# Who You Gonna Call? Optimizing Expert Assignment with Predictive Models

**Cyrus Cousins, Neha Nayak Kennard, Sheshera Mysore, Justin Payan[*], and Yair Zick[†]**
University of Massachusetts Amherst
{ccousins, kennard, smysore, jpayan, yzick}@umass.edu

## Abstract

In open-ended tasks that require domain expertise, how do we assign experts that will perform the task well? Important use-cases like community question answering, peer review, hiring committees, and crowd-sourcing platforms all require assigning experts from a pool of users. Ideally, we would assign experts to tasks so as to optimize overall task performance, but expert performance is unknown prior to making the assignment. We propose predicting these performance metrics and assigning using the predictions. Using an expert-assignment task derived from StackExchange, we show that explicitly predicting expert performance has a large impact on assignment decisions and can improve overall welfare. We demonstrate this claim using both theoretical bounds on statistical generalization guarantees and automated metrics of assignment quality. This work highlights the effectiveness of predictive assignment, and the need to collect high quality datasets linking pre- and post-allocation measures in other important expert assignment tasks.

## 1 Introduction

In the knowledge economy, many important institutional processes draw from pools of experts to accomplish tasks requiring specialized knowledge at large scales. These tasks include community question answering, reviewer assignment for peer review and grant funding allocation, and peer grading in massive online open coursework (MOOCs). In each of these applications, we can use historical performance measures, as well as other available features of experts and tasks, to predict expert performance on each task and assign experts appropriately. However, experts have limited time and resources, and can typically only respond to a small number of tasks at a time. We must be able to both predict expert performance, and use those predictions effectively to assign the limited resources of the experts.

Community question answering sites have existed for decades, serving as repositories of high-quality information and enabling users to connect with domain experts rapidly. However, many of these questions remain unanswered. For example, as of December 2023 the CS StackExchange has $12{,}962$ total questions with score at least 2, of which roughly $10\%$ ($1{,}231$) have no answer. Over $5$ million questions remain unanswered in StackOverflow [27]. These unanswered questions could be addressed if we prompt users to answer them, but each user must be matched to questions they have expertise in answering. We also must select users that have a track record of providing high quality answers, while not overloading any individual user. The same fundamental problem appears in the context of academic peer review [6, 10, 16, 25], as well as in peer evaluation of course assignments in MOOCs [1, 8, 14, 22]. Correctly trading-off among these factors (past user competence, user expertise, and user workload) is a complicated endeavor.

[*]Corresponding author.
[†]Authors are listed in alphabetical order.

Preprint. Under review.

Question recommendation in StackExchange has long incorporated components that predict downstream answer quality, either through predicting whether a user's future answer will be accepted [12, 29], or by predicting the score of the user's future answer [26, 27]. These recommendation algorithms do not consider the trade-offs that arise from users' time limitation constraints. Meanwhile, in peer review and other peer evaluation problems, existing methods assign experts using proxies of downstream task performance, which we assume correspond to task performance but often have not been verified. In peer review, these proxies are typically document-based similarity measures, reviewer bids, and keyword matching scores [11]. While trade-offs in assignments have been well-studied in this context [10, 16, 25], to date there has only been a single study showing any link between these intuitive *proxies* of performance and actual task performance [20].

In this work, we frame the expert assignment task as the combination of performance prediction and optimal assignment. We select a single target outcome of interest, which measures expert task performance. These outcomes are labeled from among a set of outcome labels. We then train a model to predict the outcome when assigning different experts to a new problem. By estimating the probabilities of each outcome, we can easily trade-off between different experts for a given task. More importantly, these probabilities help us assign many experts to many tasks at scale.

### 1.1 Contributions

Our contributions are as follows:

- we rigorously investigate feature importance of a logistic regression model trained to predict probability of upvotes, showing the strength of historical performance measures in particular (Section 4.2),
- we derive theoretical high-probability bounds for assignment quality under our predictive model (Theorem 3.1),
- we demonstrate experimentally that assignments made under the predictive model have higher worst-case predicted quality and assign users with a stronger historical track record than baselines (Section 5),
- and we study the stability of and correlations between 11 different automated metrics of assignment quality, enabling a nuanced understanding of the interactions between expert assignment measures and outcomes (Section 5.1).

### 1.2 Related work

Many existing works learn to recommend users for answering questions on community question answering (CQA) forums like StackExchange, Quora, and Yahoo Answers. Sun et al. [26] predict the future user-voted score when assigning a user to a given question. Our work additionally contributes rigorous feature importance analysis, updated NLP techniques, and examination through the lens of constrained assignment. Tondulkar et al. [29] predict which user's answer will be marked as accepted by the original question poster. Yang et al. [31] recommend users to questions with topics that interest them and in which they have expertise, irrespective of the user's competence. Qian et al. [17] study the setting where experts are sent requests for work, and they want to maximize the topical similarity of recommended experts as well as the acceptance rate of the invitations. While much attention has been paid to expert recommendation and prediction of answer quality in CQA forums, none of these works have addressed the constraints of the experts being assigned, or trade-offs among the many valid measures of answer quality in CQA.

A particularly thorny aspect of predictive expert assignment is identifying metrics for answer quality. Zhu et al. [33] asked both users and subject-matter experts to give a list of important criteria for evaluating answer quality, identifying 13 major criteria for answer quality measurement. They also asked experts to label answers as satisfying or not satisfying each criterion, and finally to rate each answer as good or bad overall. These and other similar criteria have been used or rediscovered in other important CQA studies [7, 21]. We employ them as features in Section 4, and use them as secondary evaluation metrics in Section 5.

Some prior work attempts to predict missing bids for reviewer assignment, and to use these predicted bids to compute and evaluate assignments [5, 19]. Charlin et al. [4] also evaluate reviewer assignments under a predictive model imputing missing relevance scores. Although these works study the quality

of assignment under a predictive model, none assigns using predictions of final task completion quality. Preference information and content-based relevance scores can be important factors in performance prediction as shown in Section 4.2.

Saveski et al. [20] develop a model for counterfactual evaluation of alternative reviewer assignments on past conferences. They measure review quality by the reviewers' self-reported expertise and confidence measures, and demonstrate that textual similarity measures (like the Toronto Paper Matching System [2, 3]) are more directly relevant to assigning confident and (self-reported) expert reviewers than bids and keywords.

## 2 The Approach

We wish to assign experts from a pool of $m$ experts $E$ to tasks from a pool of $n$ tasks $Q$. The experts reply to the tasks simultaneously, and have limited time. Each expert $e \in E$ can be assigned to up to $u_e$ tasks to ensure a reasonable workload, and each task must be assigned at least $v_q$ experts to get a variety of perspectives.

Every expert $e \in E$ who is assigned to the task $q \in Q$ *responds* to the task. These responses are labeled with a *quality label* by one or multiple individuals, perhaps users of our community question answering platform or meta-reviewers in the context of reviewer assignment. To determine a final *quality score* for the expert's response to the task, we map the labels to real values and take the expected value of the labels. We use a two-stage approach where quality labels map to quality scores because answer quality labeling tasks are typically discrete. Once we have collected the labels, a central decision-maker weights the labels using their domain knowledge.

Denote the set of quality labels as $L \doteq \{l_1 \dots l_k\}$. For every expert-task pair $(e, q)$, there is a ground truth probability distribution over labels $p(l_i|(e,q))$; $p(l_i \mid (e,g))$ denotes the likelihood that an annotator will assign the label $l_i$ to expert $e$'s response to task $q$. We write $l(e,q)$ to denote the random variable that is the label for $(e,q)$. $l(e,q)$ thus represents a single draw from $p(l_i|(e,q))$, or the label provided by a random annotator for $e$'s response to $q$. Let function $f : L \to \mathbb{R}$ denote the numerical value of each label. The quality score of expert $e$'s answer to task $q$ is computed as $\mathop{\mathbb{E}}_{l \sim p(l_i|(e,q))}[f(l)]$.

An *assignment* $A$ is an $m \times n$ matrix whose entries are in $\{0,1\}$, where the entry $A_{e,q} = 1$ if and only if the expert $e$ is assigned to the task $q$. As we describe earlier, the number of tasks assigned to the expert $e$ is upper bounded by $u_e$, and thus for every expert $e \in E$, $\sum_{q \in Q} A_{e,q} \leq u_e$. Similarly, since each task $q \in Q$ receives at least $v_q$ experts, we have $\sum_{e \in E} A_{e,q} \geq v_q$. Let $W(A) \doteq \sum_{q \in Q} \sum_{e \in E} A_{e,q} \mathop{\mathbb{E}}_{l \sim p(l_i|(e,q))}[f(l)]$ denote the welfare of assignment $A$. Let $\mathcal{A} \doteq \left\{ A \in \{0,1\}^{m \times n} \mid \sum_{q \in Q} A_{e,q} \leq u_e, \sum_{e \in E} A_{e,q} \geq v_q \right\}$. Our goal is to select an assignment $A \in \mathcal{A}$ that maximizes $W(A)$. Thus, when $p$ is fully known for all $(e,q)$, our goal is to solve the optimization problem $\max_{A \in \mathcal{A}} W(A)$.

