finally, we evaluate the effectiveness of the algorithm by inserting additional reviewers that model various user behaviors
- All analysis is completed without processing textual content in reviews

Review Graph – Our Solution



Review Graph – Our Solution



Many-to-many relationship

Two Methods of Computing User Helpfulness from Votes

Computing User Helpfulness

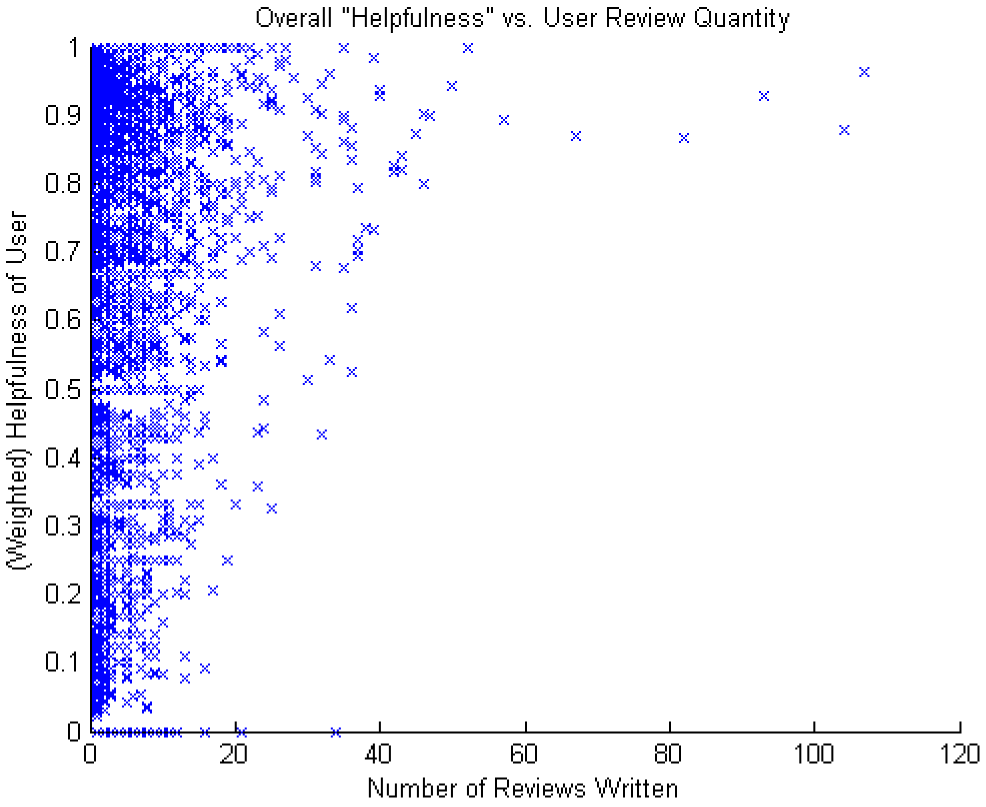
- Method 1:

$$\frac{\text{sum of helpful votes}}{\text{total sum of helpful + unhelpful votes}}$$

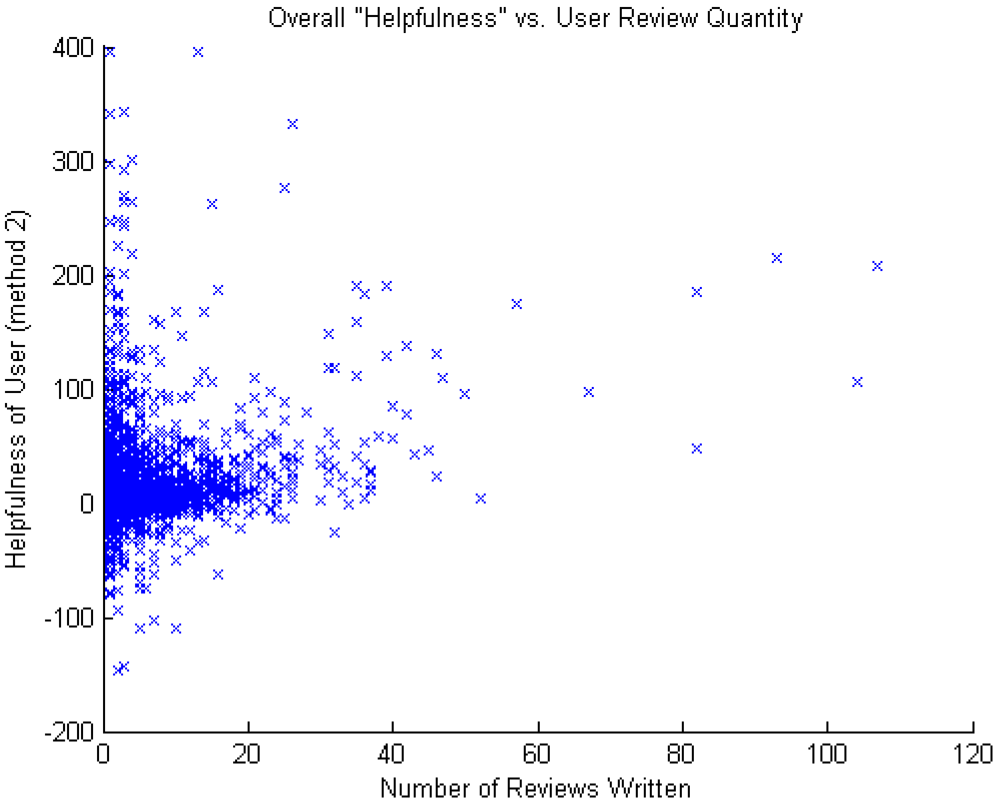
- Method 2:

$$\frac{\text{sum of helpfulness votes – sum of unhelpful votes}}{\text{total sum of helpful + unhelpful votes}}$$

Method 1


Overall "Helpfulness" vs. User Review Quantity
(Weighted) Helpfulness of User
Number of Reviews Written
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120

Method 2


Overall "Helpfulness" vs. User Review Quantity
Helpfulness of User (method 2)
400
300
200
100
0
-100
-200
0
20
40
60
80
100
120
Number of Reviews Written

Algorithm to Compute User Trust and Other Attributes

Algorithm Terminology

1. User trust (-1 distrust to +1 trust)
   - Users are trusted if their reviews are honest, users are untrusted if their reviews are dishonest.
2. Item reliability (-1 unreliable to +1 reliable)
   - Items are reliable if they have high scores by trusted users, items are unreliable if they have low scores by trusted users.
3. Review honesty (-1 dishonest to +1 honest)
   - First compute review agreement: it is high when review agrees with majority trusted opinion. Review agreement is low when it disagrees with majority trusted opinion.
   - Second compute review honesty: Normalize review agreement and take into account item reliability.

Normalization Function

To normalize value x:

$$\frac{2}{1 + e^{-x}} -1$$

Examples:

