Finding a Better Estimate for Visual Novel Ratings

eriely@mercurialtapestry.net

May, 2023

1 Problem Statement

Visual novels are a subgenre of video game focusing primarily on textual narrative. The Visual Novel Database (VNDB) collects information about these games from over 90,000 users, allowing them to add category tags and vote on each game's rating. Most games in the database do not have many votes, making the average score an unreliable estimate of the underlying quality. Out of 42,000 games, the median number of votes per game is only 6, and the mean number of votes per game is 51.65.

In order to address this issue, VNDB reports the following Bayesian average for each game:

$$\bar{x} = \frac{Cm + \sum_{i=1}^{n} x_i}{C + n}$$

Where C is the mean number of votes per game, m is the average rating of all games, and n is the actual number of votes for that game. The values of these parameters are approximately $C = 61.78$ and $m = 51.65$. As noted before, the median number of votes per game is 6, making the Bayesian prior dominate the average rating for the large majority of games on the database.

Because of the large amount of data collected on not only the games, but the voters as well, it seems reasonable to believe that there is better prior information that can be extracted than just the average rating of all games.

2 Data Source

VNDB provides most of the data shown on their website. These include:

- Votes: There are 1,309,739 votes for 41,635 games, cast by 62,576 unique users. Removed from the dataset are votes from users flagged as fake and votes with non-numeric values.
- Creator attributes: There are 31,642 producers and 16,887 staff members associated with the games. Staff members include all reported staff members associated with a game, including scenario writers, composers, artists, and localization staff.
- Game attributes: VNDB users can vote on various attribute tags for each game. This includes genre, themes, plot elements, and length of game. There are 3709 distinct tags that can be attributed to a game and a total of 1,372,131 such votes.

3 Methodology

We approach this problem in two ways. The first is to estimate the bias of the users that submitted the existing votes. If a vote is submitted by a user whose votes are close to the average score on other games, then we should place more confidence in that user's vote. Likewise, if it is known that a user has a tendency to rate games higher or lower than their average score, we can adjust for that bias in our model.

Bias is estimated using the Platt-Burges model for reviewer bias. For a given game i and reviewer j, let r_{ij} be the rating that reviewer j gives to game i, g_i be the true underlying rating of the game, and b_j be the bias of reviewer j. The Platt-Burges model is as follows:

$$r_{i,j} = g_i + b_j$$

The following least-squares problem is minimized:

$$\min_{g_i, b_j} L = \sum_{ij} A_{i,j}(r_{ij} - g_i - b_j)^2$$

Where A_{ij} indicates if user j reviewed game i. We make one modification to the Platt-Burges model. Rather then treating the true underlying rating as a variable in our optimization problem, we use the mean rating of all votes for the game as the true value. We consider only games with more than 100 votes so that the sample mean is more likely to be a good estimate of the true population mean. This makes the optimization problem notably simpler.

Taking the gradient and setting to zero, we have:

$$\frac{\partial L}{\partial b_j} = \sum_{i,j} -A_{ij} 2(r_{ij} - g_i)_{ij} + 2 \sum_{i,j} A_{ij} b_j = 0,$$

$$b_j = \frac{\sum_{i,j} A_{ij}(r_{ij} - g_i)}{\sum_{i,j} A_{ij}}$$

This equation implies that the bias for a reviewer is just the average difference from the mean for all games that the reviewer has reviewed. We only need to solve [number of reviewers] linear equations, compared to the system of [number of reviewers * number of reviewers] linear equations in the original Platt-Burges model.

Once the estimates for reviewer bias are obtained, we adjust the votes accordingly and recalculate the average unbiased ratings.

The second approach to the problem is to use the attributes of the game to estimate the mean. We combine all possible staff and attribute tags in a one hot encoded feature vector and explore whether this information can be used to predict the rating of the game. The regression models explored are Support Vector Regression, K-Nearest Neighbors Regression, Decision Trees, and Ordinary Least Squares Regression.

One issue with the data is that there are many potential features, over 40,000 if all are used. To address this issue, we use mutual information based feature selection to remove features that do not have good explanatory power. 10-fold Cross Validation is used to choose an appropriate mutual information threshold for feature selection, as well as to tune the parameters of the regression models.