**Example: StackExchange** In StackExchange, users upvote and downvote the answer provided by $e$ to question $q$. In that domain, we have $L = \{\text{Upvote}, \text{Downvote}\}$. The true $p(\text{Upvote}|(e,q))$ cannot be known, but we can estimate it by computing the empirical conditional distribution over votes. Under this estimate, $p(\text{Upvote}|\text{Vote}, (e,q))$ is computed for $e$'s answer to $q$ as $\frac{\#\text{Upvotes}}{\#\text{Votes}}$. Likewise, $p(\text{Downvote}|\text{Vote}, (e,q))$ can be estimated as $\frac{\#\text{Downvotes}}{\#\text{Votes}}$. One reasonable label value model is to set $f(\text{Upvote}) = 1$ and $f(\text{Downvote}) = -1$, similarly to how answer scores are computed for display on the site. However, upvotes are free and only require 15 reputation to cast, while downvotes are only available to users with at least 125 reputation and cost 1 reputation to cast.[1] The decision-maker could therefore decide to weight downvotes more than upvotes, using $f(\text{Upvote}) = 1$ and $f(\text{Downvote}) = -5$. Under this second model, if $e$ leaves an answer for $q$ receiving 11 upvotes and 1 downvote, the quality score for the answer is $\mathop{\mathbb{E}}_{l \sim p(l_i|(e,q))}[f(l)] = \frac{11}{12}(1) + \frac{1}{12}(-5) = .5$.

[1] `https://stackexchange.com/tour`

**Example: Reviewer Assignment** In reviewer assignment, we might ask meta-reviewers to label the quality of the reviews, using the label set $L = \{$Meets Expectations, Exceeds Expectations, Fails to Meet Expectations$\}$. When there is only one meta-reviewer per paper, the true probability of a label is estimated as 1 if assigned and 0 otherwise. A conservative conference organizer may set $f(\text{Fails to Meet Expectations}) = -10$, $f(\text{Meets Expectations}) = .8$, and $f(\text{Exceeds Expectations}) = 1$. Under this model, the quality score for any review is in the set $\{-10, .8, 1\}$, since the distribution over labels is deterministic for each review.

## 3 Learning to Predict Response Quality

The true label distribution $p^*(l_i|(e,q))$ (and thus the true welfare function $W^*$) is often unknown prior to making our assignment of experts to tasks. We must estimate a distribution $\hat{p}(l_i|(e,q))$ on existing data, and then apply that learned estimate to make our assignments. We optimize for the same welfare objective, but with the estimated $\hat{p}$. We define $\hat{W}(A) \doteq \sum_{q \in Q} \sum_{e \in E} A_{e,q} \mathop{\mathbb{E}}_{l \sim \hat{p}(l_i|(e,q))}[f(l)]$.

We select the assignment $A \in \mathcal{A}$ that maximizes $\hat{W}(A)$ over $\mathcal{A}$. This problem is a totally unimodular linear program, and thus the optimal binary solution can be found in polynomial time by relaxing the space of $A$ from $\mathcal{A}$ to $[0,1]^{|E|\times|Q|}$ and solving the corresponding continuous linear program [28]. The optimal continuous solution corresponds to the optimal binary solution.

To learn $\hat{p}$, we estimate the model in our hypothesis class with the minimum *cross-entropy loss*. The cross entropy loss of $\hat{p}$ with respect to a distribution $p$ for a single expert-task pair $(e,q)$ is computed as $\mathbb{H}(p(l|(e,q)), \hat{p}(l|(e,q))) = -\sum_{l \in L} p(l|(e,q)) \log \hat{p}(l|(e,q))$. Given a set of $t$ expert-task pairs $T$, we can compute the cross-entropy loss of $\hat{p}$ with respect to $p$ over all $T$ as $\frac{1}{t} \sum_{(e,q) \in T} \mathbb{H}(p(l|(e,q)), \hat{p}(l|(e,q)))$.

We will train our model $\hat{p}$ by minimizing the cross-entropy loss on a training set, against the ground truth distribution $p^*$ constructed from known labellings. In the context of expertise assignment, the training data is often not sampled i.i.d. As we will see in Section 4, in StackExchange we typically construct features for $(e,q)$ pairs from user $e$'s past answers and the text of the question $q$. Therefore, when learning the model $\hat{p}(l|(e,q))$ for a question $q$, we use the previously-seen labelled pairs to construct the features for $(e,q)$. This will require a somewhat non-standard approach for estimating the generalization error of $\hat{p}$.

For analysis purposes, we will assume there exists a value $\gamma \in \mathbb{R}$ such that the cross-entropy loss cannot exceed $\gamma$ for any $e$ and $q$. This can be achieved by considering a smoothed labelling, such that each label $l$ has a minimum probability $\gamma_l$ under both the true and predicted probability distributions.

### 3.1 Bounding Assignment Quality

Once we have a predictive model for $\hat{p}$, we must output a decision using our predicted values. We can easily solve the problem using the theory of total unimodularity as previously described.

We obtain high probability bounds on the approximation error introduced from optimizing for $\hat{W}$ instead of $W^*$. We first evaluate our model $\hat{p}$'s performance on a test set $T$ sampled from distribution $\mathcal{D}_{\text{TEST}}$. Consider a test set $T$ containing $t$ expert-task pairs $(e,q)$. The cross-entropy loss of $\hat{p}$ on the test set is $\xi \doteq \frac{1}{t} \sum_{(e,q) \in T} \mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))$. Using the empirical loss $\xi$, we construct generalization bounds for any assignment $A$ using McDiarmid's bounded differences inequality and likelihood weighting. We cannot apply traditional generalization error bounds based on training set loss because our training set often does not consist of i.i.d. samples. Given an assignment $A$, let $T_A$ denote the set of $(e,q)$ pairs such that $A_{e,q} = 1$, $T_A \doteq \{(e,q) \in E \times Q \mid A_{e,q} = 1\}$. We assume that these pairs are also random variables drawn from some distribution $\mathcal{D}_A$, where the support of $\mathcal{D}_A$ is $E \times Q$. We define a matrix $\Lambda \in \mathbb{R}^{m \times n}$ such that

$$\Lambda_{e,q} \doteq \frac{\Pr_{(\mathbf{e},\mathbf{q}) \sim \mathcal{D}_{\text{TEST}}}((\mathbf{e},\mathbf{q}) = (e,q))}{\Pr_{(\mathbf{e},\mathbf{q}) \sim \mathcal{D}_A}((\mathbf{e},\mathbf{q}) = (e,q))} \ .$$

For any distribution $p$, let $H(p,\hat{p}) \in \mathbb{R}^{m \times n}$ be a matrix such that $H(p,\hat{p})_{e,q} = \mathbb{H}(p(l|(e,q)), \hat{p}(l|(e,q)))$. For matrices $X, Y \in \mathbb{R}^{m \times n}$, let $\langle X, Y \rangle_{\text{F}} \doteq \sum_{1 \le i \le m, 1 \le j \le n} X_{i,j} Y_{i,j}$ denote the Frobenius matrix product.

We are now ready to state the theorem governing the generalization error of our predictor $\hat{p}$. Our generalization error will be in terms of the empirical loss $\xi$ plus an additive error term depending on a confidence parameter $\delta$.

**Theorem 3.1.** *For any $\delta \in (0,1)$, the true probability distribution $p^*$ satisfies*

$$\frac{1}{v_q n}\langle A, \langle \Lambda, H(p^*, \hat{p})\rangle_F\rangle_F \leq \xi + \sqrt{\frac{\gamma^2 \ln \frac{1}{\delta}}{2}\left(\frac{1}{t} + \frac{1}{v_q^2 n^2}\sum_{(e,q)\in T_A} \Lambda_{e,q}^2\right)}$$

*with probability at least $1-\delta$.*

We prove Theorem 3.1 in Section 8.

Once we have a candidate assignment $A$ that maximizes $\hat{W}(A)$, we can apply Theorem 3.1 to give high probability lower and upper bounds on $W^*(A)$ by maximizing or minimizing $W(A)$ over the region defined by Theorem 3.1. This problem is a linear program and can be solved using off-the-shelf solvers.

## 4 Predicting Answer Quality

Our experiments focus on publicly available StackExchange data.[2] We study the computer science (`cs.stackexchange.com`), biology (`biology.stackexchange.com`), chemistry (`chemistry.stackexchange.com`), and academia (`academia.stackexchange.com`) StackExchanges. Results are presented in the main body of the text using the CS StackExchange; the other StackExchanges show similar patterns and results are reported in Section 10.

We represent users using their profiles and their past answers. Detailed statistics about the dataset times, number of questions and answers, etc are included in Section 9.

We target the community votes on answers as a measure of answer quality. Because users lose 1 reputation point for every downvote cast,[3] we view downvotes as a stronger signal of answer quality than upvotes. We set $f(\text{Upvote}) = 1$ and $f(\text{Downvote}) = -5$.

