Efficient Model Performance Evaluation Using a Combination of Expert and Crowd-sourced Labels

Sam Corbett-Davies **Viet-An Nguyen** **Udi Weinsberg**
Meta Platforms, Inc.

Abstract

As models, particularly large language models (LLMs), are deployed on increasingly challenging tasks, correctly evaluating their performance is growing in importance and difficulty. Expert human labelers are high-quality but scarce and resource-intensive to obtain, while crowd-sourced labels are more readily accessible at scale but lower in quality. We propose MAVEN (Model And Voter EvaluatioN), a hierarchical Bayesian model that combines these two label sources to produce model performance estimates on binary tasks that are less biased than using crowd-sourced labels alone and have lower variance than using expert labels alone. By modeling the ranking of model scores, MAVEN is robust to a range of prediction distributions and achieves constant inference time regardless of dataset size. We validate our approach on both simulated and real-world data, and deploy it to measure production models at Meta.

1 INTRODUCTION

Accurately measuring the performance of model predictions in a scalable way is a crucial component of any modern AI system. However, collecting high-quality evaluation datasets can be challenging since expert human annotations require significant effort and are thus scarce. Practitioners have turned to crowd-sourced labels that are easier to acquire at scale but lower in quality, hoping that an accurate consensus will emerge out of multiple labels per item. However, concerns about the quality of evaluation datasets remain; Vasudevan et al. (2022) and Reiss et al. (2020) present two recent examples of concerning error rates found in important datasets. Furthermore, the common practice to leverage all labels (including the low quality crowd-sourced ones) for training models may cause models to learn to replicate the systematic inaccuracies in these labels, further complicating the task of accurately assessing their performance.

We present MAVEN (Model And Voter EvaluatioN), a hierarchical Bayesian model that combines expert and crowd-sourced labels to produce estimates of model performance on binary classification tasks. MAVEN produces performance assessments that are less biased than using crowd-sourced labels alone while having lower variance than using expert labels alone. Fig. 1 illustrates this by comparing MAVEN to expert-only and crowd-sourced labels, showing that MAVEN produces low-bias and narrow-CIs measures of several performance metrics, even with few expert labels.

The main contributions of our work are:

- MAVEN makes novel use of the *ranking* of model scores, making it robust to a diverse range of prediction distributions. By summarizing these rankings in terms of comparisons between sets of items with the same crowd-sourced labels, MAVEN achieves constant inference time regardless of dataset size, ensuring high scalability.
- Through extensive simulations and evaluations on real-world datasets, we show that MAVEN outperforms related work in a range of settings, and generalizes well to different score distributions and performance metrics.
- We deployed MAVEN in two production settings. In one of these, we show that MAVEN is able to improve the overall labeling efficiency by 42% while producing an equivalent estimate of model performance. These results confirm MAVEN is a practical solution for highly efficient, high-quality model evaluation at scale.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

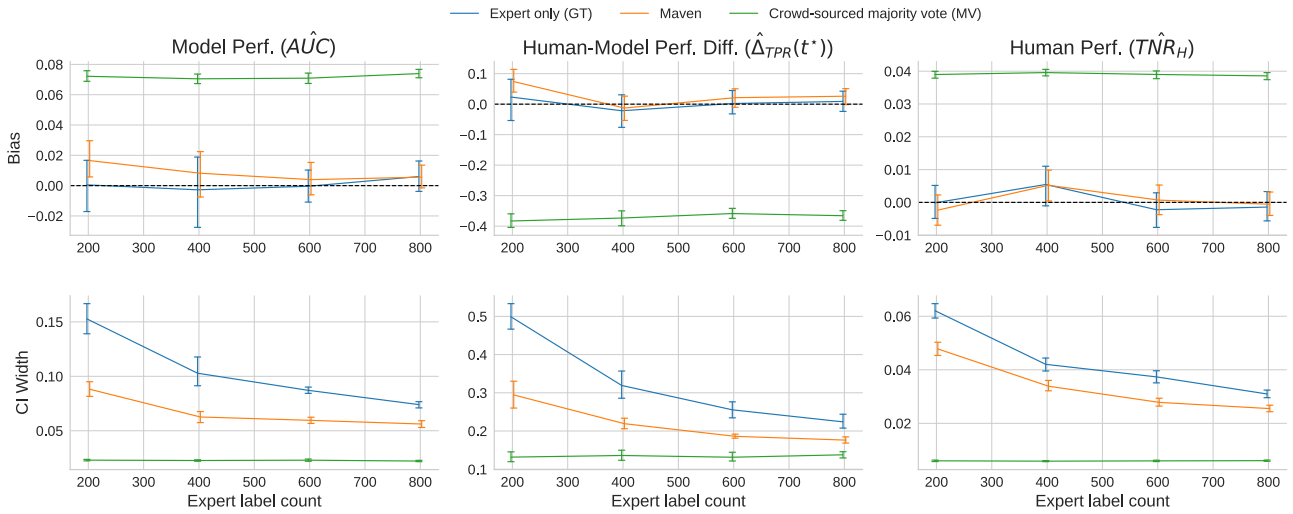


Figure 1: Performance of MAVEN when estimating metrics for a production model (Task 1, see Sec. 5.1) using 4000 items with crowd-sourced labels and a varying number of expert labels. Across three performance metrics, MAVEN achieves the same bias with smaller CIs when compared to evaluating with expert labels only. Evaluating with crowd-sourced majority vote labels produces very biased estimates. [1]

2 RELATED WORK

Crowdsourcing and annotation modeling: A common approach for aggregating multiple labels is through majority votes. However, assuming humans are equally reliable has been challenged by numerous studies (Passonneau and Carpenter, 2014; Paun et al., 2018). In practice, expertise and consistency vary significantly, motivating the development of modeling approaches for label aggregation.

The seminal work by Dawid and Skene (1979) (DS) estimates latent true labels by modeling annotator reliability via confusion matrices. This framework has inspired a plethora of extensions, such as modeling latent variables with alternative functions (Raykar et al., 2010; Rodrigues et al., 2014; Yan et al., 2014; Ruiz et al., 2019), and replacing confusion matrices with scalars (Whitehill et al., 2009; Hovy et al., 2013) and multidimensional vectors (Welinder et al., 2010; Zhou et al., 2012).

Learning using noisy labels: Research on learning approaches that utilize noisy labels is closely related to our work (Natarajan et al., 2013; Song et al., 2022; Uma et al., 2022; Wei et al., 2023). Numerous studies have proposed strategies for incorporating the noise and variability inherent in human-generated labels during model training. Recently, more work has also focused on evaluation processes using noisy labels, which include (i) "soft metrics" such as *cross entropy* (Peterson et al., 2019; Pavlick and Kwiatkowski, 2019), *entropy correlation* (Uma et al., 2020), and *Jensen-Shannon divergence* (Nie et al., 2020), and (ii) new evaluation procedures such as *disagreement deconvolution* transformation (Gordon et al., 2021). Our work is orthogonal and complementary to these works as we do not propose any new evaluation metrics and instead focus on modeling approaches to better incorporate human label noises into existing metrics.

Efficient model evaluation: LLM-as-a-judge has recently emerged as a highly scalable, high-quality approach for model evaluation. The use of LLMs as annotators and judges is a rapidly evolving research area, as highlighted by several recent surveys (Li et al., 2025, 2024; Tan et al., 2024; Gu et al., 2024). Nevertheless, multiple recent studies have shown important limitations and biases inherent in LLM-as-a-judge methods, underscoring the continued need for human-generated labels and validation (Reif and Schwartz, 2024; Krumdick et al., 2025; Bavaresco et al., 2025; Dorner et al., 2025; Baumann et al., 2025).

Another promising direction leverages the abundance of unlabeled data alongside a small set of labeled examples. Welinder et al. (2013) fit simple mixture models to the class-conditional score distributions; recent work by Shanmugam et al. (2025) extend this idea to jointly model multiple scores per task, sharing information between scores to improve the simultaneous evaluation of multiple models. Unlike MAVEN, neither model incorporates crowd-sourced labels, and both model raw

[1] Note: error bars shown in this plot are not the CIs produced by the estimators, they are the standard errors of the plotted quantities over the replicated datasets.

scores rather than score rankings. AutoEval (Boyeau et al., 2025), a frequentist approach building on the prediction-powered inference (PPI) framework (Angelopoulos et al., 2023), combines a small amount of human-labeled data with a large pool of synthetic data to yield more accurate estimates. However, AutoEval is limited in the metrics it can estimate to those that are expectations over item-level metric functions.