With two potential estimators, we explore ways to combine them for a better estimator. We split the training set into 5 folds for cross validation. Then, for each game in the training set, we select a random sample of votes of random size. The sample sizes are taken from a uniform distribution from [1,50]. This gives each game a sample mean, an unbiased sample mean, a sample size, and the attribute based rating that we found earlier. We use these features to try to model the true mean for each game, using decision trees, and the Bayesian average. This process is then repeated 25 times and take the average mean square error across all processes is calculated to evaluate the training error of our candidate models.

When tuning the Bayesian average, we model the prior as a weighted combination of the unbiased mean and the feature based rating:

$$m = \alpha(\text{unbiased rating}) + (1 - \alpha)(\text{feature rating}),$$

$$0 < \alpha < 1$$

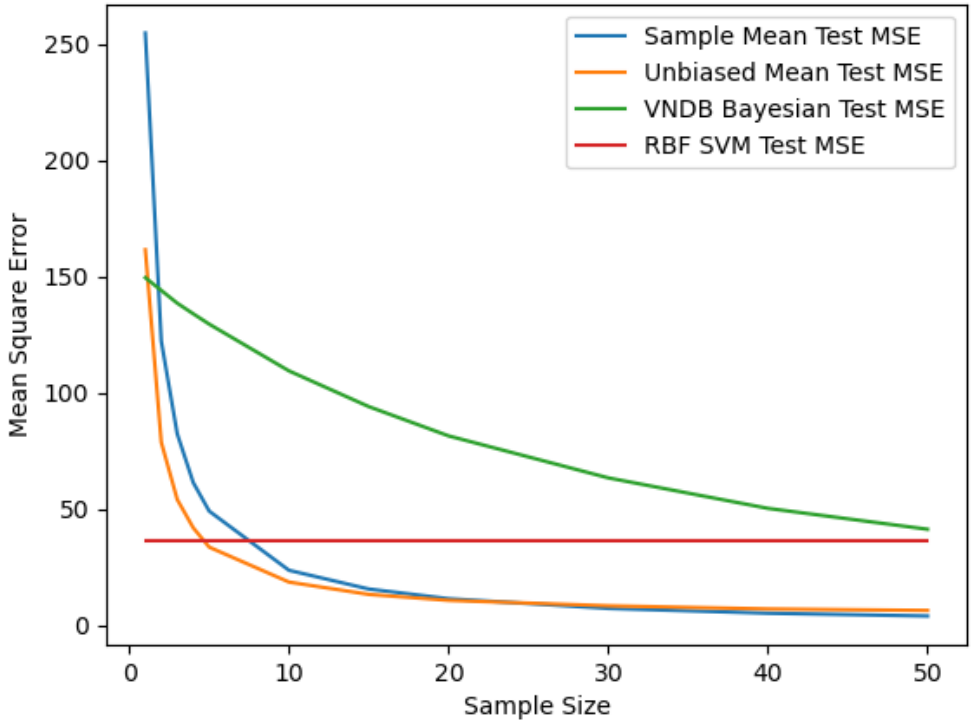
4 Evaluation and Final Results

4.1 Estimating Voter Bias

After calculating the estimates for voter bias, we tested the model on a set of games with more than 100 votes that was excluded from the training set. In order to evaluate the bias estimation model we selected 25 random samples of votes of fixed size from each game in the test set, and then calculated the mean square

errors of the unbiased sample mean compared to the true mean for each game over all samples. This was experiment was repeated with a range of fixed sample sizes. The idea was to simulate how the unbiased model performs when only a small number of votes has been collected, compared to the simple sample average. The training and test errors are tabulated in Appendix A.

Figure 1: Model Test Errors



The figure above compares the test MSE of our unbiased mean to the test MSE of the sample mean and VNDB's own Bayesian rating. Also included for reference is the estimate of the score based on the game's features which we will go over in the next section. What we find is that the unbiased mean is an improvement over the sample mean for sample sizes between 1 and 20 votes. With sample sizes greater than 20, the sample mean begins to outperform the unbiased mean, but both estimators have small MSE's as the sample size gets large. Because the median number of votes per game is 6, it is reasonable to believe that using the unbiased mean can improve the accuracy of our ratings for over half of the games in the database.

VNDB's Bayesian rating underperforms both the sample mean and the unbiased mean for all sizes greater than 1, and unlike the sample mean the MSE of the Bayesian rating remains high for sample sizes larger than 20.