Altogether, we have 22 features employed by the model. Because we represent users using their previous answers, the features are constructed for each question-answer pair sequentially. Each training point consists of a question $q$, and a user $e$ that responded to that task at any point in time and has answered at least 1 previous question. Our model incorporates user reputation, number of total views of user's profile, number of upvotes the user has issued, number of downvotes the user has issued, the average time taken to answer questions prior to the current one, mean reciprocal rank for answers posted on previous questions, average view count for questions previously answered, average absolute score for previous answers (upvotes minus downvotes), the number of accepted answers/(total number of answers + $c$), the average usefulness, relevance, and informativeness of past answers as annotated by the Vicuna-7B large language model [30, 32], and the $0, 5, 10, 25$, and $50$ percentile for $\frac{\#\text{ Upvotes}}{\#\text{ Votes}}$ on all previous answers. We refer to the latter five features as the *past* $p(\text{Upvote})$ *distribution* as a shorthand, though for each answer $\frac{\#\text{ Upvotes}}{\#\text{ Votes}}$ is only an estimate of the true conditional distribution of $p(\text{Upvote}|\text{Vote}, (e,q))$. We run the Vicuna language model feature annotation for up to 10 days on each dataset on an RTX8000 or A100 GPU on a local computing cluster, and we run the cosine embedding similarity using SentenceTransformer on the same hardware for up to 3 days per dataset. More details about the annotation process for usefulness, relevance, and informativeness are included in Section 7.1.

We also include pairwise features of the question and the user, to identify content-based similarity. We collect the set of all of the user's previous answers. We represent each user as a weighted bag of keyword tags on the questions they have previously answered, and we represent the current question as a bag of tags. We can then compute the keyword similarity score as the product of the number of matching tags times the total count of the matching tags (following [29]). We also embed all question titles, question bodies, and answer bodies using the SentenceTransformer

[2]The data is available at `https://archive.org/details/stackexchange` under a cc-by-sa 4.0 license (`https://creativecommons.org/licenses/by-sa/4.0/`). We accessed this page on December 18, 2023.

[3]`https://stackexchange.com/tour`

Table 1: Predictive performance of each model on the test set. All models except the Similarity and Reputation baseline attempt to predict $p(\text{Upvote}|\text{Vote}, (e, q))$. P@100 and AUROC are computed assuming that a positive example is one with no downvotes, and a negative example is one with at least one downvote. The Kendall's $\tau$ and Spearman's $\rho$ statistics are calculated between the ranking produced by each model and the true ranking over all test examples by $p(\text{Upvote}|\text{Vote}, (e, q))$. All $\tau$ and $\rho$ statistics are statistically significant with a p-value of less than .001.

| **Model** | XE (↓) | $\tau$ (↑) | $\rho$ (↑) | P@100 (↑) | AUROC (↑) |
|---|---|---|---|---|---|
| Constant | .0890 | – | – | .94 | .5000 |
| Constant (per User) | .1028 | .0850 | .0866 | .94 | .5308 |
| Similarity and Reputation | – | .0828 | .1029 | **.98** | .6221 |
| Logistic Regression (Badges) | .0922 | .1001 | .1245 | .95 | .6461 |
| User Embeddings | .1054 | .1331 | .1653 | **.98** | .6964 |
| Logistic Regression (All) | **.0802** | **.1472** | **.1824** | .97 | **.7155** |

`multi-qa-mpnet-base-cos-v1` model [18].[4] We then include the mean and maximum cosine similarity between the current question title and the titles of questions the user previously answered, as well as the mean and maximum cosine similarity between the current question's body and the bodies of the user's previous answers.

We train a logistic regression model[5] on the training set to minimize the cross-entropy loss against the true distribution over labels for each user-question pair. For each pair $(e, q)$, we compute the empirical conditional distribution $p(\text{Upvote}|\text{Vote}, (e, q)) \doteq \frac{\text{\# Upvotes}}{\text{\# Votes}}$ and $p(\text{Downvote}|\text{Vote}, (e, q)) \doteq \frac{\text{\# Downvotes}}{\text{\# Votes}}$, and use this distribution over labels as the target distribution.

We train two baseline models that use less information about the users' history compared to the full logistic regression model. In some domains, like reviewer assignment, the user's history may be considered private. Ideally, we could obtain strong predictive performance even without storing this private information. The *Logistic Regression (Badges)* model uses a count vector of the badges awarded to the user instead of all features except for the user-answer textual cosine similarity measures, and the tag-based similarity measure. The *User Embeddings* model uses randomly-initialized user embeddings in place of all features except for the user-answer textual similarity measures, and the tag-based similarity measure. This model is trained using a 2-layer feed-forward neural network with sigmoid activations. It is trained over 400 epochs using the Adam optimizer [9]. All data processing and model training and evaluation in this section were performed on a Dell XPS laptop with a Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz processor and 8GB RAM.

### 4.1 Label Classification Performance

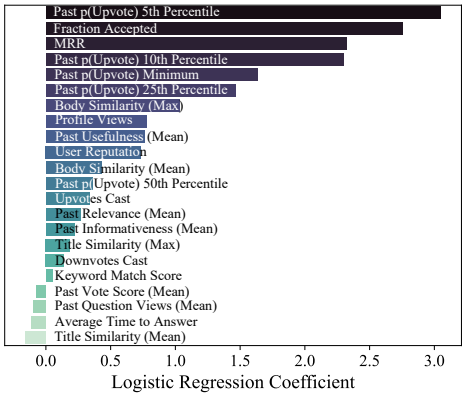
After training the logistic regression model, we evaluate the cross-entropy loss on the test set. We also compare against two baseline approaches. The *Constant* model predicts the average empirical $p(\text{Upvote}|\text{Vote}, (e, q))$ over the training set for each sample in the test set. The *Constant (per User)* model predicts the user's empirical average $p(\text{Upvote}|\text{Vote}, (e, q))$ up to the point in time of the question. Note that order statistics of the user's empirical conditional probability of receiving an upvote are also included as features in our logistic regression model. We report the cross-entropy loss on the test set in Table 1. Our trained model outperforms all the baselines, but the predictive models that use only badges or user embeddings are close to the performance of the full predictive model.

### 4.2 Feature Importance

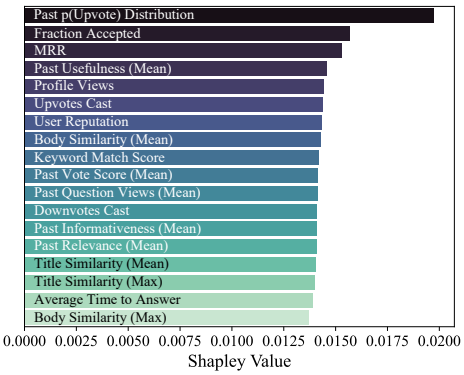
We investigate the coefficients of the logistic regression model in Figure 1a. Figure 1b shows the feature importance for each feature in our model using the exact computation of the Shapley value [23]. The Shapley value of a feature $x$ measures the average marginal decrease in test-set cross entropy loss when training the logistic regression model using a set of features $S \cup \{x\}$ compared

[4] `https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-cos-v1`.

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`




(a) Logistic regression model coefficients.




(b) Shapley values for features, measured as the average marginal contribution of the feature to decreasing cross entropy loss on the test set.

Figure 1: Logistic regression model coefficients and Shapley values on the test sets.

to using the set of features $S \setminus \{x\}$. We compute Shapley value on a computing cluster with Xeon E5-2680 v4 @ 2.40GHz with 128GB RAM, requesting up to 32GB for up to 6 days.

The top three features by both measures are measures of quality for the user's previous answers. Distributional measures of $p(\text{Upvote}|\text{Vote}, (e, q))$ over the user's past answers are incredibly important for predicting future $p(\text{Upvote}|\text{Vote}, (e, q))$. In addition, the ratio of accepted answers written by the user to total answers written by the user, as well as the mean reciprocal rank of answers written by the user are both very important.

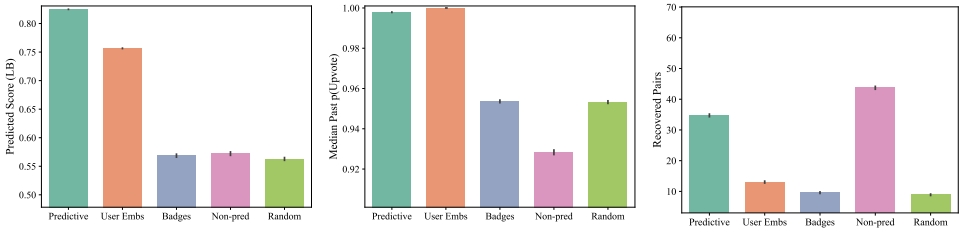
All features contribute meaningfully to reducing cross-entropy loss on the test set. This finding should encourage decision makers to collect and leverage as much information as is available in constructing predictive models of performance.