Human-AI combination: Our work shares similarities with the expanding body of research on modeling human-AI combinations (Kelly, 2025; Choudhary et al., 2025). Kerrigan et al. (2021) present methods for combining human predictions with model probabilities through the use of confusion matrices and calibration, enabling more accurate label aggregation. Steyvers et al. (2022) propose a Bayesian framework for modeling human–AI complementarity to quantify and leverage the respective strengths of humans and AI systems. Showalter et al. (2024) and Kelly et al. (2025) present approaches for incorporating machine predictions to query human labels in a resource-efficient manner. In contrast, our research addresses the problem of evaluating the performance of model predictions, instead of improving the combined predictions.

Programmatic weak supervision: Another related research area is Programmatic Weak Supervision (PWS) (Ratner et al., 2016, 2017), a data-labeling paradigm that generates large amounts of training data by applying multiple labeling functions (LFs) to produce noisy labels and then combining them into a single, high-quality label set using a label model (LM). Prior work has explored designing diverse types of LFs, automatically generating LFs, and developing LMs for label aggregation (Zhang et al., 2022). In this context, MAVEN is closely related to an LM that uses noisy crowd-sourced labels as LFs. However, MAVEN's primary focus is on efficiently and accurately evaluating the performance of input classifiers or LLMs using these noisy labels, rather than on label aggregation itself.

3 MAVEN

3.1 Preliminaries

We study the problem of evaluating predictions $X_i \in \mathbb{R}$ from a machine learning model against a (possibly latent) binary ground truth $Y_i \in \{0, 1\}$. To avoid confusion between this target model and the Bayesian model we are proposing to evaluate it, we will refer to the target model as the *predictor*. We assume the predictions contain no ties—in practice, in all reasonably-sized datasets we can break ties at random with negligible impact on predictor evaluation.

The evaluation dataset is $\mathcal{D} = \{(X_i, V_i, T_i)\}_{i=1}^n$. Crowdsourced "voters" are tasked with determining whether each item is positive; $V_i \in \{0, 1, ..., c\}$ counts how many of the c voters gave positive votes. Every item is voted on. $T_i \in \{0, 1, \emptyset\}$ records the expert label received by an item; $T_i = \emptyset$ means that no expert label is available. We consider expert labels to be ground truth, therefore $T_i \neq \emptyset \iff T_i = Y_i$. Items are selected for expert labeling uniformly at random.

We also define helper variables $T_i^{(0)} = \mathbf{1}\{T_i \neq 1\}$ and $T_i^{(1)} = \mathbf{1}\{T_i \neq 0\}$. These are indicators for whether it's *possible* for Y_i to take a given value: $T^{(t)} = 1$ indicates that Y_i could equal t (because it is either unobserved or has been observed to equal t), while $T^{(t)} = 0$ indicates that Y_i cannot equal t (because it has been observed to take the opposite value). By definition, these variables are observed for all items.

Because there are only a small number of unique (V_i, T_i) tuples, it is useful to define an item's *vote set* $\mathcal{V}_i \in \{1, 2, ...m\}$ to index which of the m unique tuples it belongs to. The functions $V(\mathcal{V}_i) = V_i$ and $T^{(t)}(\mathcal{V}_i) = T_i^{(t)}$ map the vote sets back to the actual votes and golden set labels for that vote set. $N_i = |\{j : \mathcal{V}_j = i\}|$ is the number of items in vote set i.

Finally, to tractably model the ranking of items according to the predictions (see Section 3.2.1), we introduce pairwise comparison variables W_{kl}:

$$W_{kl} = |\{(i, j) \in [n] \times [n] \,:\, \mathcal{V}_i = k, \mathcal{V}_j = l, X_i > X_j\}|$$

These compare every item in vote set k against every item in vote set l, and count the number of comparisons where the former is scored above the latter. C_{kl} is defined as the total number of such comparisons.

3.2 Bayesian Model

In order to evaluate the performance of the predictor against the latent ground truth, Maven uses a hierarchical Bayesian model (see Fig. 2). The goal is to estimate $p_i = P(Y_i = 1 \,|\, \mathcal{V}_i, X_i)$ for each item, which can be used to impute the missing labels and then estimate predictor precision, recall, AUC, and other performance metrics of interest.

The MAVEN model consists of two components: (1) a *vote model* for voter responses, and (2) a *score model* for the predictors. The following sections describe the generative processes for both.

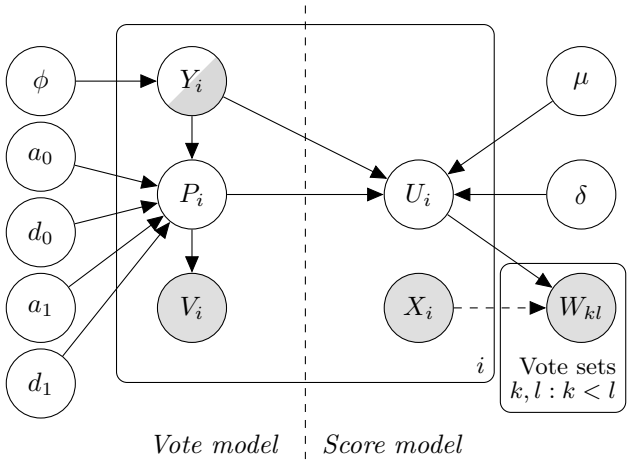


Figure 2: Graphical representation of the Bayesian model used in MAVEN. (Partly) shaded nodes are (partly) observed, and a dashed arrow indicates a deterministic relationship. The dashed line splitting the model vertically indicates the *Bayesian cut* that is used to stop inference flowing from predictions to voter performance parameters (see Section 3.3). We refer to the left side of the cut as the *vote model* and the right side as the *score model*.

Variable	Description
Y_i	Latent true binary label for item i
P_i	Latent vote probability for item i
U_i	Latent Thurstonian utility for item i
X_i	Predictor score (raw prediction) for item i
V_i	Count of positive votes received by item i
T_i	Observed golden set label for item i ($\emptyset$ if unobserved)
W_{kl}	Pairwise comparison counts (wins) between vote set k and l
ϕ	Base rate: prior probability of a positive true label ($Y_i = 1$)
a_0, d_0	Beta parameters controlling P_i when $Y_i = 0$
a_1, d_1	Beta parameters controlling P_i when $Y_i = 1$
μ	Mean utility shift capturing tendency to rank positives above negatives
δ	Utility shift capturing predictor's tendency to give high scores based on high vote probability P_i

Table 1: Summary of notations

3.2.1 Vote Model

The vote model follows the item-difficulty beta-binomial model of Carpenter (2008):

$$\begin{aligned} Y_i &\sim \text{Bern}(\phi) \\ P_i \,|\, Y_i &\sim \text{Beta}(a_{Y_i} d_{Y_i}, (1 - a_{Y_i}) d_{Y_i}) \\ V_i \,|\, P_i &\sim \text{Binomial}(P_i, c) \end{aligned}$$

Each item has a latent vote probability P_i drawn from one of two beta distributions (depending on the true label Y_i). Then, V_i (the number of positive votes an item receives) follows a binomial distribution[2] with rate P_i. The a_y terms allow us to model overall voter performance in terms of True Positive Rate (TPR) $P(V_{ij} = 1 \,|\, Y_i = 1) = a_1$ and True Negative Rate (TNR) $P(V_{ij} = 0 \,|\, Y_i = 0) = 1 - a_0$. These are given a hyperprior that is uniform over the triangle defined by $0 \le a_0 \le a_1 \le 1$. This ensures that voters are not negatively predictive on average; they are at least as likely to give a positive vote to a positive item as they are to give a positive vote to a negative item (however, there may still be individual items where voters are negatively predictive).

The beta scale parameters d_y control the correlation between pairs of votes on the same item. Conditional on the true label Y_i, this correlation in the beta-binomial model is $\lambda_y = \frac{1}{1+d_y}$. As $d_y \to \infty$, votes become conditionally independent given the true label, matching the model of Dawid and Skene (1979). As $d_y \to 0$ all voters on an item vote the same way. In practice, conditional independence does not usually hold. For example, in Task 2 (introduced in Sec. 5) the first two voters agree 17% more often than would be expected (given their observed TPR and TNR) if their votes were conditionally independent. We place a hyperprior on the correlation λ_y that is uniform on $[\epsilon, 1 - \epsilon]$, with $\epsilon = 0.01$ chosen to avoid inference instability that can occur near correlations of zero and one.

We can marginalize out P_i; the resulting likelihood for any single item in vote set $\mathcal{V}_k$ is a mixture of beta-binomial likelihoods:

$$\mathcal{L}_k = P(\mathcal{V}_k \,|\, \phi, a_0, d_0, a_1, d_1) = \sum_{t \in \{0,1\}} T^{(t)}(\mathcal{V}_k) \phi^t (1 - \phi)^{1-t} \, f_{\text{BB}}\left(V(\mathcal{V}_k);\, a_t d_t, (1 - a_t) d_t, c\right)$$

where f_{BB} is the beta-binomial density. The complete vote model log-likelihood is therefore $\sum_k N_k \log(\mathcal{L}_k)$. Note that the number of computations required to compute the log-likelihood is proportional to the number of vote sets, not the number of items.