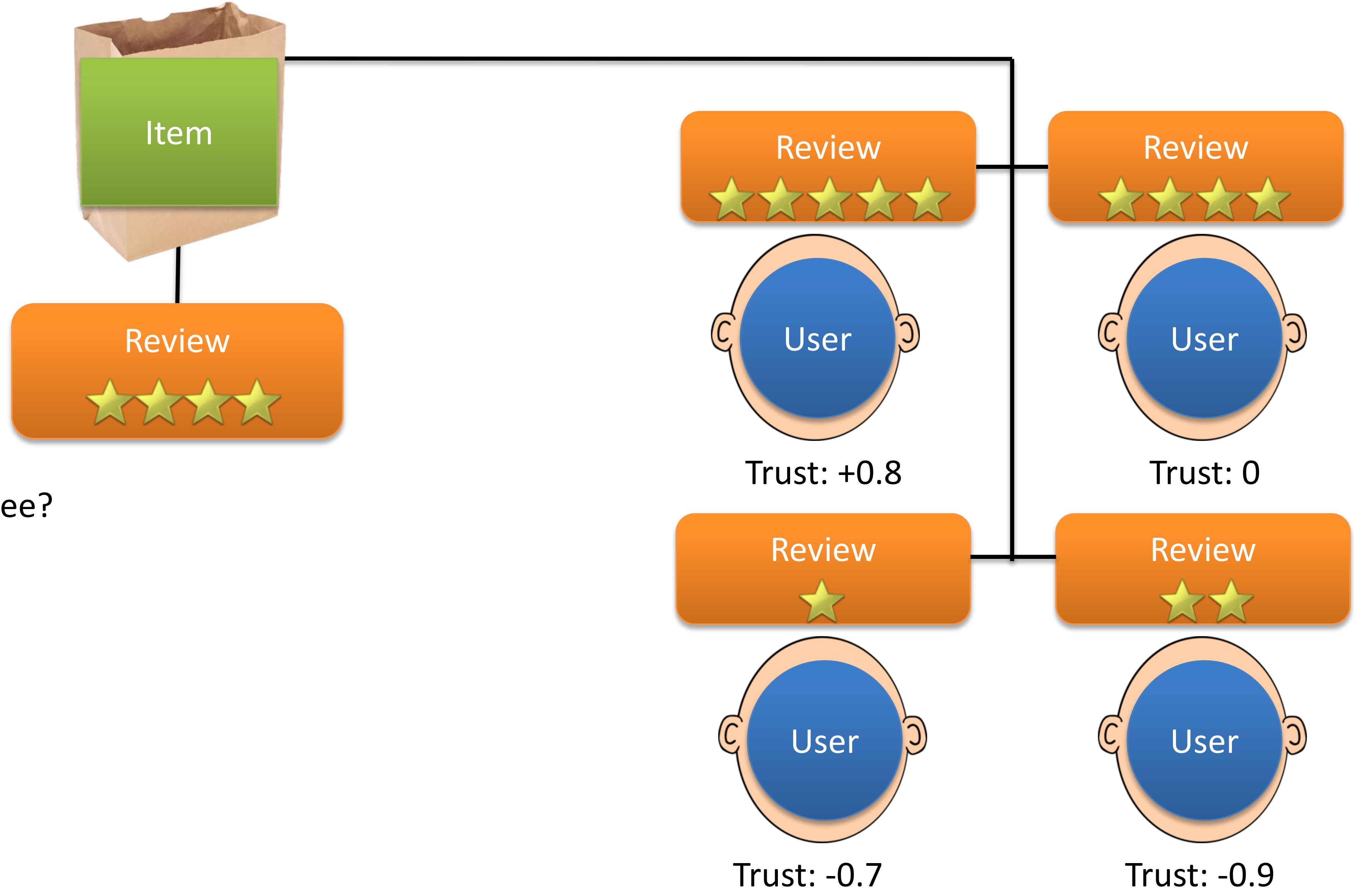
x = 10.0, Normalized(x) = 0.999

x = 2.4, Normalized(x) = 0.834

x = 1.8, Normalized(x) = 0.716

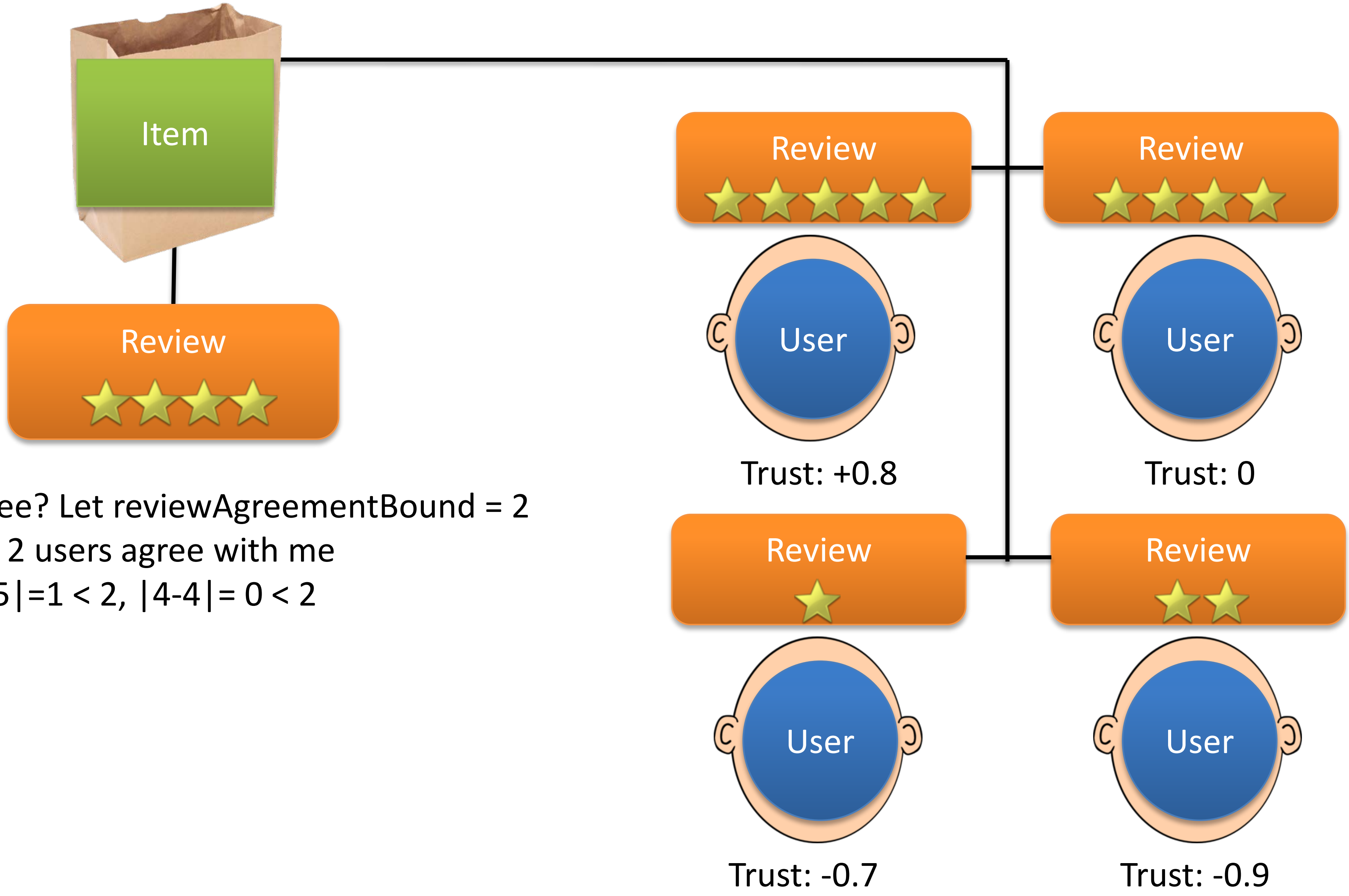
x = -1.1, Normalized(x) = -0.501

Computing Review Agreement



Agree?

Computing Review Agreement

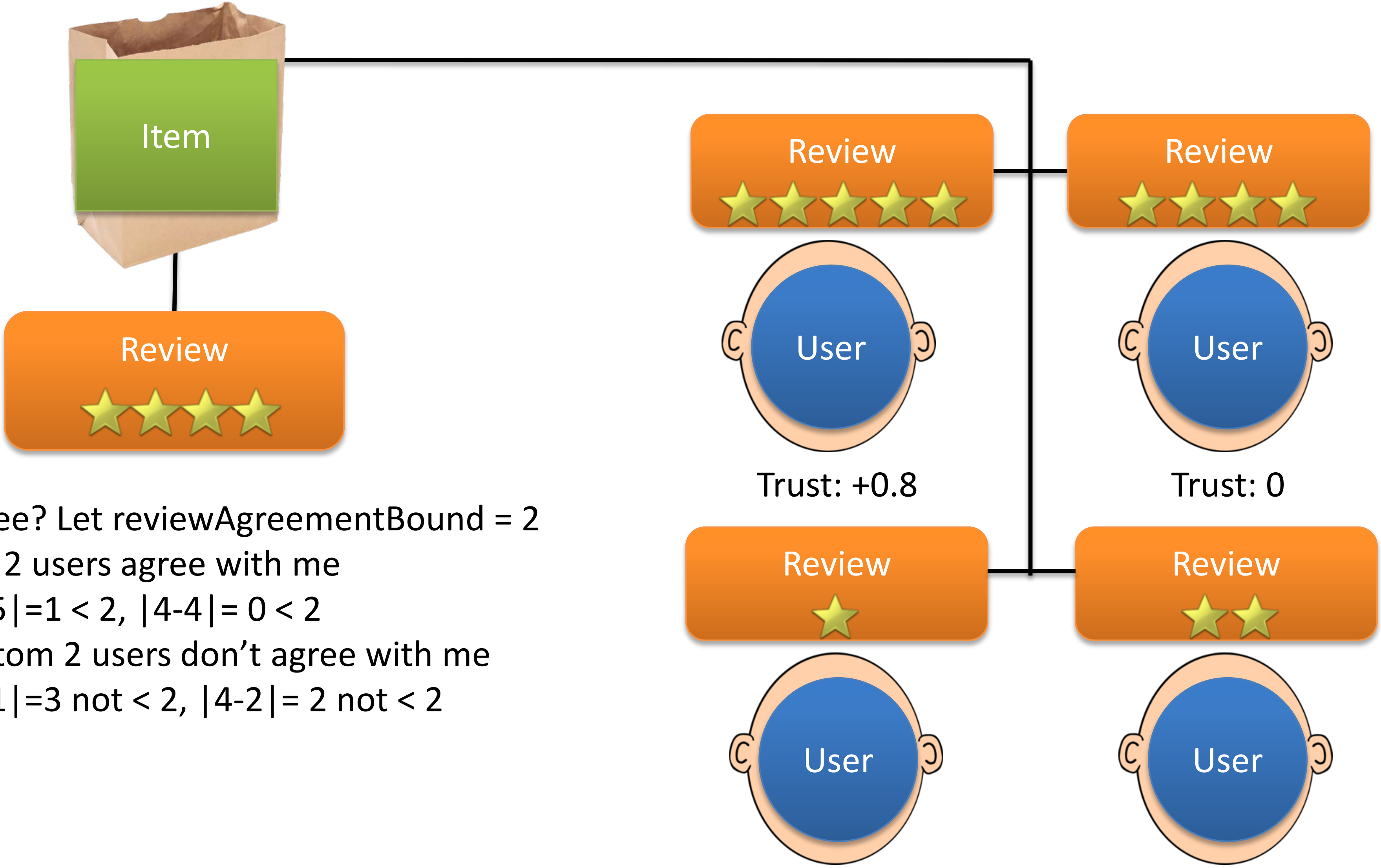


Agree? Let reviewAgreementBound = 2

Top 2 users agree with me

|4-5|=1 < 2, |4-4|= 0 < 2

Computing Review Agreement



Trust: +0.8

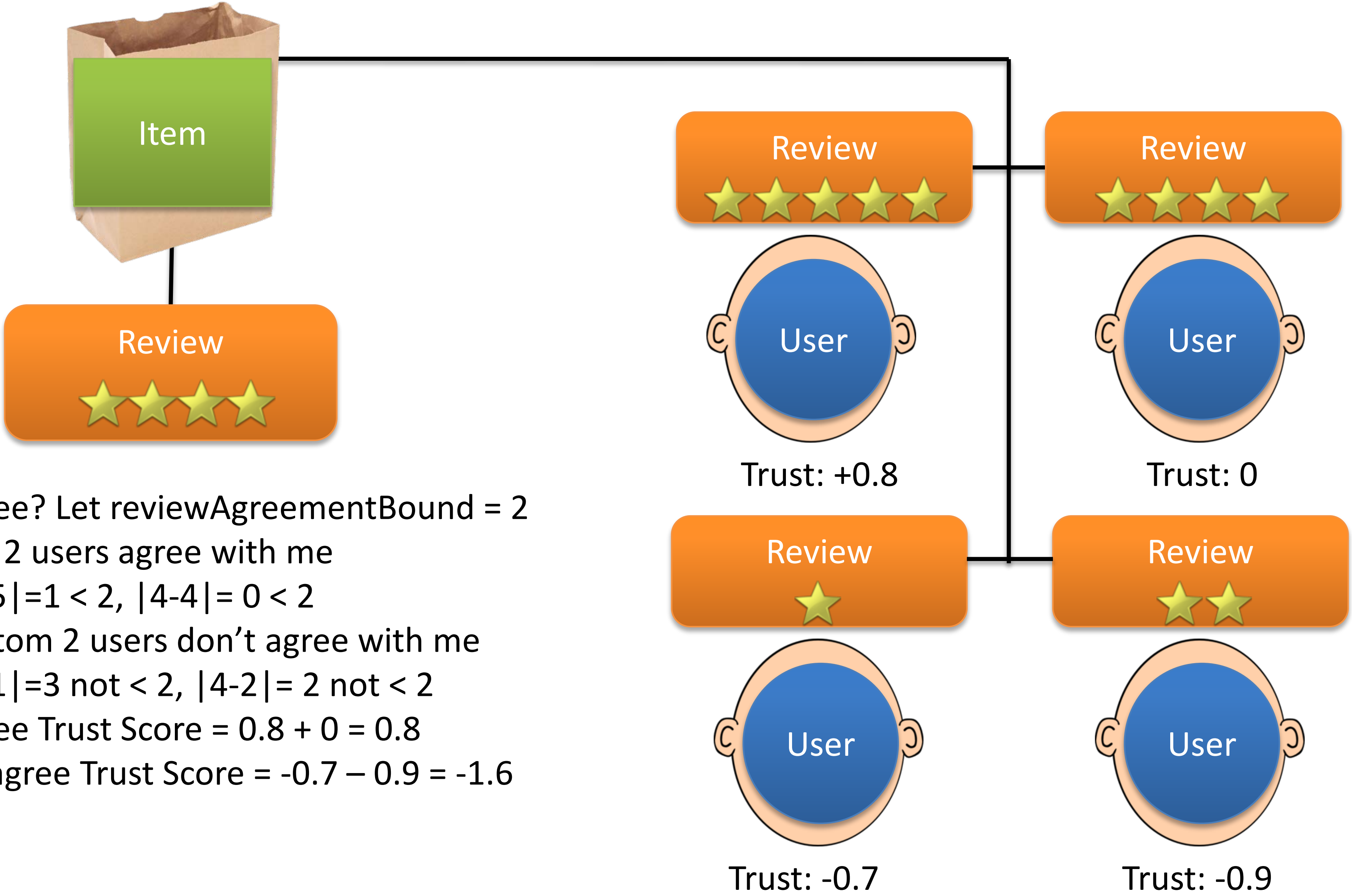
Trust: 0

Trust: -0.7

Trust: -0.9

Agree? Let reviewAgreementBound = 2
Top 2 users agree with me
|4-5|=1 < 2, |4-4|= 0 < 2
Bottom 2 users don’t agree with me
|4-1|=3 not < 2, |4-2|= 2 not < 2