One likely explanation for the relative unimportance of topic-based features is the (extreme) sampling bias in StackExchange data. Our $(e, q)$ pairs consist only of answers that were actually submitted on the website. If a user $e$ does not have the expertise or interest to answer question $q$, he or she simply will not answer that question. Topical similarity is likely more useful for predicting *whether* a user $e$ would naturally answer question $q$. These measures are less useful in determining the probability that user $e$ provides a high quality answer to question $q$, conditioned on the fact that they chose to answer the question. As we will see in Section 5, this sampling bias does not preclude our model from being very useful in making assignments.

## 5 Making and Evaluating Assignments using the Predictive Model

We also use the test set to evaluate the overall assignment quality when assigning for predicted answer quality. We assume that all questions in the test set are received simultaneously, immediately following the end of the time period of the training set. Thus, each user's past answers (used in computing user and pairwise user-task features) are limited to only the answers to the first $80\%$ of questions by time of posting. This difference from the training set allows us to simulate testing on multiple questions without interdependence between questions, and shows the trade-offs required when assigning users with limited resources to many cold questions.

Overall, to evaluate assignments we include $n = 1{,}402$ questions (our set of tasks $Q$) and the $m = 220$ users who answered any of those questions and have answered at least one question before (our set of experts $E$). For robustness, we take $1{,}000$ samples of $60\%$ of the questions and $60\%$ of the users, and report the distributions of all evaluation metrics across all $1{,}000$ runs. For each assignment, we evaluate 12 metrics: $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$ under our predictive model, worst-case $p(\text{Upvote}|\text{Vote}, (e, q))$ according to Theorem 3.1 with $\delta = .1$, $5^{th}$ percentile and median of user's historical $p(\text{Upvote}|\text{Vote}, (e, q))$, number of user-question pairs that are observed in reality, average cosine similarity between user's past answers and question body, average keyword matching




(a) Performance on the lower bound of predicted score, with probability at least $90\%$.

(b) Performance measured by the median $p(\text{Upvote}|\text{Vote})$ on user's previous answers.

(c) Performance on number of recovered user-question pairs.

Figure 2: Performance of predictive assignment, non-predictive assignment (with .5 weight on user reputation and .5 weight on embedding similarity), and random assignments. The predictive model outperforms the others on a wide range of metrics; see Section 10 for more details.

score, average assigned reputation score for assigned users, true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs, and the average usefulness, relevance, and informativeness for assigned users' past answers. These metrics capture a wide range of automated measures for assignment quality, including topical similarity between the users and questions, and the users' propensities to leave satisfying answers.

We compare the assignment made using our predictive model to baseline assignments made without predictions. While the predictive model makes assignments according to the distribution $\hat{p}(l|(e, q))$ for all $(e, q) \in (E \times Q)$, our baselines must construct alternate scores for all $(e, q) \in (E \times Q)$. We construct the baseline to mirror standard expert assignment approaches, where a linear combination of topical match scores and user characteristics are used for assignment (as in reviewer assignment, which typically employs keyword match scores, bids, and document-based similarity scores [11]). We set the value for each $(e, q)$ pair to be $\lambda \frac{x_1(e)}{\max_{e \in E} x_1(e)} + (1 - \lambda) \frac{x_2(e,q)}{\max_{(e,q) \in (E \times Q)} x_2(e,q)}$, where $x_1(e)$ is the reputation of $e$ and $x_2(e, q)$ is the cosine similarity score between the answer bodies for $e$ and $q$. We set $\lambda = .5$ based on initial experiments varying $\lambda$ over $[0, 1]$ in increments of $0.1$. In addition, we also compute $100$ random values for each $(e, q)$ pair drawn from the uniform distribution $\mathcal{U}_{[0,1]}$. We set the number of experts per task to $v_q = 2$, and the number of tasks per expert to $u_e = 26$ (twice the smallest integer such that $mu_e \geq nv_q$).

To compute the worst-case and best-case bounds for the predictive assignment model (according to Theorem 3.1), we set $\delta = .1$. We estimate the cross entropy loss on the $1{,}666$ labeled question-answer pairs in the test set. We estimate the distribution $\mathcal{D}_{\text{TEST}}$ over the test set by fitting a 2-component principal component analysis on the test data and then applying kernel density estimation on the transformed features. We also estimate the distribution $\mathcal{D}_A$ over the pairs in $T_A$ for each assignment $A$ using the same procedure.

Figure 2 shows a the distributions of three metrics over all $1{,}000$ repeated experiments for the predictive assignment, the baseline with $\lambda = .5$, and the $100$ random assignments (there are $100{,}000$ runs reported in each plot for the random assignments). We report the lower bound computed using Theorem 3.1, the median of the assigned user's historical $p(\text{Upvote}|\text{Vote}, (e, q))$, as well as the number of recovered pairs which is the number of assigned $(e, q)$ pairs where $e$ actually answered $q$ on the website. Nine additional metrics are reported in Section 10. In Section 10 we also report experimental results for the biology, chemistry, and academia StackExchange datasets.

Overall, we find that the assignments made using the predicted answer scores strongly and robustly outperform most baselines on expected and worst-case predictive score, order statistics of user's past probability of upvote, user's average usefulness, informativeness, and relevance over past answers. However, the baseline using trained user embeddings performs very well on the objectives that measure user behavior without considering content similarity. The predictive and the baseline models both achieve nearly $1.00$ probability of upvote on recovered pairs (as can be seen from Figure 2, there are few of these relative to the dataset size anyway). Since most answers receive very few downvotes in this StackExchange, it would be very instructive to perform future experiments on datasets with higher baseline failure probabilities. The baseline outperforms the predictive model on text embedding cosine similarity, but the predictive model still generally has a decently-high average cosine similarity.

### 5.1 Metric Correlation Analysis

To further understand the trade-offs between evaluation metrics, we rank the assignment approaches using each metric for each of our $1{,}000$ experiment repetitions. We then study the correlation between rankings produced by each metric. Correlations and p-values for all four StackExchanges are displayed in Section 10.1, along with a more detailed explanation of the procedure for calculating these correlations.

The rankings produced under $\hat{p}$ are generally highly correlated with the rankings produced by order statistics of the user's past $p(\text{Upvote})$, user reputation, and the average LLM-annotated usefulness, relevance, and informativeness. This further bolsters our finding in the previous section that these historical features are highly informative for predicting the future probability of an upvote. The metric producing rankings most correlated with the rankings produced by the Recovered Pairs metric is the Keyword Match Score metric, suggesting again that textual similarity is predictive of whether a user would naturally answer a question (but this does not mean that this feature is predictive of their response quality given that they chose to answer this question). Finally, we see that the similarity score produces rankings that are generally negatively correlated wth rankings produced by measures of user competence (reputation with correlation $-.7$, and order statistics of past probability of upvote with correlation $-.7$ and $-1.0$). This dynamic suggests that perhaps assignments yielding content similarity may sacrifice user competence. StackExchange uses tags to recommend questions to users, and conference management platforms like Microsoft CMT compute keyword-based similarity measures, so this dynamic is incredibly worthy of further study.

## 6 Discussion

An ideal evaluation setting would make multiple assignments, have the assigned people write a response, and see how many upvotes they receive. It is not currently possible to run this experiment through StackExchange, and such an experiment would be very expensive in reviewer assignment. However, our $11$ automated metrics demonstrate that the predictive model incorporates multiple elements of user suitability in a satisfying way.

Our approach requires access to historic behavior for every user. While this is routine for StackExchange, this information is harder to access in privacy-sensitive areas like peer review. Reviewers may not be comfortable with sharing review performance history with systems like CMT or OpenReview without additional assurances, and this persistent state would require additional consent. However, many conferences have persistent organizing teams that retain access to review data from prior years, and with proper consent these conferences could simply make use of this already accessible information. Multiple research studies have expressed concerns that high-quality data about peer review is incredibly difficult to obtain [5, 15]. We hope that the current work encourages further study of the connections between input and output measures in other areas of expert assignment, and that conferences in particular will be encouraged to cooperate further with researchers to understand how decisions made during reviewer assignment impact downstream metrics of review quality.

## 7 Conclusion

We find defining metrics of interest and then optimizing for the predicted values of those metrics is more effective than optimizing for variables that are available prior to assignment. We give probabilistic bounds on the quality of such assignments, ensuring that the metric is optimized with high probability. Historic measures of answer quality prove to be the most important predictors of future answer quality; this finding can have important implications for expert recommendation in StackExchange as well as reviewer assignment for peer review. Although different measures of assignment quality can result in different results, the predictive model-based assignments outperform baseline approaches across a wide range of metrics. The correlations between the rankings produced by different metrics reveal important connections between these competing or complementary objectives; notably, we find some indication that assigning for content similarity may conflict with assigning for user competency. These trade-offs can be incorporated directly into a predictive model, such as the one proposed in the current study.

## Acknowledgments and Disclosure of Funding

Justin Payan and Yair Zick are supported by NSF grant CISE:IIS:RI: 2327057. This work was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Funded in part under IBM Research AI through the AI Horizons Network, Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, National Science Foundation (NSF) grant numbers IIS-1922090 and SES-2244805, and in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

## References

[1] Julia Cambre, Scott Klemmer, and Chinmay Kulkarni. Juxtapeer: Comparative peer review yields higher quality feedback and promotes deeper reflection. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[2] Laurent Charlin and Richard Zemel. The Toronto paper matching system: an automated paper-reviewer assignment system. In *Proceedings of the 2013 ICML Workshop on Peer Reviewing and Publishing Models*, 2013.