3.2.2 Score Model

The scores produced by machine learning models cannot easily be summarized by a family of distributions. For example, LLM-based classifiers often produce scores

[2]We assume a fixed number of votes per item for clarity. However, in some applications votes are collected in a *best-of-k* fashion (i.e., collected until an unassailable $\lceil \frac{k}{2} \rceil$-vote majority is reached). This changes the binomial coefficients, but because these are constant w.r.t the parameters the posterior is exactly equivalent to the binomial case.

(corresponding to the first response token's probability of being affirmative) extremely close to zero or one, while a logistic regression with discrete features could produce a score distribution with many modes. Rather than trying to model raw scores in a general way, we choose to model the *ranking* of items induced by the scores. Since common methods for evaluating binary predictions—AUC, precision-recall curves, other thresholded metrics like accuracy—depend only on the ranking of items by the model, modeling score rankings is sufficient for our goal of predictor evaluation.

We use a Thurstonian (Thurstone, 1927) model of ranking where each item has a latent, normally-distributed *utility* U_i, and $X_i > X_j \iff U_i > U_j$. Specifically:

$$U_i \,|\, Y_i, P_i \sim \mathcal{N}(\mu Y_i + \delta P_i,\, 1)$$

Allowing the mean to vary by Y_i and P_i allows us to model two phenomena. Most obviously, positive μ captures the tendency of the predictions to rank positives above negatives. When $\delta = 0$, μ is directly related to the AUC of the predictions: $\text{AUC} = P(X_i > X_j \,|\, Y_i = 1, Y_j = 0) = \Phi\left(\frac{\mu}{\sqrt{2}}\right)$.

Positive δ captures the tendency of the predictor to give high scores to items that voters are likely to give positive votes to, regardless of the true label. This is important because predictors are often trained on votes rather than golden set labels, so they can learn the same systematic errors that voters exhibit.[3]

Unlike the vote model, there is no closed form marginalization of the latent variables U_i and P_i in the score model. However, for the method to scale to tens of thousands of items, we do not want to have to perform Bayesian inference over a linear number of latent variables. Furthermore, computing the likelihood of a full ranking of n items under the Thurstonian model requires evaluating the $(n-1)$-variate normal CDF, which is also infeasible to do many times per MCMC proposal.

To address these issues, we summarize the full ranking as a matrix of pairwise comparisons between items with different vote sets. The counts of *wins* W_{kl} – the number of times an item from vote set k is ranked above an item from vote set l by the predictor – are Mann-Whitney U statistics that are asymptotically normally distributed:

$$W_{kl} \sim \mathcal{N}(w_{kl} C_{kl},\, \sigma_{kl})$$

The *win rate* w_{kl} can be computed under the score model. Consider one pair of items (i, j) with known $P_i = p_i$ and $Y_i = y_i$. U_i and U_j are independently normally distributed, so the probability i is ranked above j is:

$$\begin{aligned} g(y_i, y_j, p_i, p_j) &\coloneqq P(U_i > U_j \,|\, y_i, y_j, p_i, p_j) \\ &= \Phi\left[\frac{\mu(y_i - y_j) + \delta(p_i - p_j)}{\sqrt{2}}\right] \end{aligned}$$

Marginalizing out the latent variables requires summing over all four possible (Y_i, Y_j) pairs and computing a double integral over beta densities:

$$w_{kl} = P(X_i > X_j) = P(U_i > U_j) = \sum_{\substack{y_i \in \{0,1\} \\ y_j \in \{0,1\}}} q_k(y_i) q_l(y_j) \iint_{p_i,\, p_j} g(y_i, y_j, p_i, p_j) f_k(p_i, y_i) f_l(p_j, y_j),$$

where f_k is the posterior beta density $P_i \,|\, \mathcal{V}_i = k, Y_i = y$ after conditioning on the votes in vote set k:

$$f_k(p, y) = f_{\text{Beta}}(p; a_y d_y + V(k), (1 - a_y) d_y + c - V(k))$$

and q_k is the posterior probability $P\left(Y_i = y \,|\, \mathcal{V}_i = k\right)$:

$$q_k(y) \propto^{\dagger} T^{(y)}(k) \phi^y (1 - \phi)^{1-y} f_{\text{BB}}(V(k);\, a_y d_y, (1 - a_y) d_y, c)$$

The double integral can be efficiently and accurately approximated with 2-D Gauss-Jacobi quadrature. Finally, we approximate $\sigma_{kl} = f_\sigma(w_{kl}, C_{kl}, N_k, N_l)$ using the parametric approach of Hanley and McNeil (1982).

To summarize, the pseudo-log-likelihood used for inference in the score model sums a Gaussian log-likelihood for each pair of vote sets:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_k \sum_{l<k} \log\left[f_{\mathcal{N}}(W_{kl};\, C_{kl} w_{kl}(\boldsymbol{\theta}), \sigma_{kl}(\boldsymbol{\theta}))\right]$$

where $\boldsymbol{\theta} = (\phi, a_0, d_0, a_1, d_1, \mu, \delta)$. Note again that the number of computations required to compute the score model log-likelihood does not depend on the number of items (only the number of vote sets). Thus, MAVEN can achieve constant[4] posterior inference time regardless of dataset size.

We use weakly informative priors: $\mu \sim \text{InvGamma}(1, 1)$ and $\delta \sim \text{Laplace}(0, 10)$. The former ensures that μ is positive, meaning that ground-truth positives have, on average, higher scores than ground-truth negatives.

3.3 Inference Using a Bayesian Cut

In Bayesian inference, one would typically use an algorithm like MCMC to sample from the joint posterior

[3] This is why the crowd-sourced majority vote estimate of AUC in Fig. 1 overestimates the model's true performance.

[†] The normalizer equals $q_k(1) + q_k(0)$, but is excluded for brevity.

[4] Posterior inference with MCMC makes up most of MAVEN's runtime, though of course the time required for imputation and metric estimation (see Sec. 3.4) still increases with n.

of the parameters given the observed data. However, because our goal is to use the votes and golden set labels to *measure* the performance of the predictor, joint inference is not appropriate. We only want inference to "flow" one way—we want to use the votes and labels to estimate the performance of the predictions, but we do not want to use the predictions to update our estimates of voter performance. This is achieved with a *Bayesian cut* (Plummer, 2015). First, we sample s draws from the posterior of the parameters in the *vote model*:

$$\boldsymbol{\theta}_i^V = (\phi, a_0, d_0, a_1, d_1)_i \sim P(\phi, a_0, d_0, a_1, d_1 \,|\, \mathcal{D}),$$

then we draw from the posterior of the *score model*, conditioning on the vote model samples:

$$\boldsymbol{\theta}_i^S = (\mu, \delta)_i \sim P(\mu, \delta \,|\, \boldsymbol{\theta}_i^V, \mathcal{D}).$$

Sampling is conducted in two stages using the No-U-Turn Sampler (NUTS) (Hoffman et al., 2014). The first stage (sampling from the vote model posterior) proceeds as normal. In the second stage, we follow Plummer (2015) and condition on new vote model parameters $\boldsymbol{\theta}_i^V$ every b NUTS samples, allowing for a "burn-in" phase for each new parameter set[5]. Only the final score model sample of each phase is kept; the resulting pair $\boldsymbol{\theta}_i = (\boldsymbol{\theta}_i^V, \boldsymbol{\theta}_i^S)$ is one complete sample from the Maven model.

A further benefit of conditioning on the vote model parameters is computational. The beta densities in the integration depend only on the vote model parameters, while the integration is only computed when sampling from the score model. Thus, Gauss-Jacobi roots and weights can be pre-computed for each vote model sample, before score model inference. If joint inference was conducted instead, the roots and weights—and their gradients with respect to the parameters—would have to be recomputed every step (up to thousands of steps per NUTS sample), drastically increasing inference cost.

3.4 Estimating Predictor Performance via Multiple Imputation

Finally, we transform the parameter samples from the model into estimates of predictor performance. This is done by imputing the unknown labels under the Maven model, creating s imputed datasets $\mathcal{D}^{(j)}$—one for each parameter sample $\boldsymbol{\theta}_j$:

$$\begin{aligned}
\mathcal{D}^{(j)} &= \{(X_i, V_i, \hat{Y}_i^{(j)})\}_{i=1}^n \\
\hat{Y}_i^{(j)} &\sim \text{Bern}(\hat{\gamma}_i(\boldsymbol{\theta}_j)) \\
\hat{\gamma}(\boldsymbol{\theta}_j) &= P(Y_i = 1 \,|\, \mathcal{V}_i, X_i; \boldsymbol{\theta}_j). \qquad (1)
\end{aligned}$$

[5]We found $b = 10$ was enough for the sampled distribution of $\boldsymbol{\theta}^S$ to converge.