4.2 Feature Based Estimation

The feature based regression models were trained and tested on the same training and test sets we used to calculate the bias estimates. After tuning the parameters with cross-validation, the best model was a support vector regression model with a regularization parameter of 10 and a gamma parameter of 0.01. The SVR model incorporated features with mutual information greater than 0.001, 1715 in total. The other regression models, and especially OLS regression performed poorly with such large numbers of features, and performed best when limited to features with higher mutual information. The regularization in the support vector models makes them more robust with larger numbers of features.

Detailed reporting of cross-validation results are included in Appendix B.

Table 1: Model Train and Test Errors

Model	Mutual Information	Number of Features	Train MSE	Test MSE
RBF SVR	> 0.001	1715	35.71	36.13
Linear SVR	> 0.001	1715	39.69	39.99
KNN Regression	> 0.02	34	60.68	62.40
OLS Regression	> 0.015	75	53.09	53.80
Regression Tree	> 0.025	75	62.6	62.42

The feature based estimates, and particularly the SVR estimates performed quite well relative to the sample and unbiased mean estimates for small numbers of votes. The RBF SVR estimate performed on par with the unbiased mean estimate for a sample of 5 votes, and performed notably better when the number of votes was smaller. Again, when the median number of votes is 6, this estimate has potential to be reasonable for a large number of games in the database and appears to be more robust than the VNDB Bayesian score.

By inspecting the highest and lowest rated games based on the RBF SVR model, we can confirm that the model generally does a good job of rating good games high and bad games low, even without any information from votes.

Table 2: Top and Bottom 3 Games by RBF SVR Feature Rating

Tag	Feature Rating	True Mean
Muv-Luv Alternative	90.20	90.30
Soukou Akki Muramasa	89.62	89.72
Utawarerumono: Futari no Hakuoro	89.41	89.51
Sakura Maid	48.40	46.88
Sakura Christmas Party	49.31	49.21
Sakura Maid 3	50.10	50.00

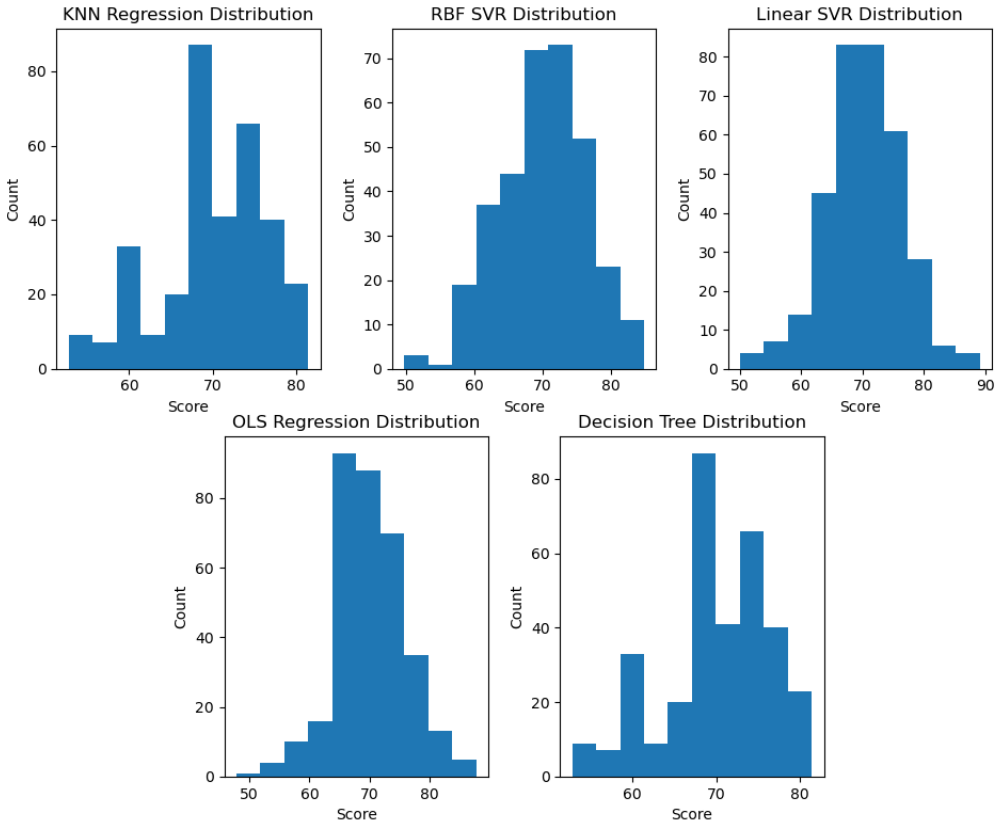
When we examine the selected features, with the exception of "High School Student Heroine", we start to see traits that are reflective of qualities that people feel strongly about. "Other Perspectives" and "Proactive Protagonist" might be correlated with stories that people particularly enjoy and less likely to be found in stories that they do not enjoy. On the other hand, "No Opening Movie" is likely associated with games with particularly low budget and quality.