[3] Laurent Charlin, Richard Zemel, and Craig Boutilier. A framework for optimizing paper matching. In *Proceedings of the 27th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 86–95, 2011.

[4] Laurent Charlin, Richard Zemel, and Craig Boutilier. Active learning for matching problems. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 139–146, 2012.

[5] Don Conry, Yehuda Koren, and Naren Ramakrishnan. Recommender systems for the conference paper assignment problem. In *Proceedings of the 3rd ACM Conference on Recommendation Systems (RecSys)*, pages 357–360, 2009.

[6] Cyrus Cousins, Justin Payan, and Yair Zick. Into the unknown: Assigning reviewers to papers with uncertain affinities. In *Algorithmic Game Theory: 16th International Symposium, SAGT 2023, Egham, UK, September 4–7, 2023, Proceedings*, page 179–197, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-43253-8. doi: 10.1007/978-3-031-43254-5_11. URL `https://doi.org/10.1007/978-3-031-43254-5_11`.

[7] Hengyi Fu and Sanghee Oh. Quality assessment of answers with user-identified criteria and data-driven features in social q&a. *Information Processing & Management*, 56(1):14–28, 2019. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2018.08.007. URL `https://www.sciencedirect.com/science/article/pii/S0306457318302498`.

[8] Vijay Kamble, Abhay Parekh, and Kannan Ramachandran. Truth serums for massively crowd-sourced evaluation tasks. *arXiv preprint arXiv:1507.07045*, 96, 2015.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Ari Kobren, Barna Saha, and Andrew McCallum. Paper matching with local fairness constraints. In *Proceedings of the 25th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1247–1257, 2019.

[11] Kevin Leyton-Brown, Yatin Nandwani, Hedayat Zarkoob, Chris Cameron, Neil Newman, Dinesh Raghu, et al. Matching papers and reviewers at large conferences. *arXiv preprint arXiv:2202.12273*, 2022.

[12] Mingrong Liu, Yicen Liu, and Qing Yang. Predicting best answerers for new questions in community question answering. In *Web-Age Information Management: 11th International Conference, WAIM 2010, Jiuzhaigou, China, July 15-17, 2010. Proceedings 11*, pages 127–138. Springer, 2010.

[13] Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1): 148–188, 1989.

[14] Sarah EM Meek, Louise Blakemore, and Leah Marks. Is peer review an appropriate form of assessment in a mooc? student participation and performance in formative peer review. *Assessment & evaluation in higher education*, 42(6):1000–1013, 2017.

[15] David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 500–509, 2007.

[16] Justin Payan and Yair Zick. I will have order! Optimizing orders for fair reviewer assignment. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 440–446, 2022.

[17] Yujie Qian, Jie Tang, and Kan Wu. Weakly learning to match experts in online community. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3841–3847, 2018.

[18] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

[19] Philippe Rigaux. An iterative rating method: Application to web-based conference management. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1682–1687, 2004.

[20] Martin Saveski, Steven Jecmen, Nihar Shah, and Johan Ugander. Counterfactual evaluation of peer-review assignment policies. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] Chirag Shah and Jefferey Pomerantz. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 411–418, 2010.

[22] Nihar B Shah, Joseph K Bradley, Abhay Parekh, Martin Wainwright, and Kannan Ramchandran. A case for ordinal peer-evaluation in moocs. In *NIPS Workshop on Data Driven Education*, 2013.

[23] Lloyd S Shapley et al. A value for n-person games. *Contributions to the Theory of Games*, 1953.

[24] C Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101, 1904.

[25] Ivan Stelmakh, Nihar B Shah, and Aarti Singh. PeerReview4All: Fair and accurate reviewer assignment in peer review. In *Proceedings of the 30th International Conference on Algorithmic Learning Theory (ALT)*, pages 828–856, 2019.

[26] Jiankai Sun, Abhinav Vishnu, Aniket Chakrabarti, Charles Siegel, and Srinivasan Parthasarathy. Coldroute: effective routing of cold questions in stack exchange sites. *Data mining and knowledge discovery*, 32:1339–1367, 2018.

[27] Jiankai Sun, Jie Zhao, Huan Sun, and Srinivasan Parthasarathy. Endcold: An end-to-end framework for cold question routing in community question answering services. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 3244–3250, 2021.

[28] Camillo J Taylor. On the optimal assignment of conference papers to reviewers. Technical report, University of Pennsylvania, 2008.

[29] Rohan Tondulkar, Manisha Dubey, and Maunendra Sankar Desarkar. Get me the best: predicting best answerers in community question answering sites. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 251–259, 2018.

[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[31] Liu Yang, Minghui Qiu, Swapna Gottipati, Feida Zhu, Jing Jiang, Huiping Sun, and Zhong Chen. Cqarank: jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 99–108, 2013.

[32] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] ZM Zhu, Delphine Bernhard, and Iryna Gurevych. A multi-dimensional model for assessing the quality of answers in social q&a, 2009.

# Appendix

### 7.1 Annotation of Usefulness, Relevance, and Informativeness

We also score all question-answer pairs for usefulness, informativeness, and relevance using Vicuna, released August 2023. These three criteria were found to be the most highly correlating criteria with overall answer quality in a survey of experts on community question answering [33]. Using these annotations, for each question-answer pair we compute the mean usefulness, relevance, and informativeness scores on the user's past answers. These scores are used as input features for our predictive model, but we also apply them as part of our automated assignment evaluation.

For all question-answer pairs, we set the system prompt for the Vicuna model as:

```
I am going to provide you with a question-answer pair from the cs
    StackExchange. Please annotate the informativeness, relevance,
     and usefulness of the answer. Your response should rate each
    of these three aspects on a scale from 1-5, with 1 being the
    least and 5 being the most. Please structure your response by
    outputting the informativeness, then the relevance, and then
    the usefulness, one per line. Please add an additional
    explanation of your ratings. Informativeness asks Does this
    answer provide enough information for the question? Relevance
    asks Is this answer relevant to the question? Usefulness asks
    Is this answer useful or helpful to address the question?Use
    this template for your output: \nInformativeness: <Rating>\
    nRelevance: <Rating>\nUsefulness: <Rating>\nExplanation: <
    Additional Explanation>\n\n
```

We then prompt the model as the user:

```
Question:\nTitle: <Question Title>\n\nBody: <Question Body>\n\
    nAnswer:\n<Answer Body>\n
```

Inserting the question title, question body, and answer body where appropriate.

## 8 Proof of Main Theorem

**Theorem 3.1.** *For any $\delta \in (0,1)$, the true probability distribution $p^*$ satisfies*

$$\frac{1}{v_q n}\langle A, \langle \Lambda, H(p^*, \hat{p})\rangle_F\rangle_F \leq \xi + \sqrt{\frac{\gamma^2 \ln \frac{1}{\delta}}{2}\left(\frac{1}{t} + \frac{1}{v_q^2 n^2}\sum_{(e,q)\in T_A} \Lambda_{e,q}^2\right)}$$

*with probability at least $1-\delta$.*

*Proof.* We bound

$$
\begin{aligned}
Z &\doteq \frac{1}{v_q n}\langle A, \langle \Lambda, H(p^*, \hat{p})\rangle_{\mathrm{F}}\rangle_{\mathrm{F}} - \xi \\
&= \frac{1}{v_q n} \sum_{(e',q')\in T_A} \Lambda_{e',q'} \mathbb{H}(p^*(l|(e',q')), \hat{p}(l|(e',q'))) - \frac{1}{t}\sum_{(e,q)\in T} \mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))\ .
\end{aligned}
$$

First, we show that $\mathbb{E}[Z] = 0$. Recall that for all $(e',q') \in T_A$, $(e',q') \sim \mathcal{D}_A$, and that for all $(e,q) \in T$, $(e,q) \sim \mathcal{D}_{\text{TEST}}$. Thus,

$$
\begin{aligned}
\mathop{\mathbb{E}}_{(e,q)\sim\mathcal{D}_{\text{TEST}},(e',q')\sim\mathcal{D}_A}[Z] = &\mathop{\mathbb{E}}_{(e',q')\sim\mathcal{D}_A}\left[\frac{1}{v_q n}\sum_{(e',q')\in T_A} \Lambda_{e',q'}\mathbb{H}(p^*(l|(e',q')), \hat{p}(l|(e',q')))\right] \\
&- \mathop{\mathbb{E}}_{(e,q)\sim\mathcal{D}_{\text{TEST}}}\left[\frac{1}{t}\sum_{(e,q)\in T}\mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))\right]\ .
\end{aligned}
$$

Let

$$
X \doteq \mathop{\mathbb{E}}_{(e',q')\sim\mathcal{D}_A}\left[\frac{1}{v_q n}\sum_{(e',q')\in T_A} \Lambda_{e',q'}\mathbb{H}(p^*(l|(e',q')), \hat{p}(l|(e',q')))\right]
$$