Computing Review Agreement



Agree? Let reviewAgreementBound = 2

Top 2 users agree with me

|4-5|=1 < 2, |4-4|= 0 < 2

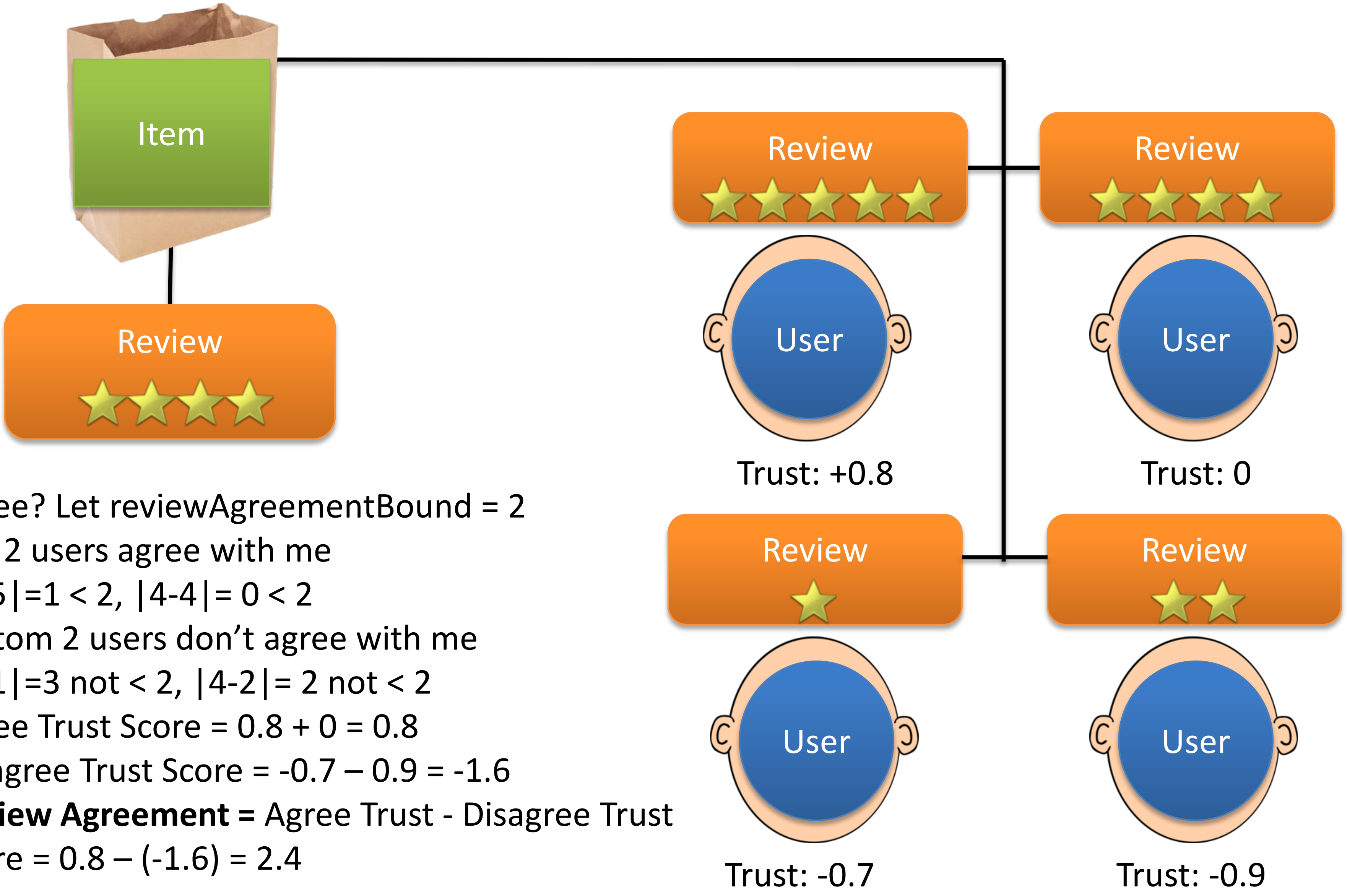
Bottom 2 users don't agree with me

|4-1|=3 not < 2, |4-2|= 2 not < 2

Agree Trust Score = 0.8 + 0 = 0.8

Disagree Trust Score = -0.7 – 0.9 = -1.6

Computing Review Agreement



Agree? Let reviewAgreementBound = 2

Top 2 users agree with me

|4-5|=1 < 2, |4-4|= 0 < 2

Bottom 2 users don’t agree with me

|4-1|=3 not < 2, |4-2|= 2 not < 2

Agree Trust Score = 0.8 + 0 = 0.8

Disagree Trust Score = -0.7 – 0.9 = -1.6

Review Agreement = Agree Trust - Disagree Trust

Score = 0.8 – (-1.6) = 2.4

Computing Review Honesty

Normalized(Review Agree) * |Item Reliability|

2.4 -> 0.834

High Reliability: 0.834 * |1| = 0.834

0.834 * |0.5| = 0.417

Medium Reliability: 0.834 * |0| = 0

0.834 * |-0.5| = 0.417

Low Reliability: 0.834 * |-1| = 0.834

Remember: Review agreement is high when review agrees with majority trusted opinion.
Review agreement is low when it disagrees with majority trusted opinion.

Computing Review Honesty

Normalized(Review Agree) * |Item Reliability|

2.4 -> 0.834

How much everyone **agrees** with my review

High Reliability:	0.834 * |1|	= 0.834
	0.834 * |0.5|	= 0.417
Medium Reliability:	0.834 * |0|	= 0
	0.834 * |-0.5|	= 0.417
Low Reliability:	0.834 * |-1|	= 0.834

Remember: Review agreement is high when review agrees with majority trusted opinion. Review agreement is low when it disagrees with majority trusted opinion.

Computing Review Honesty

Normalized(Review Agree) * |Item Reliability|

2.4 -> 0.834

High value = Very good/bad products
Low value = Middle of the road products

High Reliability: 0.834 * |1| = 0.834
0.834 * |0.5| = 0.417
Medium Reliability: 0.834 * |0| = 0
0.834 * |-0.5| = 0.417
Low Reliability: 0.834 * |-1| = 0.834

Computing Review Honesty

Normalized(Review Agree) * |Item Reliability|

2.4 -> 0.834

How honest we know my review is

High Reliability: 0.834 * |1| = 0.834

0.834 * |0.5| = 0.417

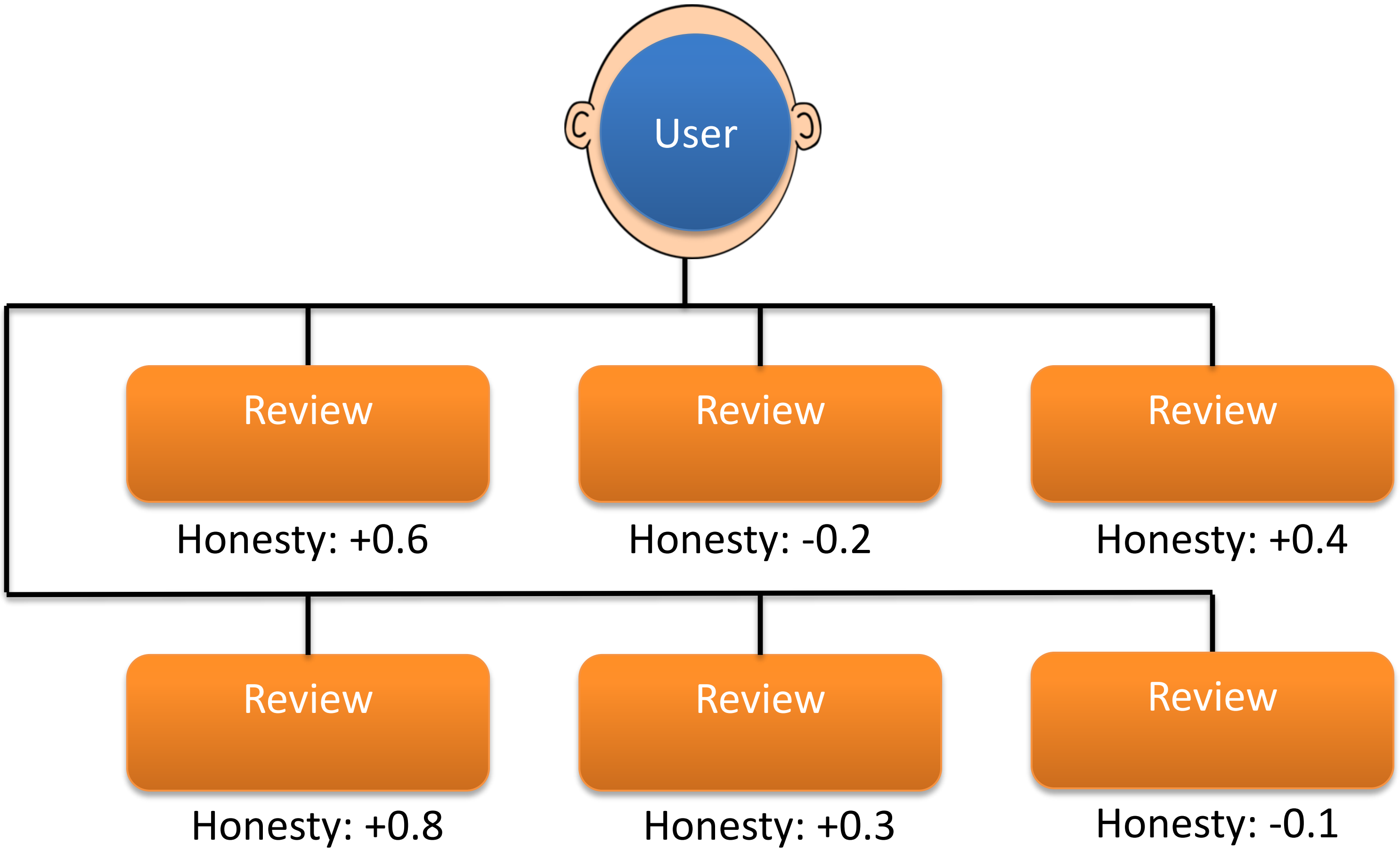
Medium Reliability: 0.834 * |0| = 0

0.834 * |-0.5| = 0.417

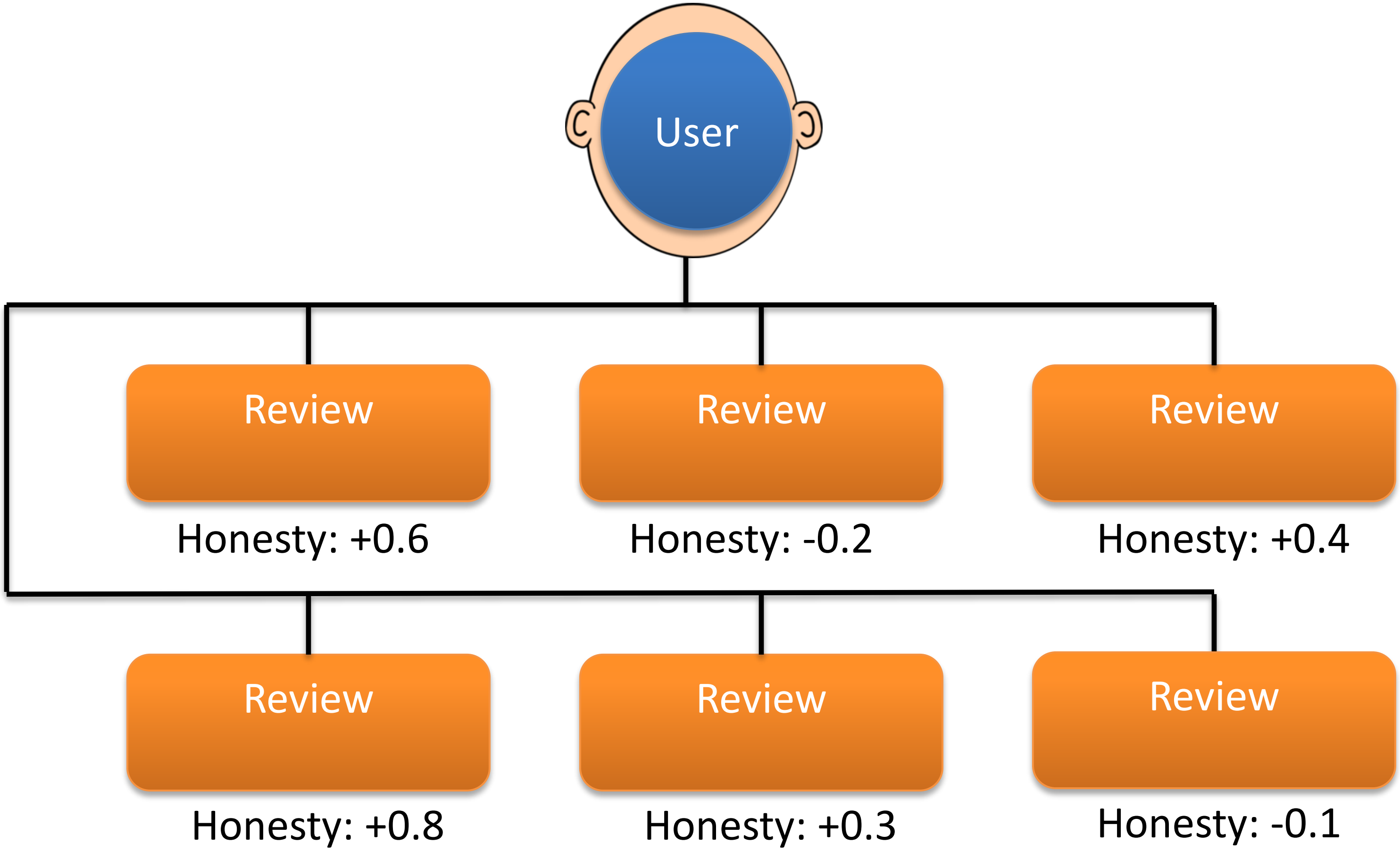
Low Reliability: 0.834 * |-1| = 0.834

Even if everyone disagrees with my review, if it is for a middle of the road product, we can NOT say that my review is dishonest for sure.