Then, let M_k be the kth *metric function* $\mathcal{D} \to \mathbb{R}$ that computes a metric of interest (e.g., precision, recall, AUC) given a dataset. Point estimates and credible intervals for each metric can be computed from the sample mean and quantiles of the set of metric estimates $\left\{M_k\left(\mathcal{D}^{(j)}\right)\right\}_{j=1}^s$.

Evaluating Eq. 1 is complicated by the fact that conditioning on X_i induces a dependence between the U_i, since the U_i must match the ordering of the raw scores. Marginalizing out all U_i and P_i simultaneously is intractable; instead, we sample from the approximate distribution of the permutation of the order statistics of U_i that matches the ranking of X_i:

$$\begin{aligned}
(Z_1, ..., Z_n) &\sim \text{Dirichlet}((1, ...1)) \\
Q_i &= \sum_{j:X_j<X_i} Z_j \\
\hat{U}_i^{(j)} &= F_U^{-1}(Q_i; \boldsymbol{\theta}_j)
\end{aligned}$$

where $F_U^{-1}(Q; \boldsymbol{\theta}_j)$ is the quantile function for the unconditional distribution of U_i under the parameters. If the U_i were sampled i.i.d. from this unconditional distribution, $\hat{U}_i^{(j)}$ would be samples from the appropriate permutation of the order statistics. However, U_i are not identically distributed conditioned on $\mathcal{V}_i$, so the sampling is only approximate. In Appendix A we show that this approximation is accurate for even very small evaluation datasets (1000 items).

Given $\hat{U}_i^{(j)}$, the posterior probability that an item is positive can then be computed:

$$\begin{aligned}
&P(Y_i = y \,|\, U_i = \hat{U}_i^{(j)}, \mathcal{V}_i = k; \boldsymbol{\theta}_j) \propto \\
&\quad q_k(y) \int_p f_{\mathcal{N}}\left(\hat{U}_i^{(j)}; \mu y + \delta p, 1\right) f_k(p, y).
\end{aligned}$$

4 EVALUATIONS ON SIMULATED DATA

We begin our empirical studies with synthetic data simulated from known generative processes, which enables us to evaluate the effectiveness of Maven, compared with different baselines, under a wide range of settings. Rather than simulate data from the Maven generative process, we instead simulated from an Item-Response Theory (Baker, 2001) model to evaluate Maven under model misspecification. The data generation process is described in detail in Appendix B.1.

We compare Maven against multiple baselines listed in Table 2, each of which infers the true label $\hat{Y}_i$ for each item i from the observed data. Implementation details are provided in Appendix C.

Baseline		Description
No modeling	MV	Uses the majority vote: $\hat{Y}_i = \lfloor \frac{V_i}{c} \rceil$
	GT	Uses only the ground truth labels available $T_i \neq \emptyset$ as $\hat{Y}_i$
	GT+MV	Combines MV and GT: $\hat{Y}_i = T_i$ if $T_i \neq \emptyset$ otherwise $\hat{Y}_i = \lfloor \frac{V_i}{c} \rceil$.
Modeling votes	DS	Dawid and Skene (1979)
	ORACLE	An approach which has access to the true $P(T_i \mid V_i)$ as determined by the known generative process. $\hat{Y}_i$ is imputed according to these probabilities. This represents the theoretical upper bound for any vote modeling approaches estimating $P(Y_i \mid V_i)$.
Modeling votes and scores	DS+S	A model extending DS by modeling the target model's scores as a mixture of logit-normals (one component for each value of Y_i).
	H-AI	A Bayesian model (Human-AI Complementarity) designed to combine the predictions from humans and AI models (Steyvers et al., 2022).

Table 2: Baseline approaches used in our simulations

4.1 Evaluation Metrics

MAVEN's imputation approach allows it to estimate any metric. To illustrate this, we choose four metrics that measure a wide range of performance characteristics:

- **Human performance**, specifically the average True Negative Rate $\hat{\text{TNR}}_H$ of the crowd-sourced voters.
- **Model performance**, using $\hat{\text{AUC}}$ as a popular metric to measure the performance of the target model.
- **Human-Model performance difference**, comparing human $\hat{\text{TPR}}_H$ to the predictor's $\hat{\text{TPR}}_M(t^\star)$ at the threshold $t^\star$ where the TNR estimates for both model and human are the same:

$$\hat{\Delta}_{\text{TPR}}(t^\star) = \hat{\text{TPR}}_M(t^\star) - \hat{\text{TPR}}_H$$
$$\text{where} \quad \hat{\text{TNR}}_M(t^\star) = \hat{\text{TNR}}_H.$$

- **Model-Model performance difference**, measuring the difference in AUC for two target models A and B, $\hat{\Delta}_{\text{AUC}} = \hat{\text{AUC}}_A - \hat{\text{AUC}}_B$.

4.2 Results

Using the true values known from the simulation process, we compute the bias and credible interval (CI) width of the estimates from MAVEN and different baselines. Figure 4 summarizes the results when we vary the amount of ground truth labels available.

As expected across all metrics, GT, which only uses the ground truth labels available, provides estimates with small biases but large CIs. MV's estimates, on the other hand, have small CIs but large biases. By leveraging the ground truth labels, GT+MV's biases get smaller when more ground truth labels are available.

The other baselines all exhibit substantial bias, especially when a small fraction of items have ground truth labels. ORACLE, having access to the true $P(Y_i \mid V_i)$, can perfectly estimate $\hat{\text{TNR}}_H$. However, even such a perfect model of votes cannot measure predictor performance accurately, demonstrating the need for joint modeling of votes and scores.

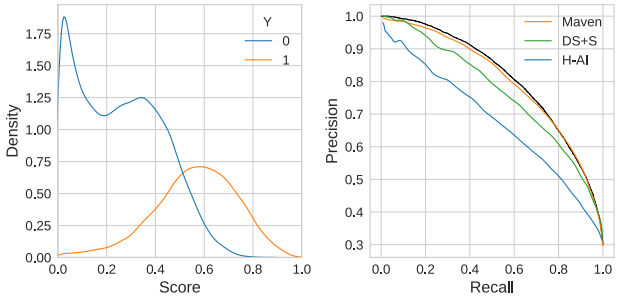


Figure 3: Left: simulated class-conditional score distributions, showing bimodality in the negative class and a heavy tail in the positive class. Right: estimated precision-recall curves for these scores, with the true curve in black. Only MAVEN is able to estimate the curve accurately.

DS+S and H-AI perform generally better than DS, especially in measuring model performance ($\hat{\text{AUC}}$) and human-model performance difference ($\hat{\Delta}_{\text{TPR}}(t^\star)$).

Finally, MAVEN's estimates consistently have small biases across all metrics, which track closely with those from GT. Moreover, the CI widths produced by MAVEN are substantially narrower than those from GT (shown in Figure 4's second row), demonstrating MAVEN's effectiveness in reducing estimate variance.

4.3 Robustness to Non-normal Scores

A key reason for modeling the score ranking, rather than the raw scores like Steyvers et al. (2022) and

Shanmugam et al. (2025), is to make MAVEN robust to different raw score distributions. Fig. 3 shows a dramatic example of MAVEN successfully modeling synthetic scores that deviate substantially from normality. Appendix D explores the distributional robustness of MAVEN in more detail.

5 EVALUATION ON REAL-WORLD DATA

MAVEN was developed to improve the performance estimation of production models used at Meta. In this section we present results on models developed for two binary tasks. Because of the sensitive nature of these tasks, we can only present them at a high level:

- TASK 1: Approximately 10% base rate. Voters have high TNR but lower TPR. The model is a boosted decision tree. The dataset contains 90k items, of which 23k[6] have expert labels.
- TASK 2: Approximately 75% base rate. Voters have moderate-high TPR and TNR. The model is an LLM. The dataset contains 358k items, of which 34k have expert labels.

In Sec. 5.2 we also evaluate MAVEN on the publicly-available Wikipedia Abusive Conversations dataset (Cécillon et al., 2020).

5.1 Measuring Estimator Quality

To measure the quality of MAVEN and the alternative approaches, we randomly construct a dataset $\mathcal{D}' \subset \mathcal{D}$ to have n items from the full dataset (ensuring n_e items have expert labels). The remaining expert-labeled items (denoted $\bar{\mathcal{D}}$, $\bar{\mathcal{D}} \cap \mathcal{D}' = \emptyset$) are used to compute the "true" metric value $m^* = M(\bar{\mathcal{D}})$, allowing us to evaluate the bias of the estimate produced by each method. Results are averaged over 10 dataset replicates.