We have that,

$$
\begin{aligned}
X &= \frac{1}{v_q n}\sum_{(e',q')\in T_A}\mathop{\mathbb{E}}_{(e',q')\sim\mathcal{D}_A}[\Lambda_{e',q'}\mathbb{H}(p^*(l|(e',q')), \hat{p}(l|(e',q')))] \\
&= \frac{1}{v_q n}\sum_{(e',q')\in T_A}\mathop{\mathbb{E}}_{(e,q)\sim\mathcal{D}_{\text{TEST}}}[\mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))] \\
&= \frac{1}{t}\sum_{(e,q)\in T}\mathop{\mathbb{E}}_{(e,q)\sim\mathcal{D}_{\text{TEST}}}[\mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))] \\
&= \mathop{\mathbb{E}}_{(e,q)\sim\mathcal{D}_{\text{TEST}}}\left[\frac{1}{t}\sum_{(e,q)\in T}\mathbb{H}(p^*(l|(e,q)), \hat{p}(l|(e,q)))\right]
\end{aligned}
$$

Now that we have shown $\mathbb{E}[Z] = 0$, we apply McDiarmid's method of bounded differences to obtain a tail bound on $Z$ [13], as $Z$ is a function of negatively-dependent random variables. To use this method, we bound the impact of changing any of the random variables in $Z$. We assume that the cross-entropy loss cannot exceed $\gamma$ for a single sample, so $t$ terms in $Z$ contribute at most $\frac{\gamma}{t}$ each, and each of the first $v_q n$ terms contribute at most $\frac{\gamma\Lambda_{e,q}}{v_q n}$ for each $(e,q) \in T_A$. The sum of the squared bounded differences is

$$
\gamma^2\left(\frac{1}{t} + \frac{1}{v_q^2 n^2}\sum_{(e,q)\in T_A}\Lambda_{e,q}^2\right)\ ,
$$

which implies finally that

$$
\Pr\left(Z \geq \sqrt{\frac{\gamma^2 \ln\frac{1}{d}}{2}\left(\frac{1}{t} + \frac{1}{v_q^2 n^2}\sum_{(e,q)\in T_A}\Lambda_{e,q}^2\right)}\right) \leq \delta\ .
$$

□

## 9 Details of Dataset

For each question, we compute features representing each user's topical affinity for the question and overall answering proficiency. Using these features, we predict labels of answer quality. We

Table 2: Predictive performance of each model on the test set for biology.

| **Model** | XE (↓) | $\tau$ (↑) | $\rho$ (↑) | P@100 (↑) | AUROC (↑) |
|---|---|---|---|---|---|
| Constant | .1470 | – | – | .88 | .5000 |
| Constant (per User) | .1688 | .0381 | .0399 | .87 | .5150 |
| Similarity and Reputation | – | .0759 | .0962 | .85 | .5771 |
| Logistic Regression (Badges) | .1531 | .0987 | .1253 | **.9** | .6036 |
| User Embeddings | .1613 | .1046 | .1323 | **.9** | .6070 |
| Logistic Regression (All) | **.1363** | **.1225** | **.1547** | **.9** | **.6221** |

Table 3: Predictive performance of each model on the test set for chemistry.

| **Model** | XE (↓) | $\tau$ (↑) | $\rho$ (↑) | P@100 (↑) | AUROC (↑) |
|---|---|---|---|---|---|
| Constant | .1408 | – | – | .86 | .5000 |
| Constant (per User) | .1582 | .1319 | .1364 | .88 | .5545 |
| Similarity and Reputation | – | .1246 | .1563 | **.97** | .6455 |
| Logistic Regression (Badges) | .1624 | .0906 | .1136 | .94 | .6025 |
| User Embeddings | .1381 | .1078 | .1351 | **.97** | .6250 |
| Logistic Regression (All) | **.1177** | **.1767** | **.2206** | .96 | **.7056** |

construct a train and test set for building our predictive model of answer quality. We then evaluate the model's performance for labeling the test set. We also construct an assignment problem using the set of test questions, and use our predicted distribution over labels to assign a set of users to these questions. This setting simulates assigning users from a pool to the set of cold questions. We evaluate the assignments using the theoretical bounds developed in this paper and by comparing multiple automated metrics of assignment.

We first filter the questions and answers for quality. We select all questions that have an accepted answer and have a score of at least 3, where the score is computed by adding 1 for each upvote and subtracting 1 for each downvote. We sort the questions by creation time. The first $10\%$ of questions are used to initialize user representations. The next $70\%$ of questions are used as the training set, and the remaining $20\%$ as the test set.

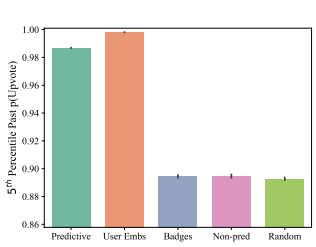
The CS StackExchange includes questions from November 25, 2008 to December 2, 2023. The training set starts at November 30, 2012 and contains 7045 question-answer pairs from 4831 unique questions. The test set starts at February 5, 2019 and contains 1666 question-answer pairs from 1182 unique questions.

## 10 Additional Metrics and all StackExchanges

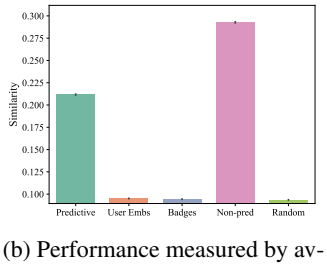
Results indicating the power of the statistical models are shown in Tables 2 to 4.

Table 4: Predictive performance of each model on the test set for academia.

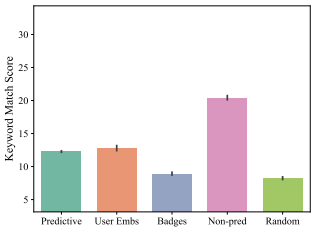
| **Model** | XE (↓) | $\tau$ (↑) | $\rho$ (↑) | P@100 (↑) | AUROC (↑) |
|---|---|---|---|---|---|
| Constant | **.2403** | – | – | .77 | .5000 |
| Constant (per User) | .2848 | .0400 | .0437 | .77 | .5089 |
| Similarity and Reputation | – | $-.0004$ | $-.0003$ | .69 | .4878 |
| Logistic Regression (Badges) | .3255 | .0480 | .0640 | .67 | .5292 |
| User Embeddings | .2860 | **.1199** | **.1546** | .8 | **.5969** |
| Logistic Regression (All) | **.2403** | .1060 | .1384 | **.83** | .5708 |



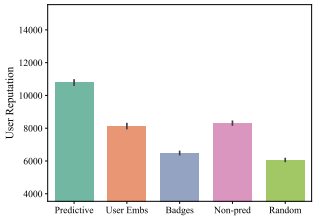

(a) Performance on the $5^{th}$ percentile of historical probability of user's past answers to the upvote given vote.



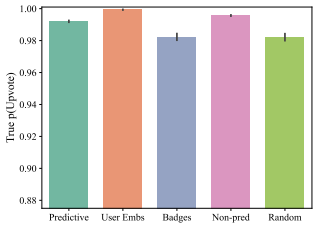

(b) Performance measured by average cosine similarity of the question body.



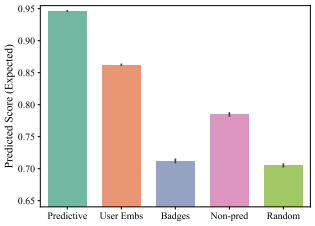

(c) Performance on the keyword matching score.



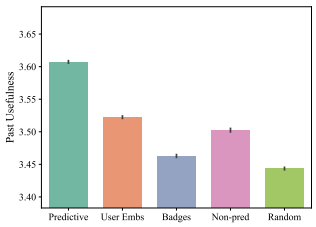

(d) Performance measured by user's reputation score.



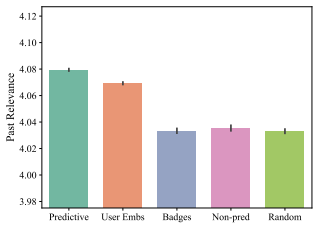

(e) Performance on true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs.



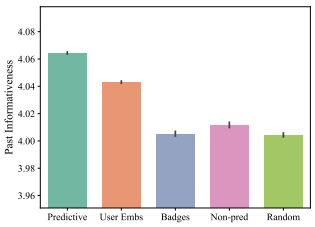

(f) Performance on $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$.




(g) Performance measured by user's average usefulness over past answers.




(h) Performance measured by user's average relevance over past answers.



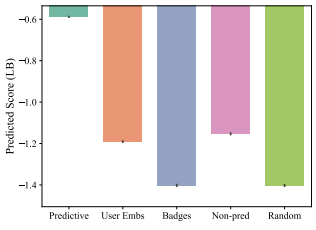

(i) Performance measured by user's average informativeness over past answers.