Computing User Trust



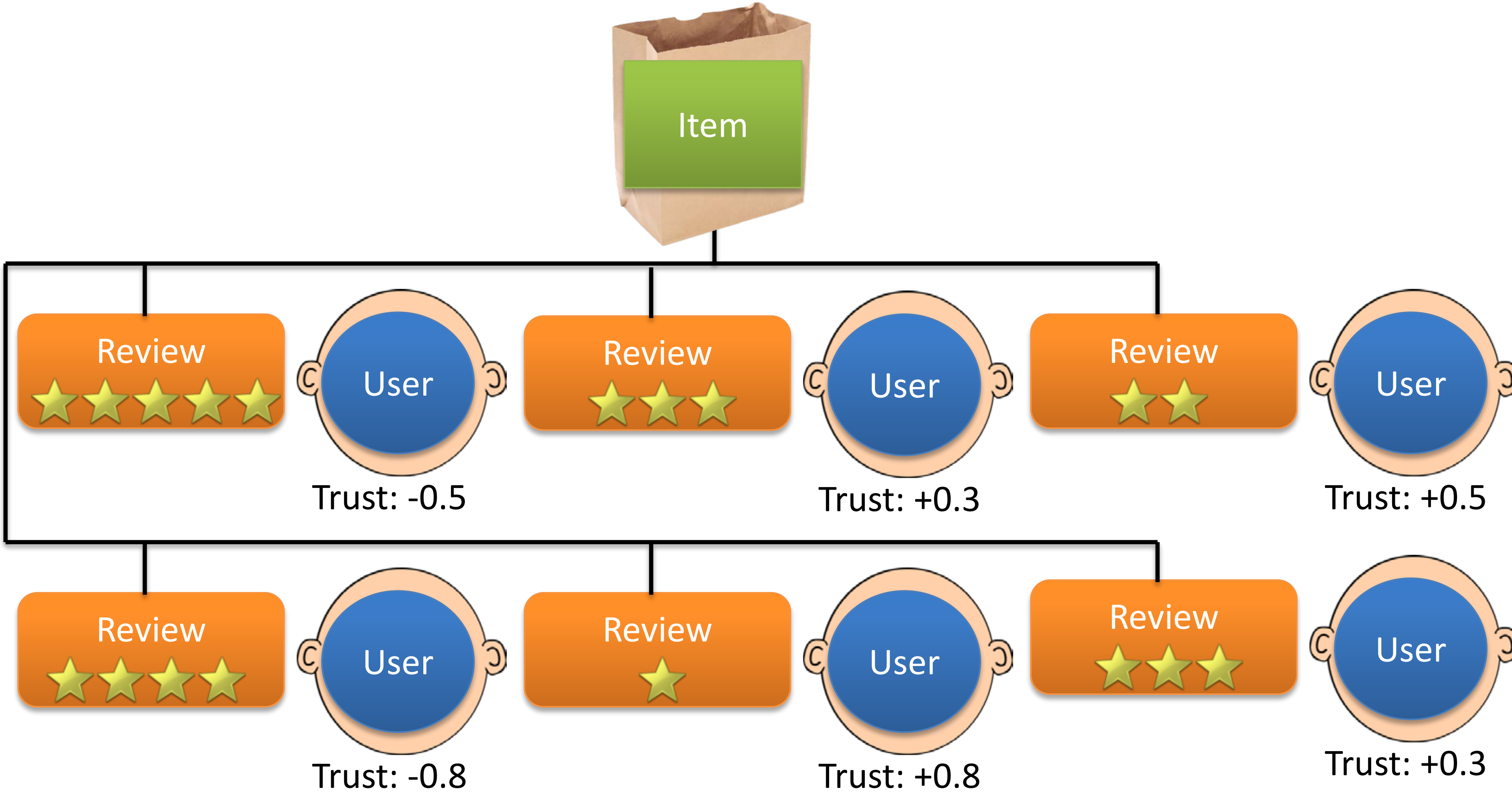
Computing User Trust



Total Trust: 0.6 – 0.2 + 0.4 + 0.8 + 0.3 – 0.1 = 1.8

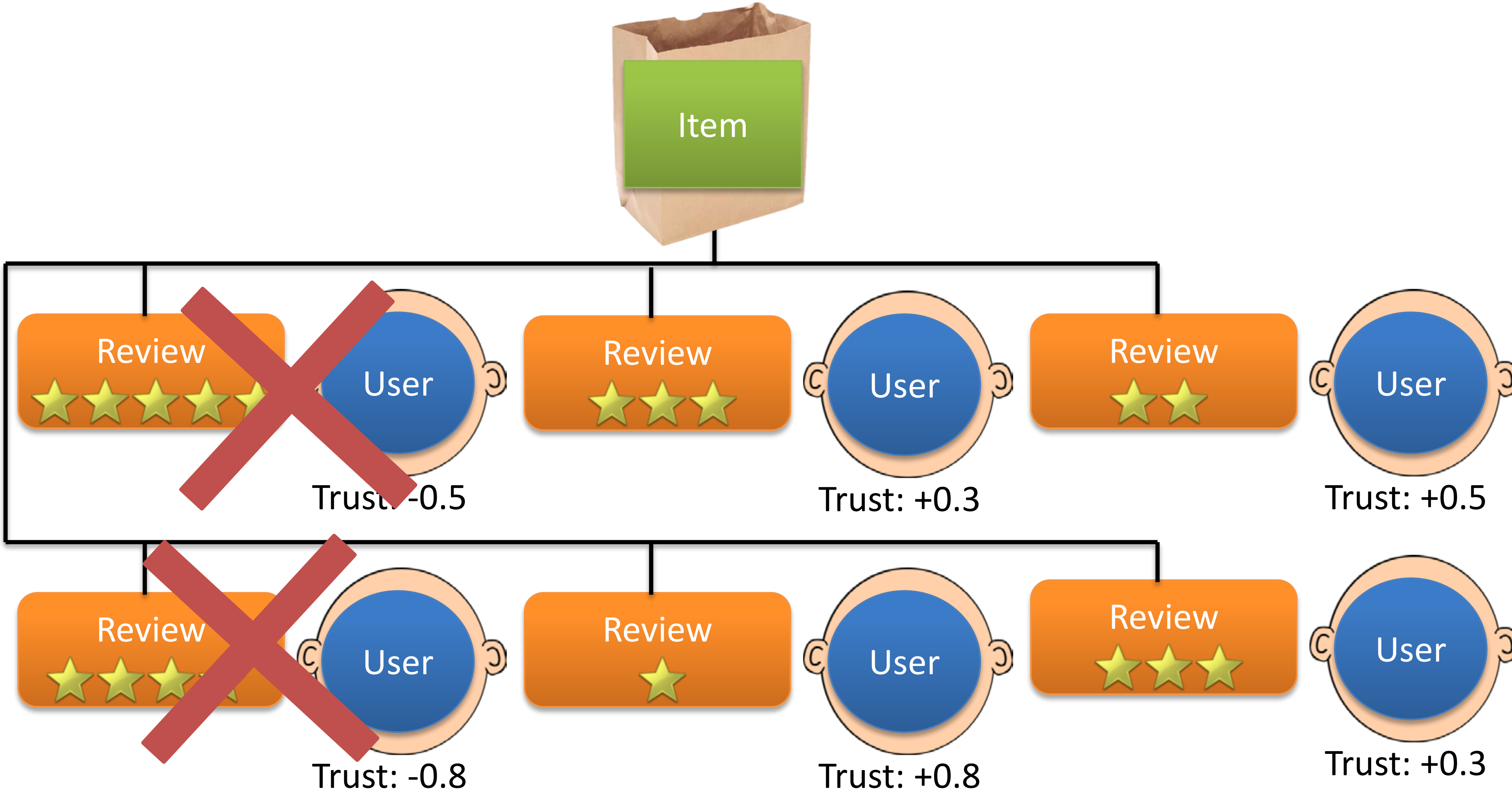
Normalized Trust: 1.8 -> 0.716

Computing Item Reliability



Trust must be > 0.

Computing Item Reliability



Trust must be > 0. Reliability: 0.3(3-3) + 0.5(2-3) + 0.8(1-3) + 0.3(3-3) = -0.5 – 1.6 = -2.1 -> -0.782

Algorithm Implementation

Algorithm

Initialize items' reliability to 1 *# assume all items are reliable*

Initialize users' trust to 1 *# assume all users are trusted*

Set reviewAgreementBound = 2

# two reviews agree if the absolute value of their difference is less than reviewAgreementBound

Algorithm

```
def computeReviewAgreement(): # Define function for review agreement
  for each item:
    for review per item:
      reviewsThatAgreeWithMe = []
      reviewsThatDisgreeWithMe = []
      for every other review for this item:
        if abs(review score – other review score) < reviewAgreementBound:
          reviewsAgreeWithMe.add(otherReview)
        else:
          reviewsDisgreeWithMe.add(otherReview)
      agreeTrust = sum of trusts for all users who wrote reviews that agree
      disagreeTrust = sum of trusts for all users who wrote reviews that disagree
      reviewAgree = agreeTrust – disagreeTrust

computeReviewAgreement() # Compute reviewAgree based on defaults
```