Table 3: Predictors With the Highest Mutual Information

Tag	Mutual Information	Games in Training Set with Tag
Protagonist with a Face	0.0514	917
Other Perspectives	0.0507	585
Proactive Protagonist	0.0447	265
High School Student Heroine	0.0394	625
No Opening Movie	0.0388	173

Despite the large number of features and relative sparsity of the data, we are still able to capture useful information that differentiates the quality of games.

Figure 2: Test Prediction Distributions



The distribution plots do show that the test predictions tend to be weighted more heavily towards ratings higher than 70. All of the regression models had a larger MSE when predicting the rating for games with a mean score less than 70. We see that the RBF SVR model had the greatest density of predictions less than 70, and was able to perform better relative to the other models on these lower rated games.

Table 4: Model Test Errors by Mean Score

Model	MSE, Score < 70	MSE, Score ≥ 70
RBF SVR	48.13	26.97
Linear SVR	54.07	29.25
KNN Regression	67.54	58.48
OLS Regression	64.17	45.9
Regression Tree	81.02	48.22

4.3 Combining the Models

The parameters for the Bayesian average that performed the best under cross validation were an alpha value of 0.2 and a prior constant of 24. The decision tree model performed the best with a minimum split size of 4, a minimum leaf size of 6, and a maximum depth of 7. Detailed cross validation results are tabulated in Appendix C.

After generating the test set using the same method we generated the training sets, we found that the mixed models both performed better than sample mean and the unbiased mean alone.

Table 5: Mixed Model Train and Test Error

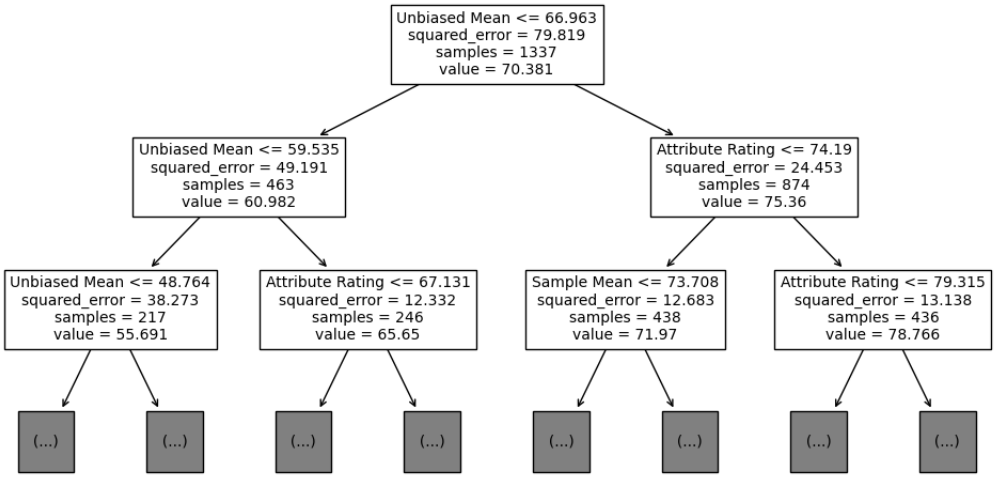
Model	Train MSE	Test MSE
Sample Mean	20.85	23.25
Unbiased Mean	16.54	17.85
Decision Tree	7.95	13.01
Bayesian Average	6.12	10.75

An alpha value of 0.2 in the Bayesian average model means that most of the weight on our prior comes from the attribute based rating estimate. This is in line with our observation that the attribute based rating outperforms the unbiased mean estimator when the sample size is small, and does worse as the sample size gets larger. Giving a higher weight to the attribute rating makes the model more accurate for small samples. As the sample size gets larger, the Bayesian priors make up a smaller proportion on the estimated rating, and because the unbiased rating is correlated with the actual vote, more weight is given indirectly to the unbiased rating.