In Fig. 1 we vary n_e (holding $n = 4000$) and demonstrate that MAVEN can reduce the CI width (compared to GT) on estimates of Task 1 performance without introducing bias. Table 3a tests MAVEN against other baselines and finds that they produce biased estimates of performance for the production model used in Task 2. In each case, results for the other task are very similar, despite the differences in base rate, voter quality, and model type between the tasks.

[6]Although the total number of expert labels per task is in the tens of thousands, these datasets were collected over a long period—too slowly to enable continuous monitoring of models deployed in production against expert labels (especially when restricted to segments of interest). In a given segment and time period of interest we will only have on the order of hundreds of expert labels available, so that is what we use in our experiments.

	$\hat{AUC}$	$\hat{TNR}_H$	$\hat{\Delta}_{TPR}(t^\star)$
MV	1.18 ± 0.04	4.47 ± 0.03	4.85 ± 0.07
DS	**0.34 ± 0.06**	3.30 ± 0.12	3.84 ± 0.14
H-AI	1.90 ± 0.23	3.23 ± 0.13	1.89 ± 0.28
DS+S	3.47 ± 0.10	2.89 ± 0.12	**0.35 ± 0.07**
GT	**1.00 ± 0.17**	**1.00 ± 0.18**	**1.00 ± 0.15**
MAVEN	**1.11 ± 0.21**	**0.76 ± 0.18**	**0.50 ± 0.14**

(a) Task 2

	$\hat{AUC}$	$\hat{TNR}_H$	$\hat{\Delta}_{TPR}(t^\star)$
MV	1.44 ± 0.26	3.19 ± 0.16	7.59 ± 0.36
DS	3.07 ± 0.26	2.55 ± 0.26	6.88 ± 0.27
H-AI	2.86 ± 0.13	**0.68 ± 0.17**	1.84 ± 0.15
DS+S	2.22 ± 0.15	2.06 ± 0.29	**0.33 ± 0.12**
GT	**1.00 ± 0.43**	**1.00 ± 0.32**	**1.00 ± 0.30**
MAVEN	**1.16 ± 0.21**	**1.01 ± 0.31**	**0.78 ± 0.37**

(b) Wikipedia Abusive Conversations

Table 3: Absolute bias relative to GT when estimating metrics for two real-world datasets ($n = 4000$, $n_e = 400$ in each case). Bolded entries indicate bias within 1 standard error of GT: in both datasets, all non-GT baselines show substantial bias on one or more metrics.

5.2 Wikipedia Abusive Conversations Dataset

The Wikipedia Abusive Conversations dataset (Cécillon et al., 2020) consists of 156k items, each annotated by 10 reviewers for whether it contains "toxic" content. Items are also assigned a probability of being toxic by the Perspective API toxicity model. To evaluate MAVEN on this dataset, 3 out of 10 annotations (randomly chosen) are provided to MAVEN as the crowd-sourced votes, while the "ground truth" (GT) for the item is determined by the majority vote of the other 7 annotators.

Table 3b shows the bias of different methods when estimating metrics using this dataset. Again, we see that MAVEN shows no more bias than using ground truth alone, while other methods are biased (often quite badly) on at least one metric.

6 DISCUSSION

Scaling Figure 5 depicts the time required by MAVEN compared to the alternative Bayesian approaches we tested. As expected, MAVEN's runtime is almost constant in the number of items used in the evaluation.

AutoEval MAVEN can complement AutoEval (Boyeau et al., 2025) when estimating an appropriate metric like model accuracy. To show this,

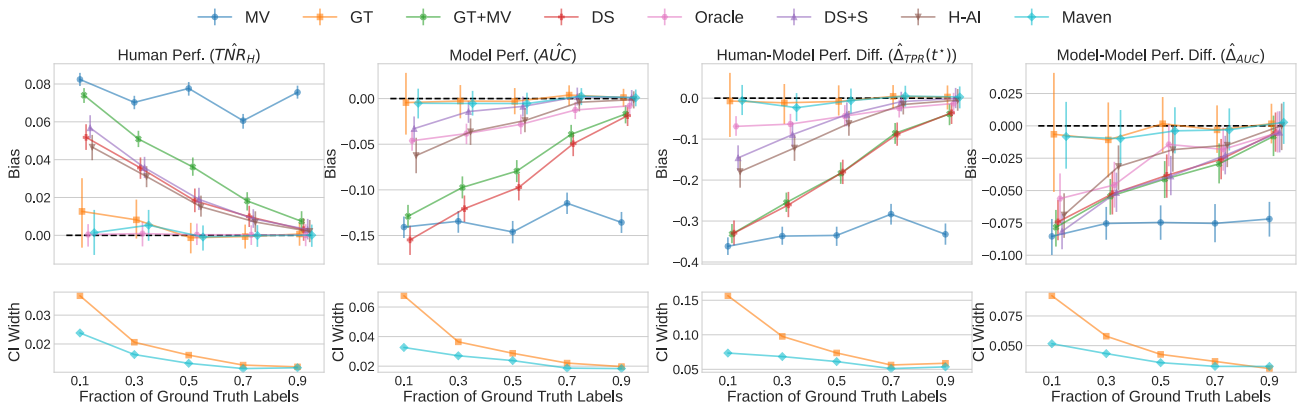


Figure 4: Bias and CI width when varying the amount of ground truth labels available. As we increase the fraction of ground truth labels, the bias and variance are expected to decrease. The bias plots show that MAVEN and GT perform best and track closely together, while MAVEN's CIs are significantly smaller than those from GT.

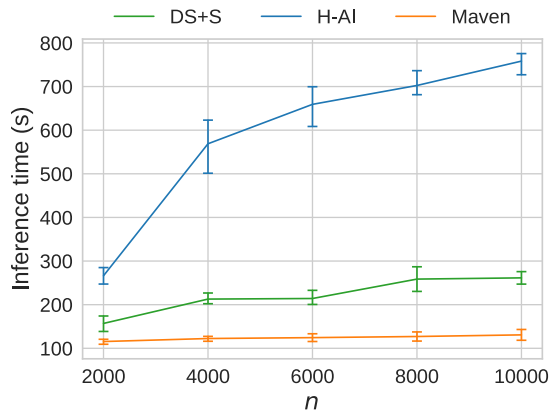


Figure 5: Compared to alternative Bayesian models of votes and scores, MAVEN achieves both a lower absolute runtime and more favorable scaling as the number of items increases.

we evaluated Task 1 model accuracy using AutoEval, provided with either the raw model scores (T_i, X_i) or the MAVEN posterior probabilities $(T_i, \hat{\gamma}_i)$. Across the different runs described in Sec. 5.1, we found that combining MAVEN and AutoEval resulted in an average of 13% less absolute bias and 36% narrower confidence intervals compared to AutoEval with raw scores alone. The methods are therefore strong complements to one another; MAVEN also inherits the frequentist guarantees of AutoEval when used in this way.

Labeling Efficiency To study the gain in labeling efficiency that MAVEN could enable (by reducing the reliance on scarce expert labels), we ran MAVEN on the Task 2 dataset, varying n and n_e. We found that a dataset with $n = 50k$ and $n_e = 16k$ produced the same estimates of $\hat{\Delta}_{\text{TPR}}(t^\star)$ (in terms of point estimate and CI width) as a ground-truth only estimate that used all $34k$ expert labels. Factoring in the relative labeling effort required to obtain expert versus crowd-sourced labels for this task, using MAVEN improved overall labeling efficiency by 42%.

Limitations and Future Work MAVEN performs well in real-world systems; however, there are two main extensions we plan to pursue. First, MAVEN cannot model predictors that are locally non-monotonic (where increasing score corresponds to *decreasing* probability of a positive) conditional on the votes. In many cases it is reasonable to have a strong prior of monotonicity—indeed, this limitation is shared by both of the Bayesian approaches we compare to. Future work will explore the possibility of enriching the MAVEN model to handle these cases.

Second, we hope to extend MAVEN to multiple classes and multiple scores. However, it is non-trivial to do this while maintaining the inference efficiency that we have achieved.

References

Angelopoulos, A. N., Bates, S., Fannjiang, C., Jordan, M. I., and Zrnic, T. (2023). Prediction-powered inference. *Science*, 382(6671):669–674.

Baker, F. B. (2001). *The basics of item response theory*. ERIC.

Baumann, J., Röttger, P., Urman, A., Wendsjö, A., del Arco, F. M. P., Gruber, J. B., and Hovy, D. (2025). Large language model hacking: Quantifying the hidden risks of using llms for text annotation.