Algorithm

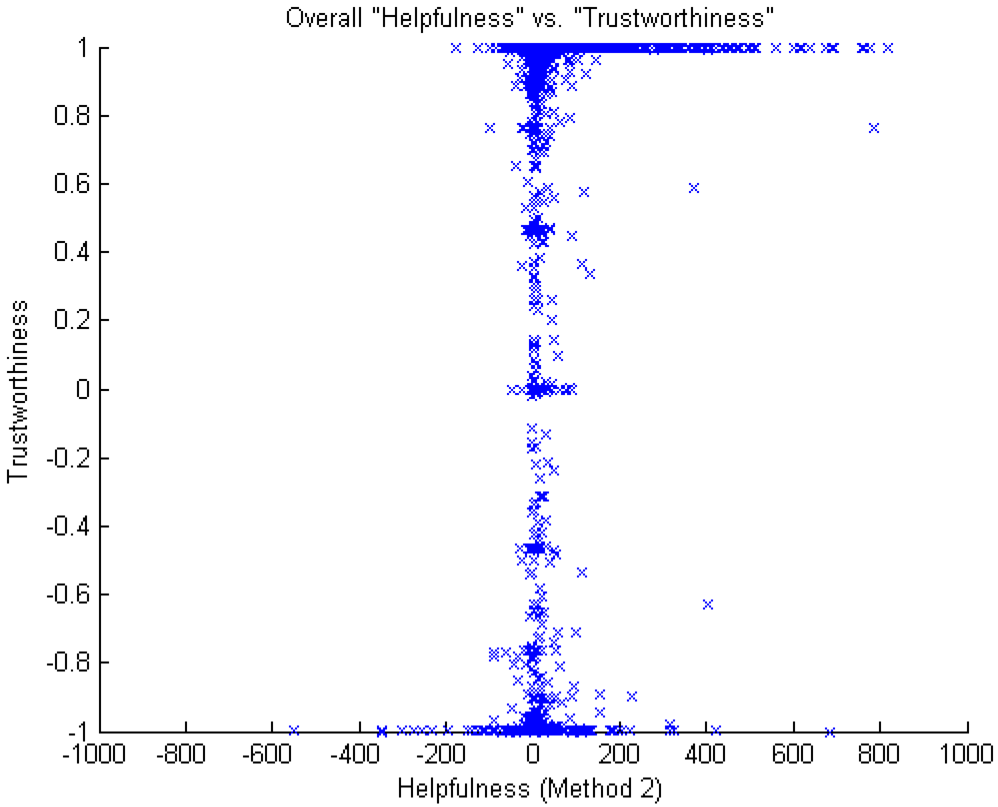
```
Repeat for a number of rounds or until convergence:
  # Reviews are honest if they are agreeable for highly reliable or highly unreliable items, reviews are dishonest if they are disagreeable for highly reliable or highly unreliable items.
  for each review:
    agreeNormalized = (2 / float(1. + math.exp(-reviewAgree))) - 1
    reviewHonesty = abs(itemReliability of item being reviewed) * agreeNormalized


  # Users are trusted if their reviews are honest, users are untrusted if their reviews are dishonest.
  for each user:
    trust = sum of reviewHonesty for all reviews this user wrote
    userTrust = (2 / float(1. + math.exp(-honesty))) – 1 # normalize trust
```

Algorithm

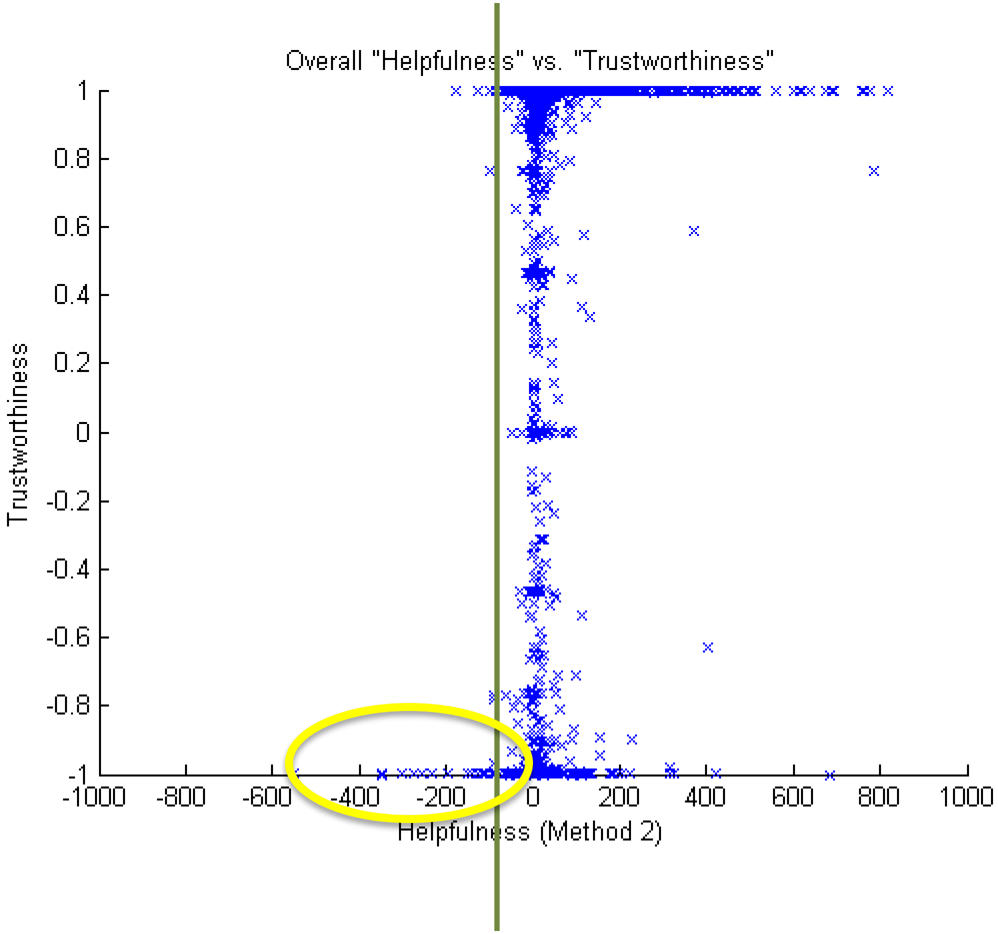
```
# Continuing in loop
 # Items are reliable if they have high scores by trusted users, items are unreliable if they have low scores by trusted users.
 for each item:
   reliability = 0
   for each review of this item:
     if the trust of user who wrote this review > 0: # trusted
       reliability += trust of user * (reviewScore  -  3)
         # 3 is middle review score
   itemReliability = (2 / float(1. + math.exp(-reliability))) – 1 # normalize reliability value

 # Reviews are agreeable if they agree with most reviews for the same item by trusted users, reviews are disagreeable if they disagree with most reviews for the same item by trusted users.
 computeReviewAgreement()
```

Compare Helpfulness and Trust

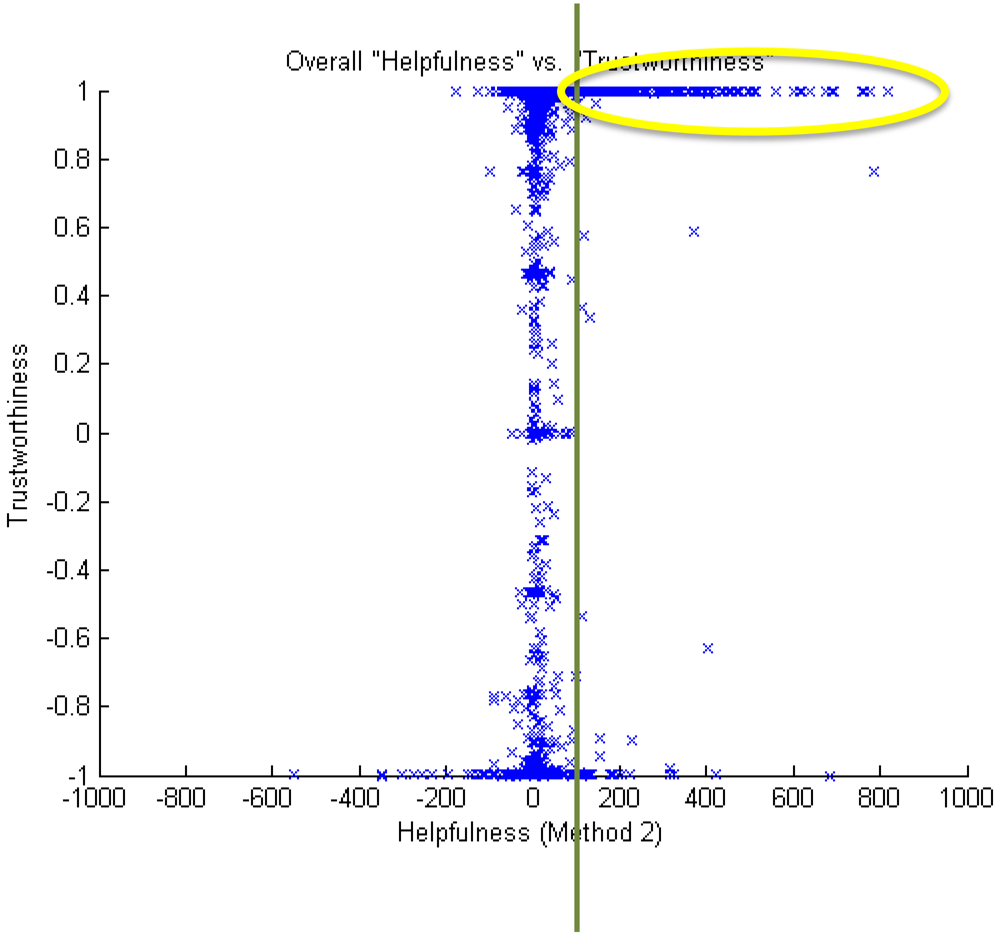
Relating Helpfulness and Trust



Relating Helpfulness and Trust



Relating Helpfulness and Trust



Relating Helpfulness and Trust

- Correlation is strongest Method 2 when only outlier values are considered.
- For example, 87% of users with helpfulness less than -100 have trust ratings less than -0.9.
- This could prove useful in spam detection.