One observation is that the tuned model does not introduce much distortion to the rating when the number of votes is high. When the model is applied to the entire set of votes, the MSE of the mixed model is 0.15, which is a significant improvement to the MSE of 5.70 when using VNDB's Bayesian model.

If we inspect the branches of our decision tree, we can observe that the model does not usually branch based on the sample size. The unbiased mean and the attribute rating working together do a good job of reducing the prediction error, regardless of the variance that comes from small sample sizes. Even in the first two branches, most nodes in the tree have a smaller squared error than the unbiased mean alone, with the model only really struggling with games that have very low vote averages.

Figure 3: Decision Tree Branches



4.4 Final Results and Conclusions

The idea of using a Bayesian average is reasonable given the generally low vote counts for most games. While the VNDB implementation is generally worse than the sample average, by lowering the constant parameter and finding a better prior estimate for the game's rating, we can generate an estimate that works better than the sample mean alone for a small number of votes.

For games that have a larger number of votes, we have shown that the sample mean performs better than the Bayesian average. Even though the effects of the Bayesian average will naturally decay as the number of votes increases it may be a good idea to simply use the sample mean as the rating once the number of votes is greater than 30.

5 Appendix

A Bias Estimation Results

Table 6: Model Test Errors by Sample Size

Sample Size	Sample Mean MSE	Unbiased Mean MSE	Bayesian MSE
1	254.73	161.51	149.47
2	122.16	78.65	143.98
3	82.28	54.03	138.5
4	61.35	41.88	133.95
5	49.05	33.55	129.49
10	23.58	18.51	109.37
15	15.52	13.21	94.0
20	11.36	10.65	81.4
30	7.18	8.31	63.35
40	5.12	6.95	50.23
50	3.91	6.33	41.25

B Feature Based Estimation Results

Table 7: SVR Cross Validation Error, RBF Kernel, C=10, Gamma = 0.01

Mutual Information (>)	CV MSE
0.04	71.75
0.03	64.63
0.025	62.14
0.02	57.96
0.015	51.80
0.01	45.58
0.005	38.88
0.001	35.71

Table 8: SVR Cross Validation Error, Linear Kernel, C=0.1

Mutual Information (>)	CV MSE
0.04	72.15
0.03	66.04
0.025	63.61
0.02	59.99
0.015	53.99
0.01	47.97
0.005	42.53
0.001	39.69

Table 9: KNN Cross Validation Error

Mutual Information (>)	k=2	k=3	k=5	k=7	k=10
0.04	97.37	90.62	79.45	75.73	76.95
0.03	90.3	79.64	71.32	68.46	66.61
0.025	78.57	72.71	66.31	62.95	61.69
0.02	72.19	64.87	63.43	61.24	60.68
0.015	76.03	66.9	63.25	62.67	61.92
0.01	74.47	72.65	67.33	66.93	66.37
0.005	80.62	77.43	78.0	78.46	79.04
0.001	94.9	85.66	82.73	81.92	85.08

Table 10: OLS Regression Cross Validation Error

Mutual Information (>)	CV MSE
0.04	69.69
0.03	63.48
0.025	60.97
0.02	57.58
0.015	53.09
0.01	9.56e+23
0.005	2.36e+25
0.001	3.96e+26

Table 11: Regression Tree Cross Validation Error, Minimum Split Size = 37, Minimum Leaf Size = 6

Mutual Information (>)	CV MSE
0.04	69.83
0.03	63.8
0.025	62.6
0.02	63.67
0.015	63.23
0.01	65.65
0.005	65.2
0.001	67.34