Bavaresco, A., Bernardi, R., Bertolazzi, L., Elliott, D., Fernández, R., Gatt, A., Ghaleb, E., Giulianelli, M.,

Hanna, M., Koller, A., Martins, A., Mondorf, P., Neplenbroek, V., Pezzelle, S., Plank, B., Schlangen, D., Suglia, A., Surikuchi, A. K., Takmaz, E., and Testoni, A. (2025). LLMs instead of human judges? a large scale empirical study across 20 NLP evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–255, Vienna, Austria. Association for Computational Linguistics.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2019). Pyro: deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20(1):973–978.

Boyeau, P., Angelopoulos, A. N., Li, T., Yosef, N., Malik, J., and Jordan, M. I. (2025). Autoeval done right: Using synthetic data for model evaluation. In *Forty-second International Conference on Machine Learning*.

Carpenter, B. (2008). Multilevel Bayesian models of categorical data annotation. *Unpublished manuscript*, 17(122):45–50.

Choudhary, V., Marchetti, A., Shrestha, Y. R., and Puranam, P. (2025). Human-ai ensembles: When can they work? *Journal of Management*, 51(2):536–569.

Cécillon, N., Labatut, V., Dufour, R., and Linarès, G. (2020). WAC: A corpus of Wikipedia conversations for online abuse detection. In *12th Language Resources and Evaluation Conference*, pages 1375–1383, Marseille, FR.

Dawid, A. P. and Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*.

Dorner, F. E., Nastl, V. Y., and Hardt, M. (2025). Limits to scalable evaluation at the frontier: LLM as judge won't beat twice the data. In *The Thirteenth International Conference on Learning Representations*.

Gordon, M. L., Zhou, K., Patel, K., Hashimoto, T., and Bernstein, M. S. (2021). The disagreement deconvolution: Bringing machine learning performance metrics in line with reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., et al. (2024). A survey on llm-as-a-judge. *The Innovation*.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Hoffman, M. D., Gelman, A., et al. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., and Hovy, E. (2013). Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Kelly, M. (2025). *Machine Classifiers and Human Decision-Makers: Calibration, Perceptions, and Collaboration*. PhD thesis, University of California, Irvine.

Kelly, M., Boyd, A. J., Showalter, S., Steyvers, M., and Smyth, P. (2025). Bayesian inference for correlated human experts and classifiers. In *Forty-second International Conference on Machine Learning*.

Kerrigan, G., Smyth, P., and Steyvers, M. (2021). Combining human predictions with model probabilities via confusion matrices and calibration. In *Advances in Neural Information Processing Systems*.

Krumdick, M., Lovering, C., Reddy, V., Ebner, S., and Tanner, C. (2025). No free labels: Limitations of LLM-as-a-Judge without human grounding.

Kull, M., Filho, T. S., and Flach, P. (2017). Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *AISTATS*, pages 623–631.

Li, D., Jiang, B., Huang, L., Beigi, A., Zhao, C., Tan, Z., Bhattacharjee, A., Jiang, Y., Chen, C., Wu, T., Shu, K., Cheng, L., and Liu, H. (2025). From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V., editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2757–2791, Suzhou, China. Association for Computational Linguistics.

Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., and Liu, Y. (2024). LLMs-as-Judges: A comprehensive survey on LLM-based evaluation methods.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013). Learning with noisy labels. *Advances in Neural Information Processing Systems*, 26.

Nie, Y., Zhou, X., and Bansal, M. (2020). What can we learn from collective human opinions on natural language inference data? In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9131–9143, Online. Association for Computational Linguistics.

Passonneau, R. J. and Carpenter, B. (2014). The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Paun, S., Carpenter, B., Chamberlain, J., Hovy, D., Kruschwitz, U., and Poesio, M. (2018). Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Pavlick, E. and Kwiatkowski, T. (2019). Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

Peterson, J., Battleday, R., Griffiths, T., and Russakovsky, O. (2019). Human uncertainty makes classification more robust. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9616–9625.

Plummer, M. (2015). Cuts in Bayesian graphical models. *Statistics and Computing*, 25(1):37–43.

Ratner, A., De Sa, C., Wu, S., Selsam, D., and Ré, C. (2016). Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.

Ratner, A. J., Bach, S. H., Ehrenberg, H. E., and Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *JMLR*, 11:1297–1322.

Reif, Y. and Schwartz, R. (2024). Beyond performance: Quantifying and mitigating label bias in LLMs. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6784–6798, Mexico City, Mexico. Association for Computational Linguistics.

Reiss, F., Xu, H., Cutler, B., Muthuraman, K., and Eichenberger, Z. (2020). Identifying incorrect labels in the CoNLL-2003 corpus. In Fernández, R. and Linzen, T., editors, *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 215–226, Online. Association for Computational Linguistics.

Rodrigues, F., Pereira, F. C., and Ribeiro, B. (2014). Gaussian Process Classification and Active Learning with Multiple Annotators. In *ICML*.

Ruiz, P., Morales-Álvarez, P., Molina, R., and Katsaggelos, A. K. (2019). Learning from crowds with variational Gaussian processes. *Pattern Recognition*, 88:298–311.

Shanmugam, D., Sadhuka, S., Raghavan, M., Guttag, J., Berger, B., and Pierson, E. (2025). Evaluating multiple models using labeled and unlabeled data.

Showalter, S., Boyd, A. J., Smyth, P., and Steyvers, M. (2024). Bayesian online learning for consensus prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 2539–2547.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. *IEEE transactions on neural networks and learning systems*, 34(11):8135–8153.

Steyvers, M., Tejeda, H., Kerrigan, G., and Smyth, P. (2022). Bayesian modeling of human–AI complementarity. *Proceedings of the National Academy of Sciences*, 119(11):e2111547119.

Tan, Z., Li, D., Wang, S., Beigi, A., Jiang, B., Bhattacharjee, A., Karami, M., Li, J., Cheng, L., and Liu, H. (2024). Large language models for data annotation and synthesis: A survey. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, Florida, USA. Association for Computational Linguistics.

Thurstone, L. (1927). A law of comparative judgement. *Psychological Review*, 34:273–286.

Uma, A., Fornaciari, T., Hovy, D., Paun, S., Plank, B., and Poesio, M. (2020). A case for soft loss functions. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 8, pages 173–177.

Uma, A. N., Fornaciari, T., Hovy, D., Paun, S., Plank, B., and Poesio, M. (2022). Learning from disagreement: A survey. *J. Artif. Int. Res.*, 72:1385–1470.

Vasudevan, V., Caine, B., Gontijo Lopes, R., Fridovich-Keil, S., and Roelofs, R. (2022). When does dough become a bagel? analyzing the remaining mistakes on imagenet. *Advances in Neural Information Processing Systems*, 35:6720–6734.

Wei, J., Zhu, Z., Luo, T., Amid, E., Kumar, A., and Liu, Y. (2023). To aggregate or not? learning with separate noisy labels. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2523–2535, New York, NY, USA. Association for Computing Machinery.

Welinder, P., Branson, S., Perona, P., and Belongie, S. J. (2010). The multidimensional wisdom of crowds. In *NeurIPS*, pages 2424–2432.

Welinder, P., Welling, M., and Perona, P. (2013). A lazy man's approach to benchmarking: semisupervised classifier evaluation and recalibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3262–3269.

Whitehill, J., fan Wu, T., Bergsma, J., Movellan, J. R., and Ruvolo, P. L. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, pages 2035–2043.

Yan, Y., Rosales, R., Fung, G., Subramanian, R., and Dy, J. (2014). Learning from multiple annotators with varying expertise. *Mach. Learn.*, 95(3):291–327.

Zhang, J., Hsieh, C.-Y., Yu, Y., Zhang, C., and Ratner, A. (2022). A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*.

Zhou, D., Platt, J. C., Basu, S., and Mao, Y. (2012). Learning from the wisdom of crowds by minimax entropy. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2195–2203. Curran Associates Inc., Red Hook, NY, USA.

CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A APPROXIMATING THE ORDER STATISTICS

To test the accuracy of the approximation of the order statistics presented in Sec. 3.4, we simulate data from the MAVEN data generating process (Algorithm 1) with parameters $\boldsymbol{\theta}^*$, and compare the MAVEN-estimated posterior probabilities $\hat{\gamma}_i = P(Y_i = 1 \,|\, \mathcal{V}_i, X_i, \boldsymbol{\theta}^*)$ (where only the ranking of the X_i are used, not the raw values) with the true probabilities γ_i calculated with the raw X_i values (computed by setting $U_i = X_i$ in the equations in Sec. 3.4). Note that in both cases the true parameters $\boldsymbol{\theta}^*$ are used; the only difference is how we condition on the scores.

Figure 6 shows this comparison for simulated data with $\phi = 0.5$, $a_0 = 0.1$, $a_1 = 0.6$, $d_0 = 1.5$, $d_1 = 1.5$, $\mu = 0.5$, $\delta = 3$, $c = 3$, conditioning on scores at the 1st, 50th, and 99th percentile of the distribution of X_i, and on each of the four possible vote sets. With 100 items, the distribution of MAVEN estimates $\hat{\gamma}_i$ closely follows the true distribution of γ_i, though there are small differences. With 1000 items, however, these differences disappear and the approximation is essentially perfect.