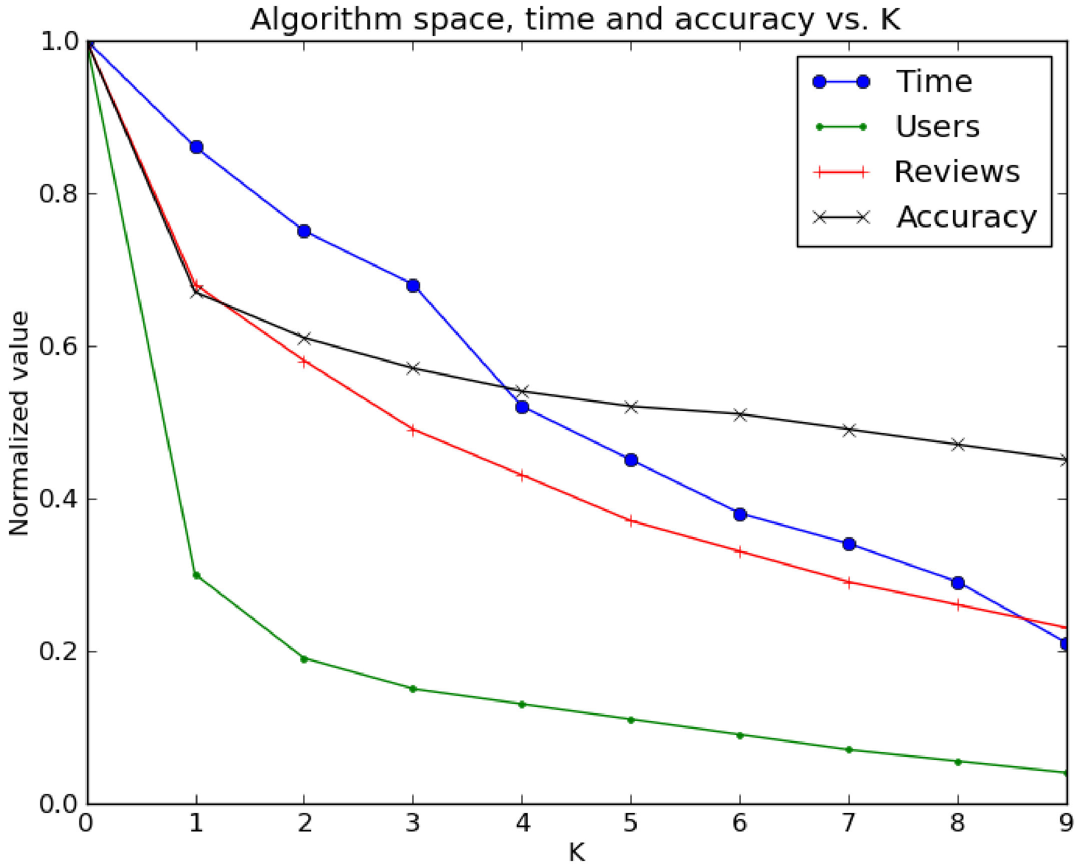
Additions

Algorithm Improvement using K Threshold Approximation

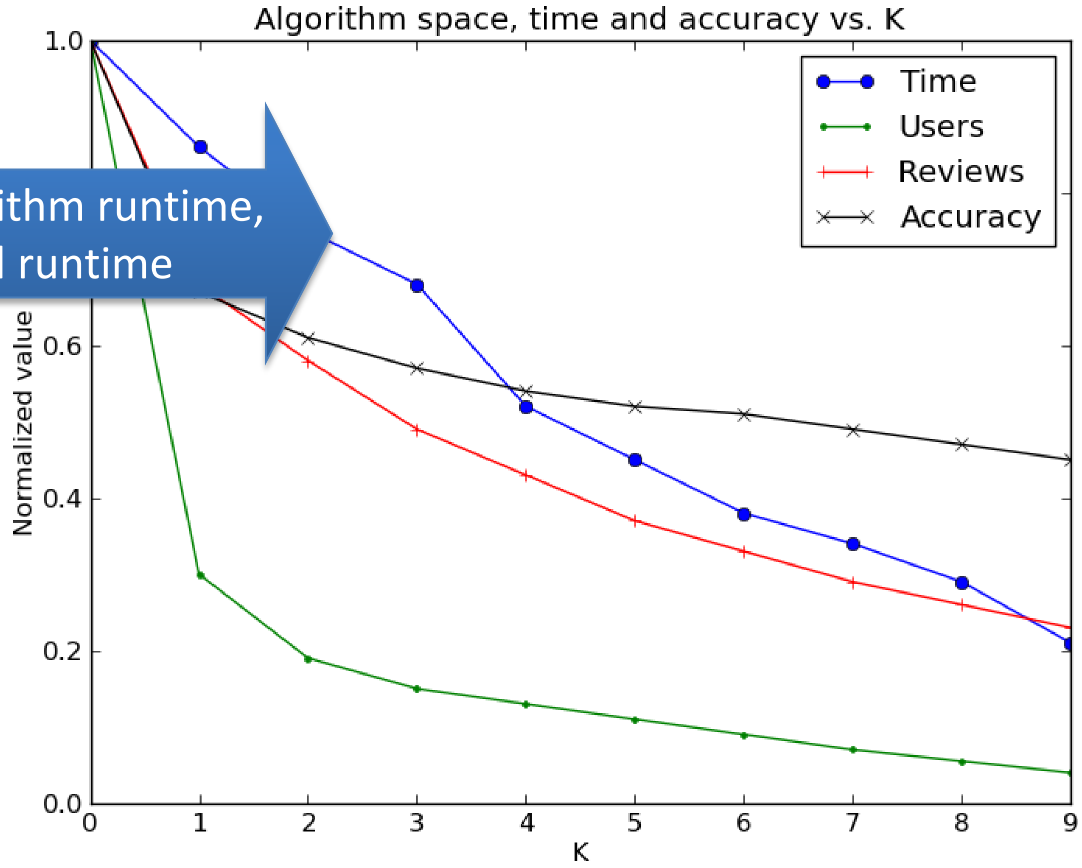
K Threshold Approximation

- Graph size is a major consideration when doing calculations to compute user trust.
- We ignore users who have written less than or equal to k reviews by always assigning them a trust of 0
  - Trust is on range of -1 (untrusted) to 1 (trusted), so 0 means undecided
- Can yield a substantial decrease on algorithm runtime in practice