Figure 3: Additional metrics for computer science. Performance of predictive assignment, non-predictive assignment (with .5 weight on user reputation and .5 weight on embedding similarity), and random assignments on $5^{th}$ percentile of user's historical probability of upvote given vote, average cosine similarity between user's past answers and question body, keyword matching score, user's reputation score, true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs, $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$, and user's average usefulness, relevance, and informativeness as rated by Vicuna-7B on past answers.

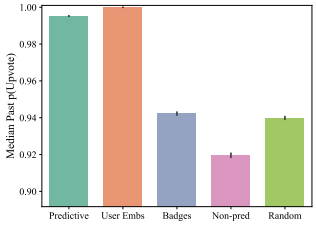
Figure 3 shows the performance distribution of the predictive assignment, non-predictive baseline with $\lambda = .5$, and random assignments, over $1{,}000$ subsamples of the Computer Science StackExchange test set. Although the non-predictive baseline does best on the embedding-based similarity measure, the predictive assignment is much better on most other metrics. Note that Figure 3e computes the true $p(\text{Upvote})$ based on only the user-question pairs that were present on the website, and is likely a very biased and high-variance measure of the true underlying probability of an upvote for the assigned pairs.

### 10.1 Correlation of Rankings

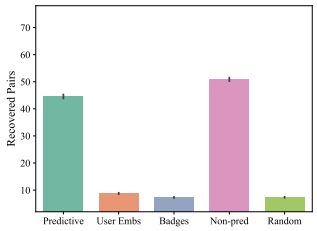
For each of the $1{,}000$ experimental runs, we compute the ranking of assignment methods under each of 11 metrics. The assignment methods are the predictive method, the baseline with $\lambda \in [0, 1]$ stepping by .1, and random. For metrics, we exclude the average empirical probability of an upvote for recovered pairs, as it is based on a small number of recovered pairs and equals $1.0$ for a majority of assignments. This results in $11{,}000$ rankings over 13 assignment methods. We then compute the



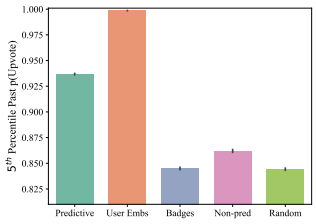

(a) Performance on the lower bound of predicted score, with probability at least $90\%$.



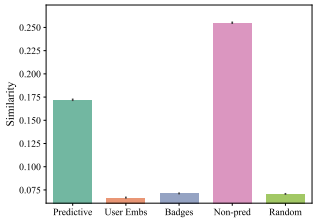

(b) Performance measured by the median $p(\text{Upvote}|\text{Vote})$ on user's previous answers.



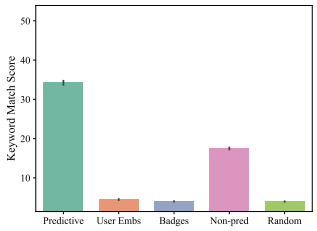

(c) Performance on the number of recovered user-question pairs.



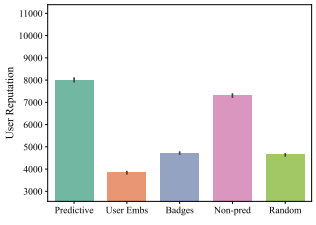

(d) Performance on the $5^{th}$ percentile of historical probability of user's past answers to the upvote given vote.



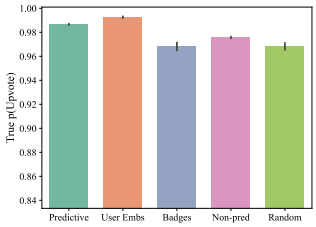

(e) Performance measured by average cosine similarity of the question body.



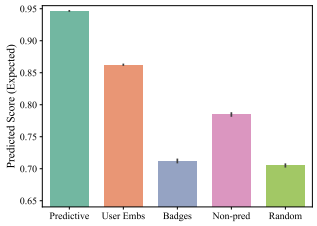

(f) Performance on the keyword matching score.



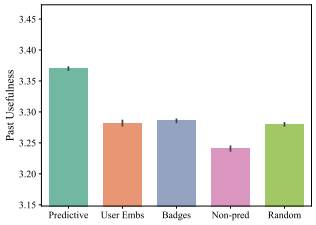

(g) Performance measured by user's reputation score.



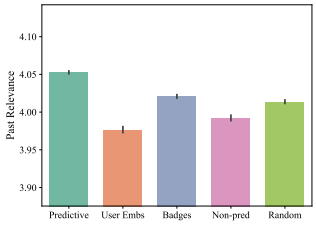

(h) Performance on true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs.



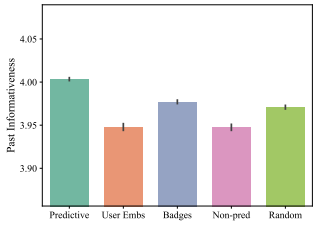

(i) Performance on $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$.




(j) Performance measured by user's average usefulness over past answers.

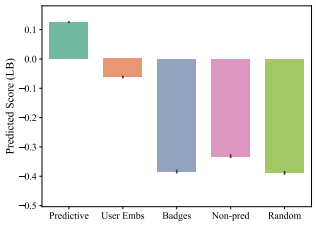



(k) Performance measured by user's average relevance over past answers.

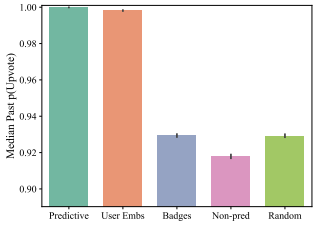



(l) Performance measured by user's average informativeness over past answers.

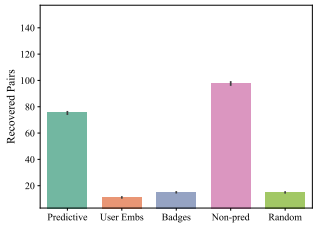
Figure 4: Metrics for biology. Performance of predictive assignment, non-predictive assignment (with .5 weight on user reputation and .5 weight on embedding similarity), and random assignments on predicted value (expected and worst-case at $90\%$ confidence), recovered user-query pairs, $5^{th}$ percentile of user's historical probability of upvote given vote, average cosine similarity between user's past answers and question body, keyword matching score, user's reputation score, true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs, and user's average usefulness, relevance, and informativeness as rated by Vicuna-7B on past answers.



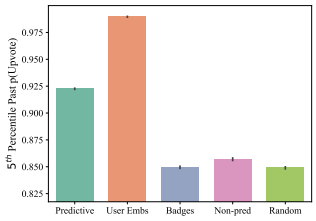

(a) Performance on the lower bound of predicted score, with probability at least $90\%$.



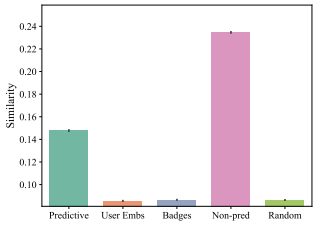

(b) Performance measured by the median $p(\text{Upvote}|\text{Vote})$ on user's previous answers.



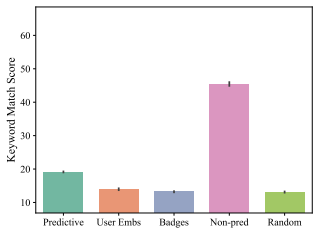

(c) Performance on the number of recovered user-question pairs.



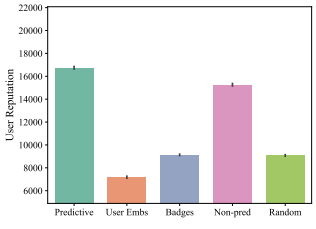

(d) Performance on the $5^{th}$ percentile of historical probability of upvote given vote.



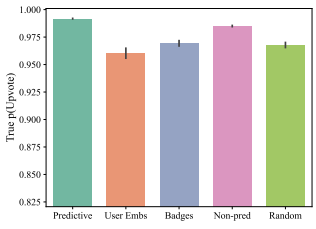

(e) Performance measured by average cosine similarity of the user's past answers to the question body.



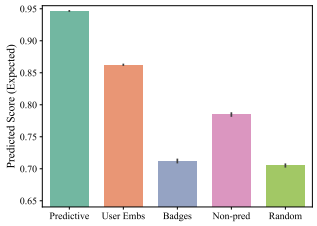

(f) Performance on the keyword matching score.



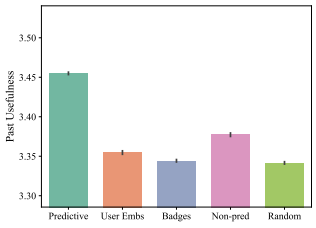

(g) Performance measured by user's reputation score.



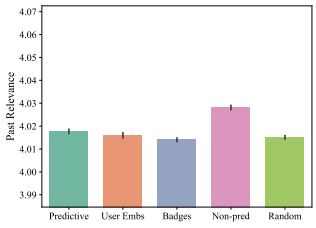

(h) Performance on true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs.



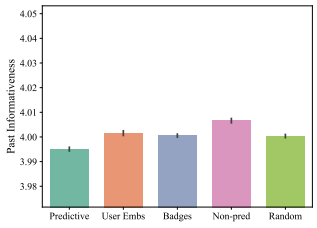

(i) Performance on $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$.