Algorithm 1: Maven data generation process

Input: $\boldsymbol{\theta} = (\phi, a_0, a_1, d_0, d_1, \mu, \delta)$, c
foreach item i in $[1, n]$ **do**
  Draw $Y_i \sim \text{Bern}(\phi)$
  Draw $P_i \sim \text{Beta}(a_{Y_i} d_{Y_i}, (1 - a_{Y_i}) d_{Y_i})$;
  Draw $X_i \sim \text{Normal}(\mu Y_i + \delta P_i, 1)$;
  Draw $V_i \sim \text{Binomial}(P_i, c)$;

B SIMULATION DETAILS

B.1 IRT data generation procedure

Algorithm 2 describes in detail the generation process of the simulated data used in Sec. 4 of the main paper. Specifically, each human voter j is characterized by a *skill* S_j which is normally distributed with mean μ_d and variance σ_d. Similar to Algorithm 1, each item i has a true label Y_i generated from a Bernoulli distribution parameterized by the base rate ϕ. In addition, each item i has a normally distributed *difficulty* D_i. For each of the c votes that an item i receives, we randomly select a human voter j and the corresponding vote V_{ij} is generated from a Bernoulli distribution $\text{Bern}(P_{ij})$ parameterized by $P_{ij} = \sigma\left(o + (S_j - D_i)(2Y_i - 1)\right)$.

Figure 4 in the main paper reports the results for simulated data of size $n = 10000$ with $\phi = 0.2$, $\mu_s = 3.0$, $\sigma_s = 2.0$, $\mu_d = 0.5$, $\sigma_d = 2.0$, $o = -1.0$, and $c = 3$.

Algorithm 2: IRT data generation procedure

Input: $\boldsymbol{\theta} = (\phi, \mu_s, \sigma_s, \mu_d, \sigma_d, o)$, c
foreach voter j in $[1, n_v]$ **do**
  Draw $S_j \sim \text{Normal}(\mu_s, \sigma_s)$
foreach item i in $[1, n]$ **do**
  Draw $Y_i \sim \text{Bern}(\phi)$;
  Draw $D_i \sim \text{Normal}(\mu_d, \sigma_d)$;
  Draw $A_i \sim \binom{\{1,\ldots,n_v\}}{c}$;
  foreach voter j in A_i **do**
    Set $P_{ij} = \sigma\left(o + (S_j - D_i)(2Y_i - 1)\right)$ where σ denotes the sigmoid function;
    Draw $V_{ij} \sim \text{Bern}(P_{ij})$;
  Draw $X_i \sim \text{Normal}\Big(\mu T_i + \frac{\delta}{c} \sum_j P_{ij}, 1\Big)$;

C BASELINE IMPLEMENTATION DETAILS

The three Bayesian baselines we compare to—Dawid-Skene (DS), Dawid-Skene + Scores (DS+S), and Human-AI Complementarity (H-AI, Steyvers et al. (2022))—are implemented in Pyro (Bingham et al., 2019) and posterior samples are drawn using NUTS (Hoffman et al., 2014). 1000 warmup samples are used for each of 10 chains, before a total of 500 samples are drawn from the posterior (50 per chain). Each model produces per-item estimates of $\hat{\gamma}_i = P(Y_i = 1 \,|\, \mathcal{V}_i, X_i, \boldsymbol{\theta})$ in terms of the posterior parameters $\boldsymbol{\theta}$; metrics are calculated by imputing the missing labels according to these probabilities as detailed in Sec. 3.4. In the subsequent sections we describe implementation details specific to each baseline.

C.1 Dawid-Skene

Our implementation of Dawid-Skene exactly follows the vote model presented in Sec. 3.2.1, except P_i is fixed rather than drawn from a beta distribution: $P_i = a_{Y_i}$. Thus, we model all voters as having a single confusion matrix with $TPR = a_1$ and $TNR = 1 - a_0$. Without the beta distribution, parameters d_0 and d_1 are unnecessary. We use the same weak priors as the MAVEN vote model.

C.2 Dawid-Skene + Scores

To incorporate scores into the Dawid-Skene model, we model the scores as a mixture of Gaussians:

$$\begin{aligned} \mu_0, \mu_1 &\sim \text{Normal}(0, 1) \\ \sigma &\sim \text{HalfNormal}(1) \\ X_i \,|\, Y_i &\sim \text{Normal}(\mu_{Y_i}, \sigma)\,. \end{aligned}$$

If the observed scores are probabilities, we logit-transform them before applying this model. The rest of

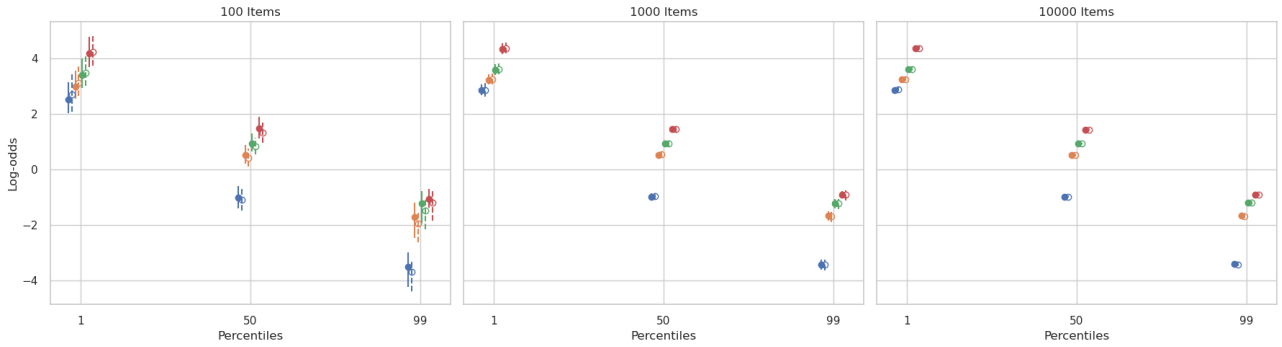


Figure 6: Distribution (mean and 95% CIs) of $\text{logit}(\hat{\gamma}_i(\boldsymbol{\theta}^*))$ computed by MAVEN compared to distribution of true $\text{logit}(P(Y_i = 1 \mid \mathcal{V}_i, X_i; \boldsymbol{\theta}^*))$, at different score percentiles and dataset sizes. Each color conditions on a different vote set $\mathcal{V}_i$ (from left to right: $V_i = 0$ through to $V_i = 3$). For evaluation datasets larger than 1000 items the MAVEN distributions match the true distributions very closely, even at extreme percentiles.

the model matches the Dawid-Skene model described above.

C.3 Human-AI Complementarity

Small modifications to the model presented by Steyvers et al. (2022) were required to make it work in our setting. The core of their model remains: the mixture of bivariate normal distributions that models the latent tendency of the human and the target model to give, respectively, positive labels and high scores:

$$(\lambda_{H,i}, \lambda_{M,i}) \mid Y_i \sim \text{Normal}\left(\begin{pmatrix}\mu_{H,Y_i}\\ \mu_{M,Y_i}\end{pmatrix}, \begin{pmatrix}\sigma_H^2 & \rho\sigma_H\sigma_M\\ \rho\sigma_H\sigma_M & \sigma_M^2\end{pmatrix}\right)$$

They model the scores as $X_i = \sigma(\lambda_{M,i})$; we omitted this sigmoid transformation when scores were already supported on the reals (as in Alg. 2 and Alg. 1) rather than $[0, 1]$. Otherwise, the model score part of their Bayesian model was left unchanged.

On the vote side, their setting uses a single human label per item (V_i), as well as a discrete score (R_i) measuring the human's confidence (low, medium, high). For binary tasks, these are modeled as follows:

$$\begin{aligned}\gamma_{H,i} &= \sigma(\lambda_{H,i})\\ V_i &\sim \text{Bern}(\sigma(2\gamma_{H,i} - 1))\\ R_i &\sim \text{OrderedProbit}(\gamma_{H,i}, s, \delta)\end{aligned}$$

(The repeated sigmoid is because the softmax is used to generate categorical distribution probabilities when Y_i can take more than 2 classes.)