K Threshold Space, Runtime, Accuracy

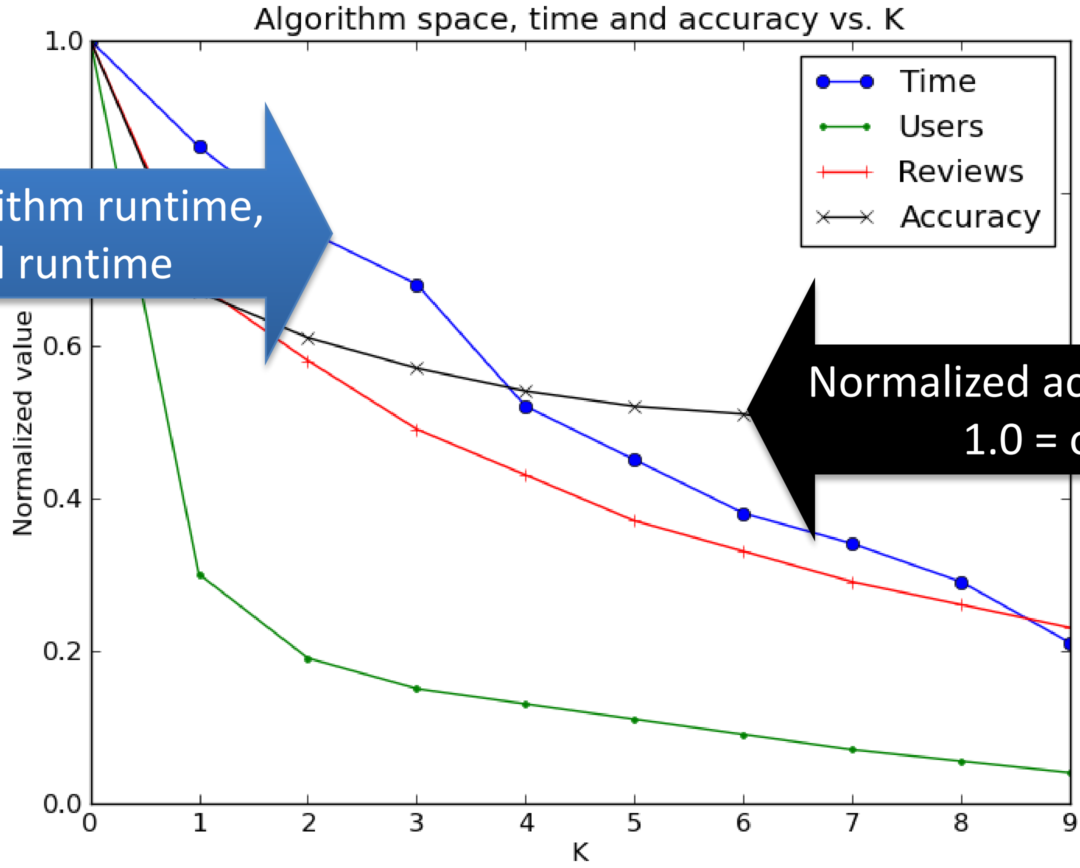


K Threshold Space, Runtime, Accuracy



Normalized algorithm runtime, 1.0 = original runtime

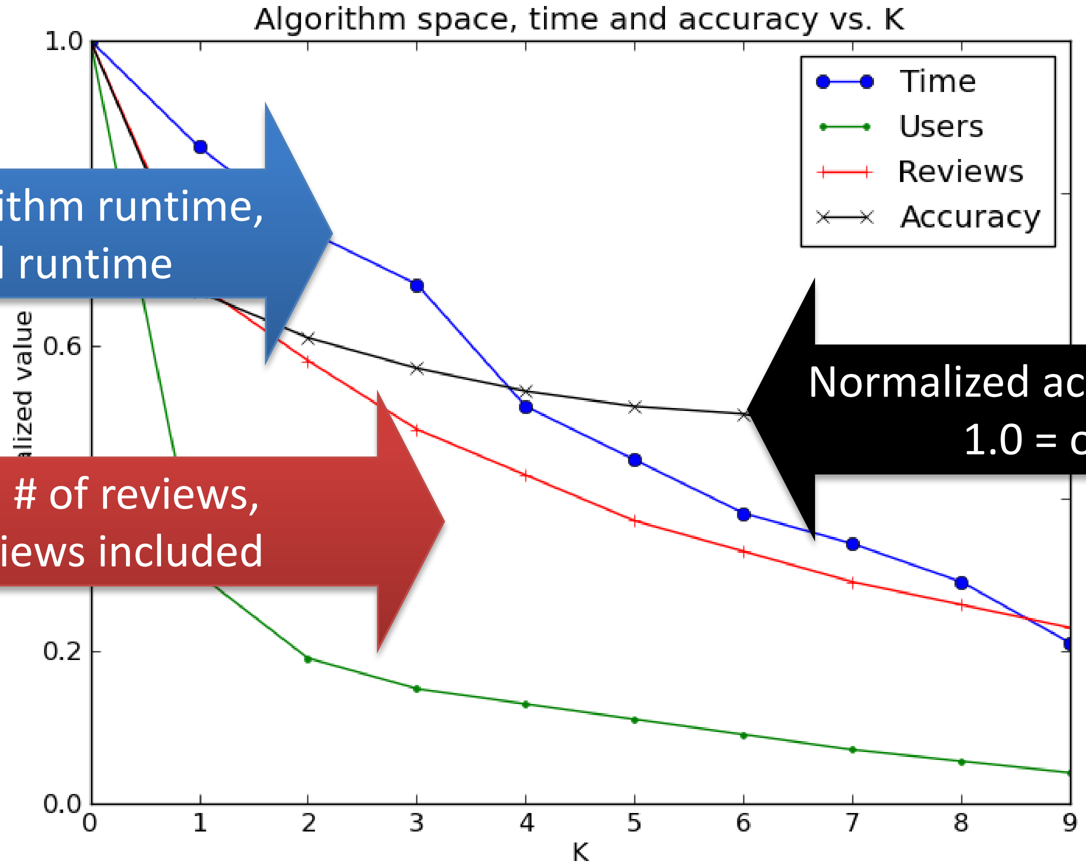
K Threshold Space, Runtime, Accuracy



Normalized algorithm runtime, 1.0 = original runtime

Normalized accuracy of user trust, 1.0 = original trust

K Threshold Space, Runtime, Accuracy

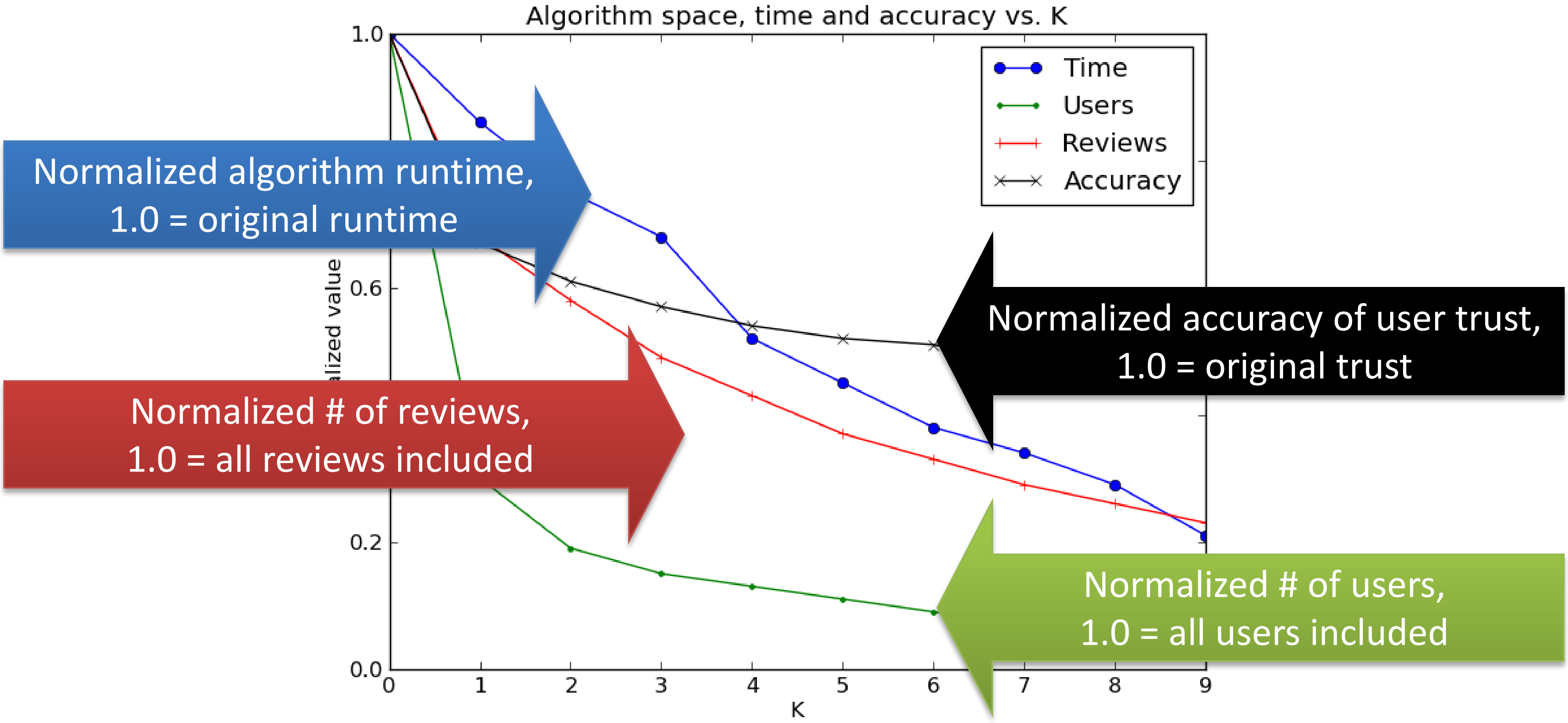


Normalized algorithm runtime, 1.0 = original runtime

Normalized accuracy of user trust, 1.0 = original trust

Normalized # of reviews, 1.0 = all reviews included

K Threshold Space, Runtime, Accuracy



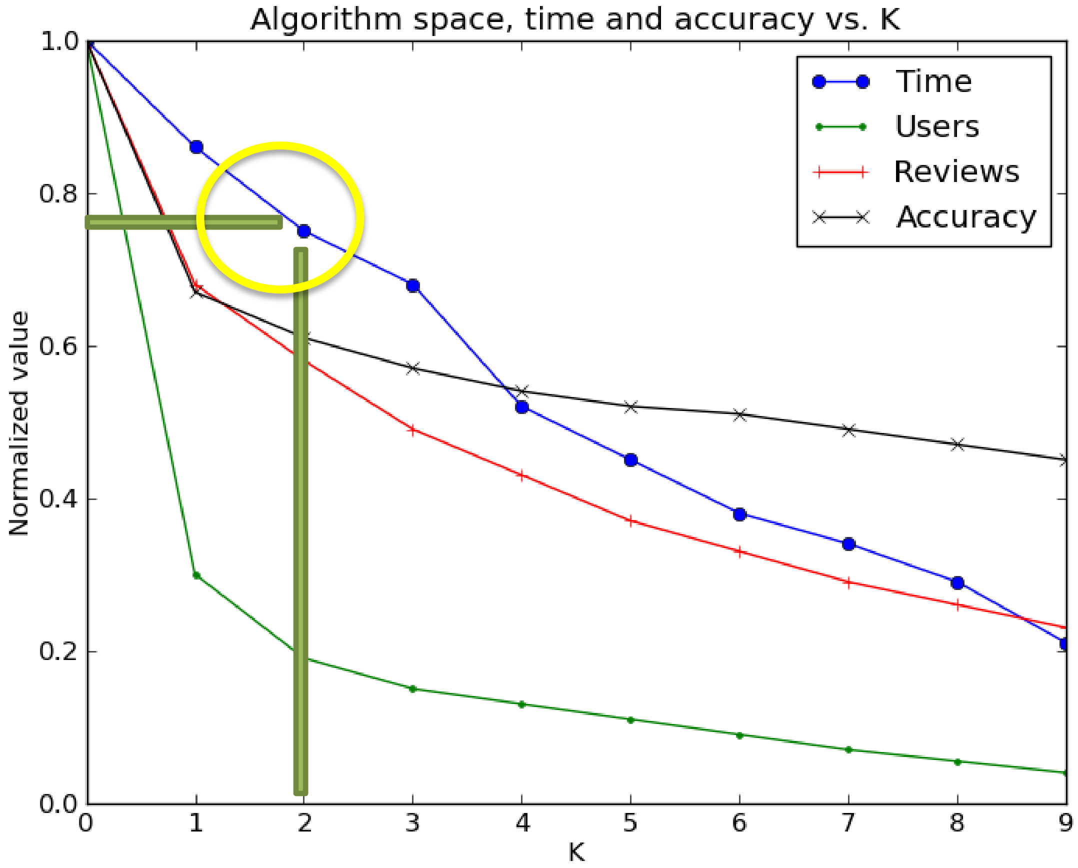
Normalized algorithm runtime,
1.0 = original runtime