C Mixed Model Results

Table 12: Bayesian Average Cross Validation Results

Number of Priors	1	2	3	4	5	6	7	8	9	10
α =0.1	13.9	11.24	9.78	8.84	8.18	7.7	7.34	7.06	6.85	6.68
α =0.2	14.3	11.71	10.25	9.29	8.6	8.09	7.69	7.38	7.14	6.93
α =0.3	14.73	12.23	10.8	9.84	9.14	8.61	8.2	7.87	7.6	7.37
α =0.4	15.19	12.81	11.43	10.49	9.8	9.28	8.86	8.51	8.23	7.99
α =0.5	15.67	13.45	12.14	11.24	10.58	10.07	9.66	9.32	9.04	8.8
α =0.6	16.19	14.14	12.93	12.1	11.48	11.0	10.61	10.29	10.02	9.79
α =0.7	16.73	14.89	13.8	13.06	12.5	12.06	11.71	11.42	11.18	10.97
α =0.8	17.29	15.7	14.76	14.11	13.64	13.26	12.96	12.72	12.51	12.33
α =0.9	17.89	16.56	15.79	15.27	14.89	14.6	14.36	14.17	14.01	13.88
Number of Priors	11	12	13	14	15	16	17	18	19	20
α =0.1	6.55	6.46	6.38	6.32	6.28	6.26	6.24	6.23	6.23	6.24
α =0.2	6.77	6.64	6.53	6.44	6.37	6.31	6.26	6.22	6.19	6.17
α =0.3	7.18	7.03	6.89	6.78	6.68	6.6	6.53	6.47	6.42	6.37
α =0.4	7.79	7.62	7.47	7.34	7.23	7.13	7.05	6.97	6.9	6.84
α =0.5	8.6	8.42	8.26	8.13	8.01	7.9	7.81	7.73	7.65	7.58
α =0.6	9.6	9.42	9.27	9.14	9.02	8.92	8.82	8.74	8.66	8.59
α =0.7	10.79	10.63	10.5	10.37	10.27	10.17	10.08	10.0	9.93	9.87
α =0.8	12.18	12.05	11.93	11.83	11.74	11.66	11.59	11.52	11.47	11.41
α =0.9	13.77	13.67	13.59	13.51	13.45	13.39	13.34	13.3	13.26	13.23
Number of Priors	21	22	23	24	25	26	27	28	29	30
α =0.1	6.25	6.27	6.29	6.31	6.34	6.37	6.4	6.44	6.47	6.51
α =0.2	6.16	6.15	6.14	6.14	6.14	6.14	6.15	6.16	6.17	6.19
α =0.3	6.34	6.3	6.28	6.26	6.24	6.22	6.21	6.2	6.2	6.19
α =0.4	6.79	6.75	6.7	6.67	6.64	6.61	6.58	6.56	6.54	6.52
α =0.5	7.52	7.47	7.42	7.37	7.33	7.29	7.26	7.23	7.2	7.18
α =0.6	8.53	8.47	8.42	8.37	8.33	8.29	8.25	8.22	8.18	8.15
α =0.7	9.81	9.76	9.71	9.66	9.62	9.58	9.55	9.52	9.49	9.46
α =0.8	11.37	11.32	11.28	11.25	11.21	11.18	11.15	11.13	11.11	11.08
α =0.9	13.2	13.17	13.15	13.12	13.1	13.09	13.07	13.06	13.05	13.04

Table 13: Regression Tree Cross Validation Results, Max Depth = 7

Split Size	Leaf Size 1	Leaf Size 2	Leaf Size 3	Leaf Size 4	Leaf Size 5	Leaf Size 6	Leaf Size 7	Leaf Size 8
1	8.85	8.5	8.26	8.14	8.01	7.95	7.96	8.01
2	8.92	8.5	8.26	8.13	8.01	7.95	7.96	8.01
3	8.85	8.51	8.26	8.13	8.01	7.95	7.96	8.01
4	8.82	8.53	8.24	8.14	8.01	7.94	7.96	8.01
5	8.86	8.47	8.27	8.13	8.0	7.95	7.96	8.01
6	8.78	8.5	8.27	8.13	8.0	7.95	7.96	8.01
7	8.82	8.48	8.25	8.13	8.01	7.95	7.96	8.01
8	8.72	8.47	8.26	8.13	8.0	7.95	7.96	8.01
9	8.77	8.45	8.24	8.11	8.01	7.95	7.96	8.01
10	8.73	8.45	8.2	8.11	8.01	7.95	7.96	8.01
11	8.74	8.45	8.24	8.12	8.02	7.95	7.96	8.01
12	8.72	8.45	8.23	8.1	8.03	7.95	7.96	8.01
13	8.76	8.46	8.24	8.09	8.04	7.97	7.96	8.01
14	8.74	8.49	8.26	8.11	8.04	7.99	7.96	8.01
15	8.74	8.5	8.28	8.16	8.07	8.01	7.99	8.01

D References

VNDB Source Code. https://code.blicky.net/yorhel/vndb

VNDB Database Dump. https://vndb.org/d14

Platt, B. and Burges, C. (2012). Regularized least squares to remove reviewer bias.