Since our setting is binary, has multiple votes, and no confidence score, we simplified this to a probit model:

$$\begin{aligned}P_i &= \Phi(\lambda_{H,i})\\ V_i &\sim \text{Binomial}(P_i, c)\,.\end{aligned}$$

We use the following priors:

$$\begin{aligned}\phi &\sim \text{Unif}(0, 1)\\ \mu_{H,0} &\sim \text{Normal}(0, 1)\\ \mu_{M,0} &\sim \text{Normal}(0, 1)\\ \mu_{H,1} - \mu_{H,0} &\sim \text{HalfNormal}(1)\\ \mu_{M,1} - \mu_{M,0} &\sim \text{HalfNormal}(1)\\ \rho &\sim \text{Unif}(\epsilon - 1, 1 - \epsilon)\\ \sigma_H &= 1\\ \sigma_M &\sim \text{HalfNormal}(1)\end{aligned}$$

The differences between means μ are constrained to be positive to ensure that voters are more likely to give positive labels to positives than negatives, and that higher model scores correspond to a higher probability of a positive.

D DISTRIBUTIONAL ROBUSTNESS

We use the following simulation approach to test MAVEN's ability to handle score distributions that deviate from the mixture of normal distributions used to model the latent utility. First, we select parameters a, b, and m that describe a beta calibration curve (Kull et al., 2017). Then we draw scores X_i from a mixture of beta distributions, and transform them according to the beta calibration function. Finally, we draw labels Y_i according to the calibrated probabilities. The resulting conditional distributions $X_i \mid Y_i = 0$ and $X_i \mid Y_i = 1$ (with CDFs F_0 and F_1 respectively) can take a wide variety of shapes (see Fig. 7), while the beta calibration step ensures that the probability of a positive label is monotonic in the scores—which is expected by the methods we test.

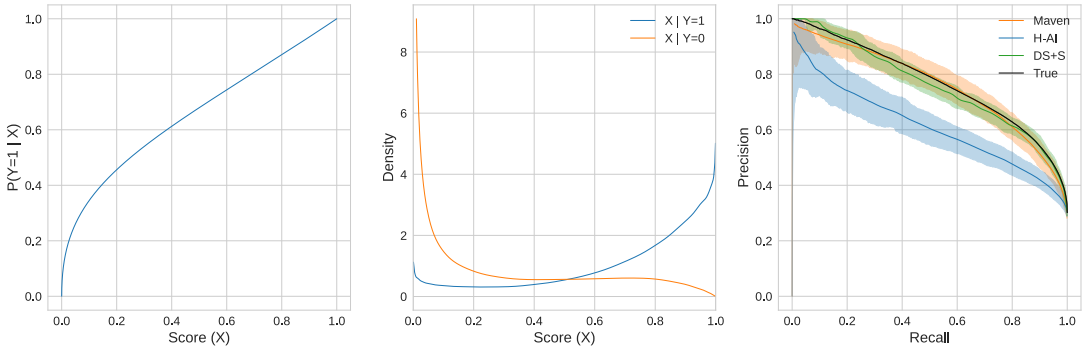
To generate a complete dataset of scores and votes,

we use Algorithm 3. Votes are drawn from the beta-binomial model used by MAVEN, while a Gaussian copula is used to induce correlation between the scores X_i (with class-conditional marginal distributions defined by F_0 and F_1) and the latent vote probability P_i.

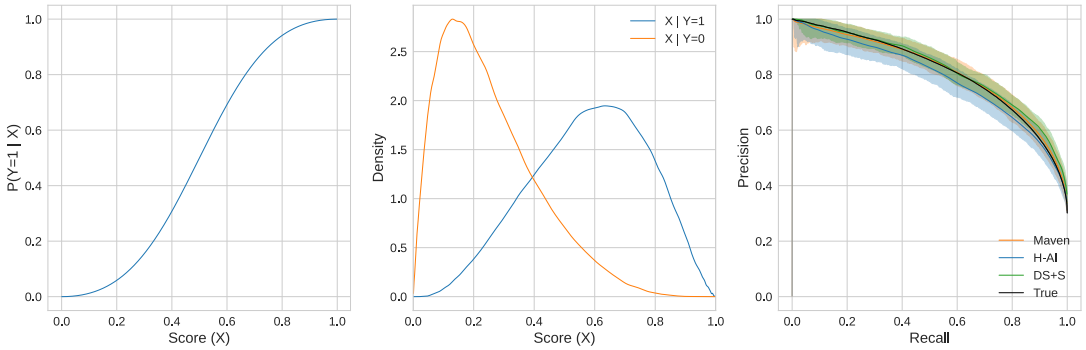
Algorithm 3: Data generation procedure to test robustness to different score distributions

Input: ϕ, a_0, a_1, d_0, d_1, ρ, F_0, F_1
foreach item i in $[1, n]$ **do**
  Draw $Y_i \sim \text{Bern}(\phi)$
  Draw $(U_i^P, U_i^X) \sim \text{Copula}_{\text{Gauss}}(\rho)$;
  Set $P_i = F^{-1}_{\text{Beta}}(U_i^P; a_{Y_i} d_{Y_i}, (1 - a_{Y_i}) d_{Y_i})$;
  Set $X_i = F^{-1}_{Y_i}(U_i^X)$;
  Draw $V_i \sim \text{Binomial}(P_i, c)$;

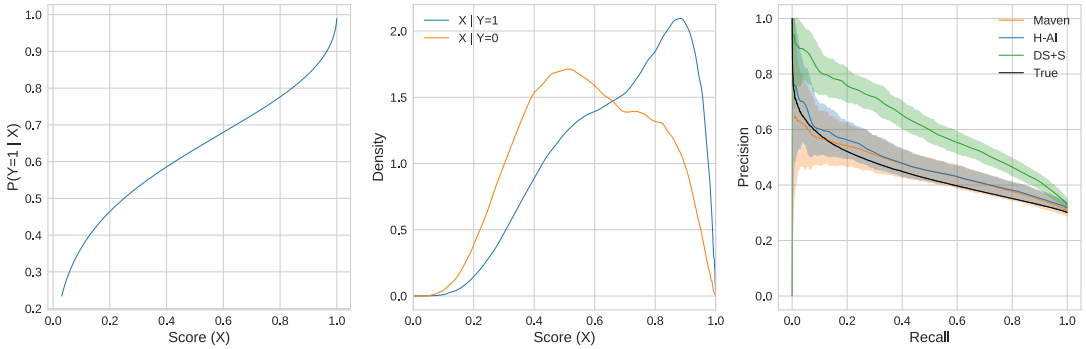
Figure 7 shows results on four different score distributions. Only the score distributions change between plots, the other parameters are held at $\phi = 0.3$, $a_0 = 0.2$, $a_1 = 0.4$, $d_0 = 1.5$, $d_1 = 1.5$, $\rho = 0.5$.



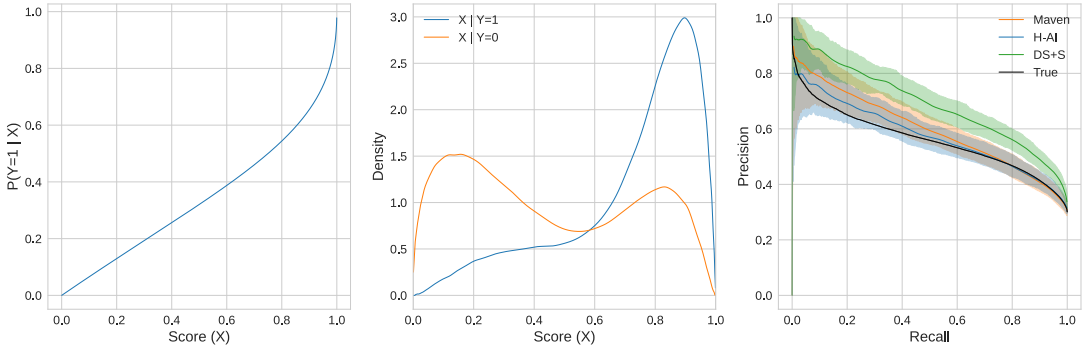
(a) $a = 0.5, b = 1, m = 0.25, q = 0.5, \phi_0 = 0.3, \phi_1 = 0.8, d_0 = 1, d_1 = 5$



(b) $a = 2, b = 2, m = 0.5, q = 0.5, \phi_0 = 0.2, \phi_1 = 0.5, d_0 = 10, d_1 = 5$



(c) $a = 0.5, b = 0.5, m = 0.25, q = 0.5, \phi_0 = 0.5, \phi_1 = 0.8, d_0 = 10, d_1 = 10$



(d) $a = 1, b = 0.5, m = 0.75, q = 0.5, \phi_0 = 0.3, \phi_1 = 0.8, d_0 = 5, d_1 = 10$

Figure 7: Comparison of different Bayesian models combining scores and votes across various score distributions. Left column: beta calibration curve $P(Y_i = 1 \,|\, X_i)$. Center column: class-conditional score distributions $X|Y = y$. Right column: precision-recall curves estimated by each model using $n = n_e = 4000$ synthetic samples (generated using the parameters in the sub-captions). The true precision-recall curves are shown in black.