Normalized accuracy of user trust,
1.0 = original trust

Normalized # of reviews,
1.0 = all reviews included

Normalized # of users,
1.0 = all users included

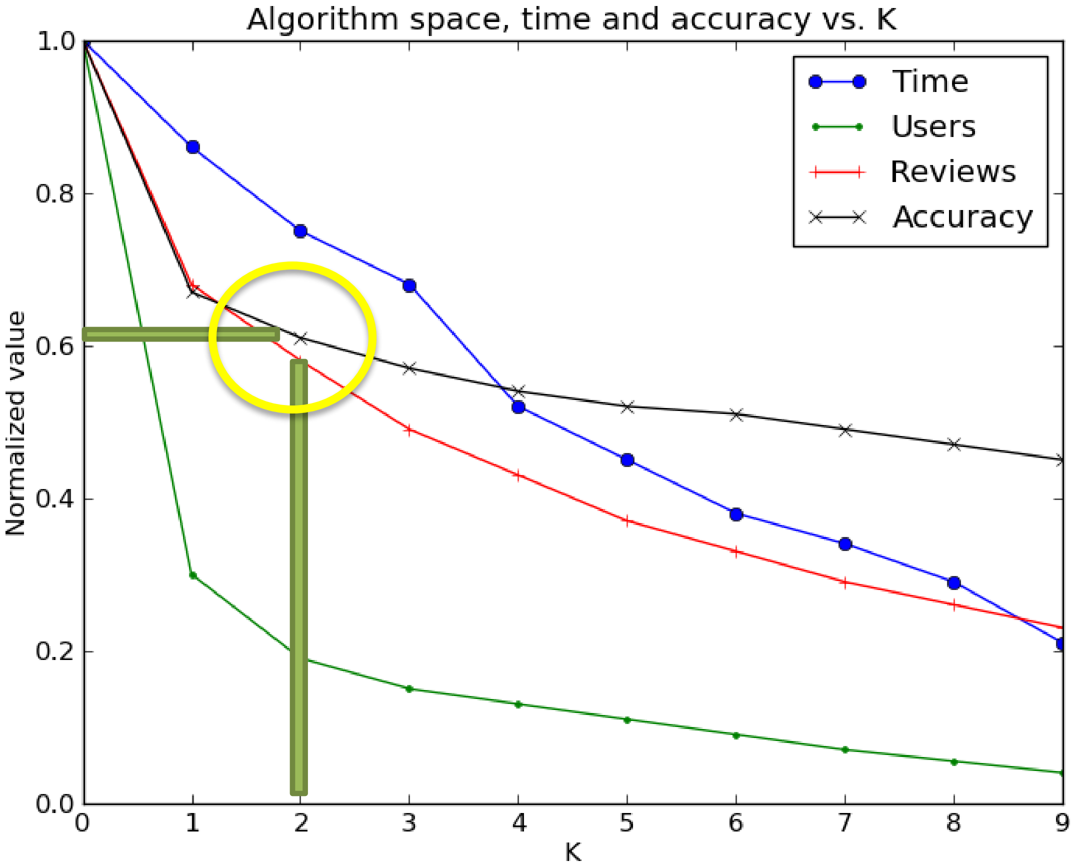
K Threshold Space, Runtime, Accuracy



0.75

Ignore users with 2 or fewer reviews

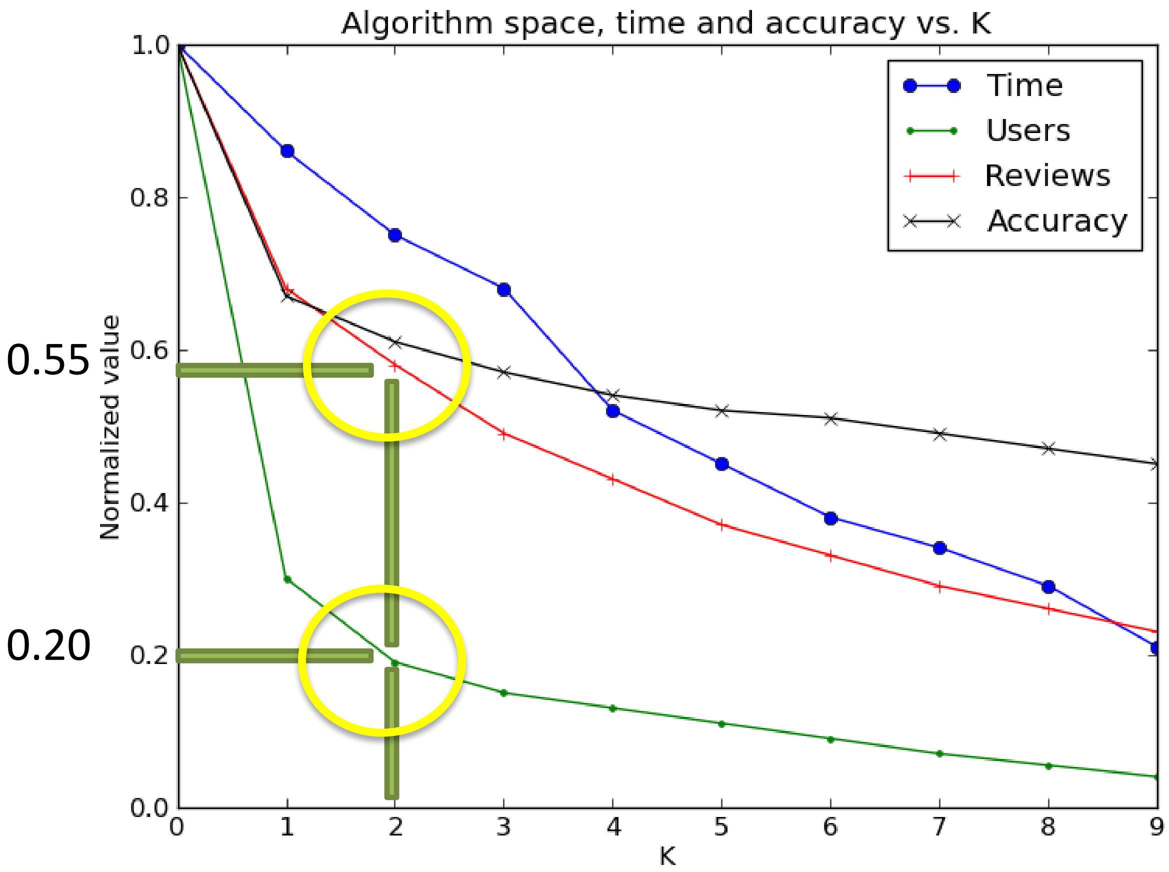
K Threshold Space, Runtime, Accuracy



0.60

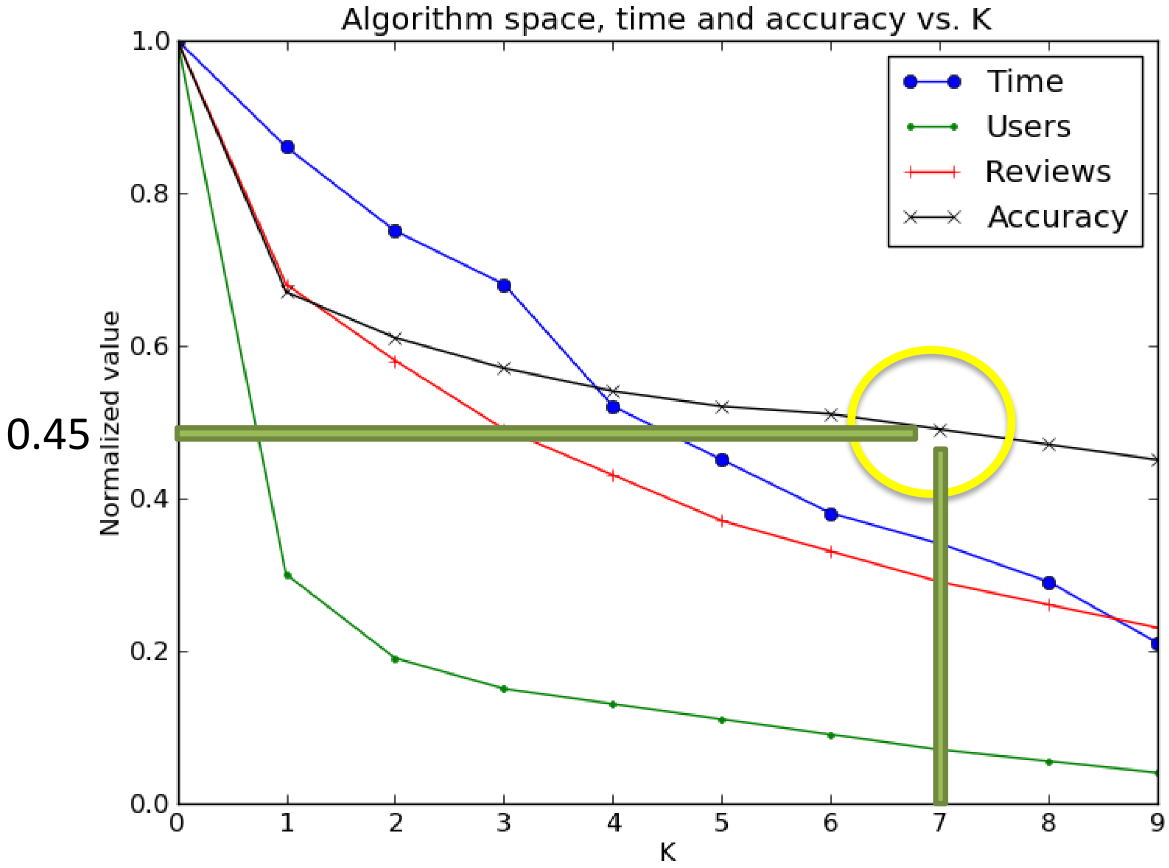
Ignore users with 2 or fewer reviews

K Threshold Space, Runtime, Accuracy



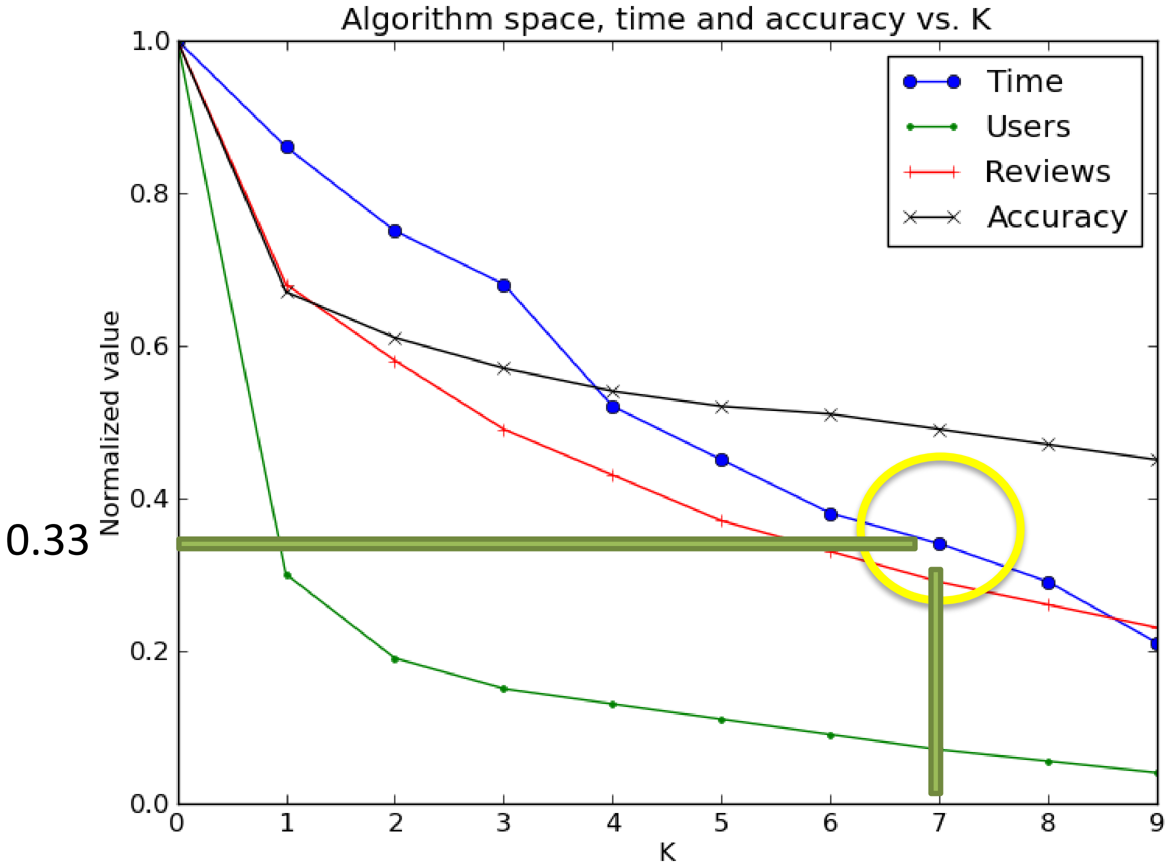
Ignore users with 2 or fewer reviews

K Threshold Space, Runtime, Accuracy



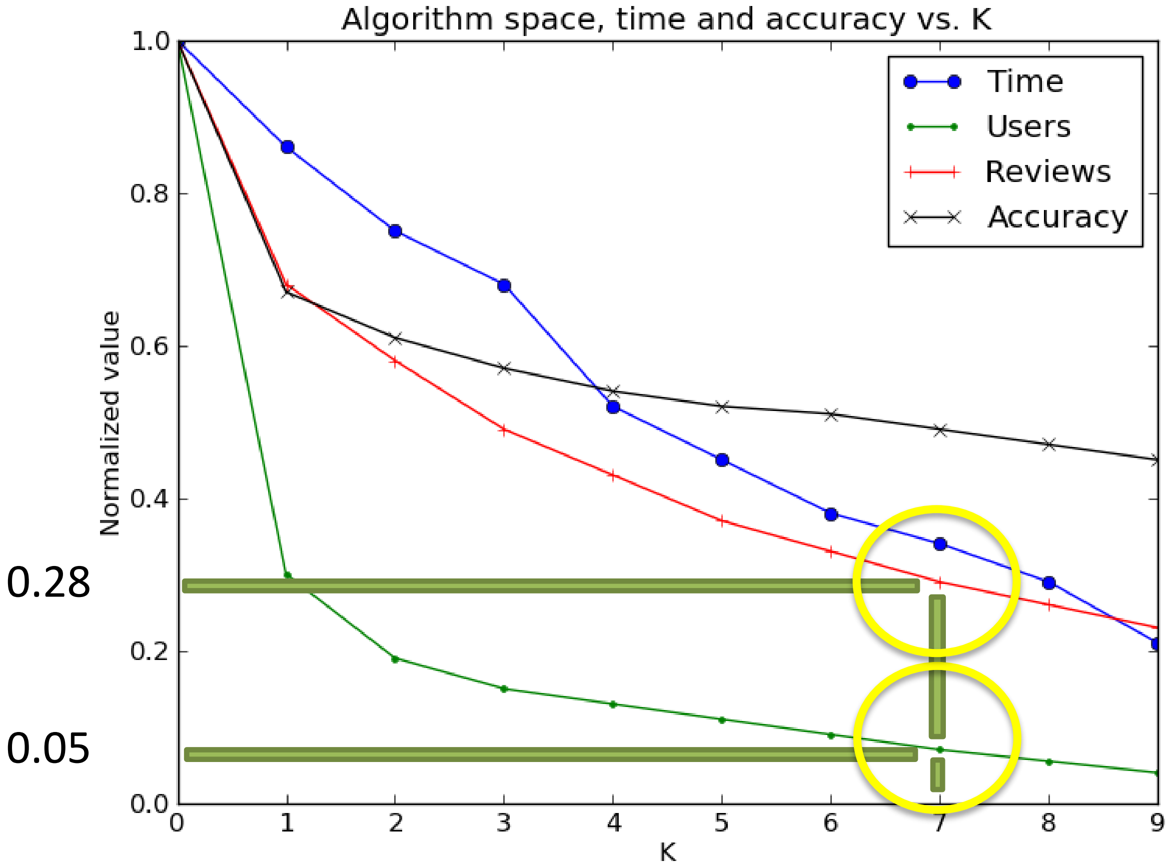
Ignore users with 7 or fewer reviews

K Threshold Space, Runtime, Accuracy



Ignore users with 7 or fewer reviews

K Threshold Space, Runtime, Accuracy



Ignore users with 7 or fewer reviews

Evaluation Using Bots

Evaluation

- To evaluate the algorithm, the dataset was seeded with bots who generated reviews from the following models.

Model	Description
Downvote Bot	Always review 1.0
Upvote Bot	Always reviews 5.0
Conformist	Always reviews the average score
Random	Score taken uniformly at random from 1.0 to 5.0 inclusive

Bot Trust

Model	Average	Standard deviation	Median
Downvote Bot	-0.768	0.381	-0.094
Upvote Bot	0.941	0.096	0.073
Conformist	0.946	0.053	0.942
Random	-0.334	0.603	-0.613

- While the Downvote bot and Conformist had expected behavior
- The Upvote bot did unexpectedly well. The Random model was not 0.

Evaluation

- This can be explained by the score breakdown of dataset. The majority of scores are 5.0, meaning that the Upvote bot often appears to be agreeing with the majority.

Score	1	2	3	4	5
Count	51691	29430	42090	79428	357730
Percentage	9.22%	5.25%	7.51%	14.20%	63.80%

Conclusion

- Some correlation between helpfulness votes and computed trust for very unhelpful/untrusted and very helpful/trusted users (outliers)
- K Threshold Approximation reduces algorithm runtime by ignoring many users with few reviews
- Upvote and conformist bots score high trust due to 4 or 5 star reviews in most of data and nature of algorithm
- Remember this is pure graph analysis, no helpfulness or review text taking into user trust computation

Selected References

- [1] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee, “How opinions are received by online communities: a case study on Amazon.com helpfulness votes,” in Proceedings of the 18th international conference on World wide web, pp. 141–150, ACM, 2009.
- [2] G. Wang, S. Xie, B. Liu, and P. S. Yu, “Review graph based online store review spammer detection,” in Data Mining (ICDM), 2011 IEEE 11th International Conference on, pp. 1242–1247, IEEE, 2011.

Questions