(j) Performance measured by user's average usefulness over past answers.

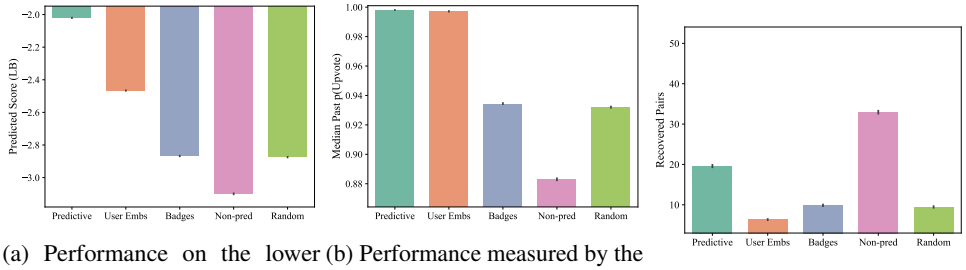



(k) Performance measured by user's average relevance over past answers.




(l) Performance measured by user's average informativeness over past answers.

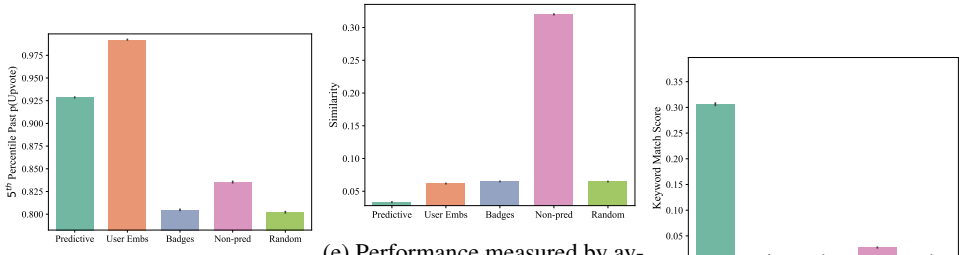
Figure 5: Additional metrics for chemistry. Performance of predictive assignment, non-predictive assignment (with .5 weight on user reputation and .5 weight on embedding similarity), and random assignments on predicted value (expected and worst-case at $90\%$ confidence), recovered user-query pairs, $5^{th}$ percentile of user's historical probability of upvote given vote, average cosine similarity between user's past answers and question body, keyword matching score, user's reputation score, true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs, and user's average usefulness, relevance, and informativeness as rated by Vicuna-7B on past answers.




(a) Performance on the lower bound of predicted score, with probability at least $90\%$.

(b) Performance measured by the median $p(\text{Upvote}|\text{Vote})$ on user's previous answers.

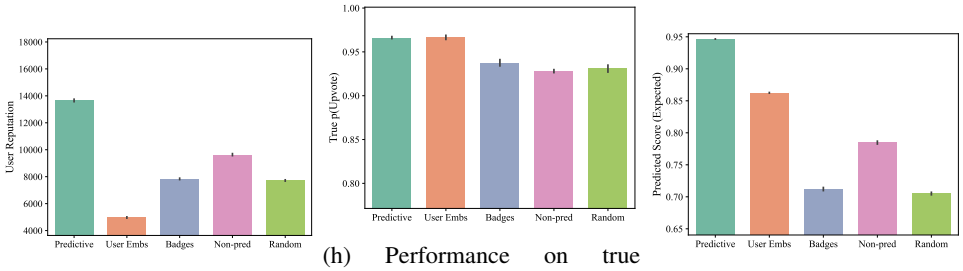
(c) Performance on the number of recovered user-question pairs.




(d) Performance on the $5^{th}$ percentile of historical probability of upvote given vote.

(e) Performance measured by average cosine similarity of the user's past answers to the question body.

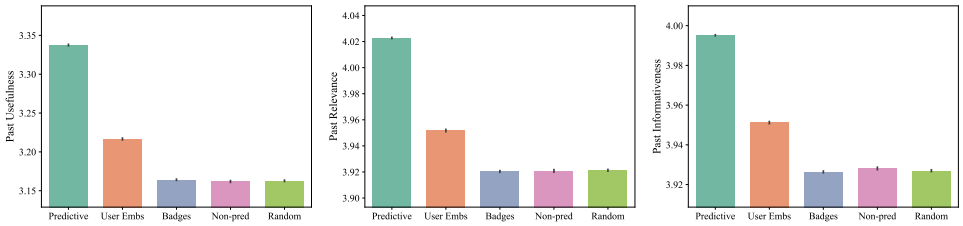
(f) Performance on the keyword matching score.




(g) Performance measured by user's reputation score.

(h) Performance on true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs.

(i) Performance on $\hat{p}(\text{Upvote}|\text{Vote}, (e, q))$.




(j) Performance measured by user's average usefulness over past answers.

(k) Performance measured by user's average relevance over past answers.

(l) Performance measured by user's average informativeness over past answers.

Figure 6: Additional metrics for academia. Performance of predictive assignment, non-predictive assignment (with .5 weight on user reputation and .5 weight on embedding similarity), and random assignments on predicted value (expected and worst-case at $90\%$ confidence), recovered user-query pairs, $5^{th}$ percentile of user's historical probability of upvote given vote, average cosine similarity between user's past answers and question body, keyword matching score, user's reputation score, true $p(\text{Upvote}|\text{Vote}, (e, q))$ for recovered pairs, and user's average usefulness, relevance, and informativeness as rated by Vicuna-7B on past answers.

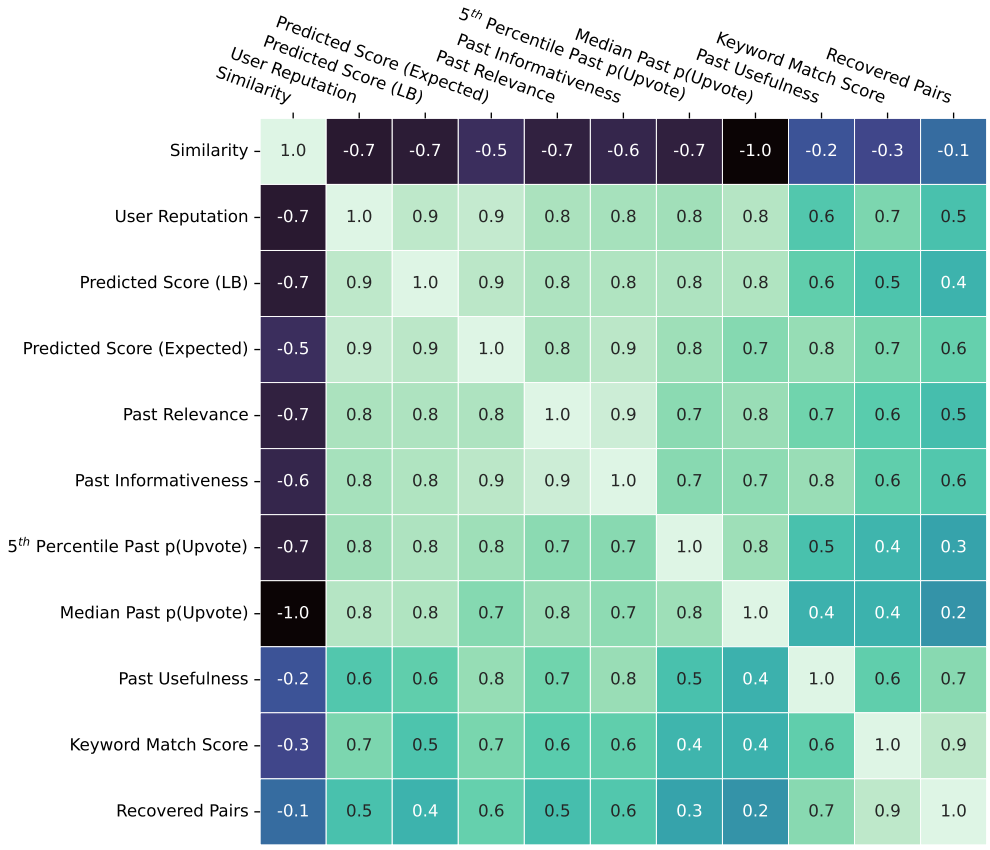



Figure 7: Correlation of ranking of assignments by each metric for computer science StackExchange. Higher values indicate a stronger positive correlation between rankings produced by the two metrics, while lower negative values indicate stronger negative correlation between rankings. Values close to 0 indicate less correlation overall.

Spearman's ranking coefficient [24] between each pair of rankings, giving us an understanding of both the robustness of each metric and the correlations in rankings across metrics.

Figure 7 displays the average correlations (over 1,000 samples) of the rankings over assignments produced by each metric. High positive values indicate the metrics tend to rank assignments in the same order, low negative values indicate the metrics tend to rank assignments in opposite order. Values near zero indicate low correlation in rankings in either direction. The p-value of these statistics is displayed in Figure 8. Similar results are shown for the other StackExchanges in Figures 9 to 14.

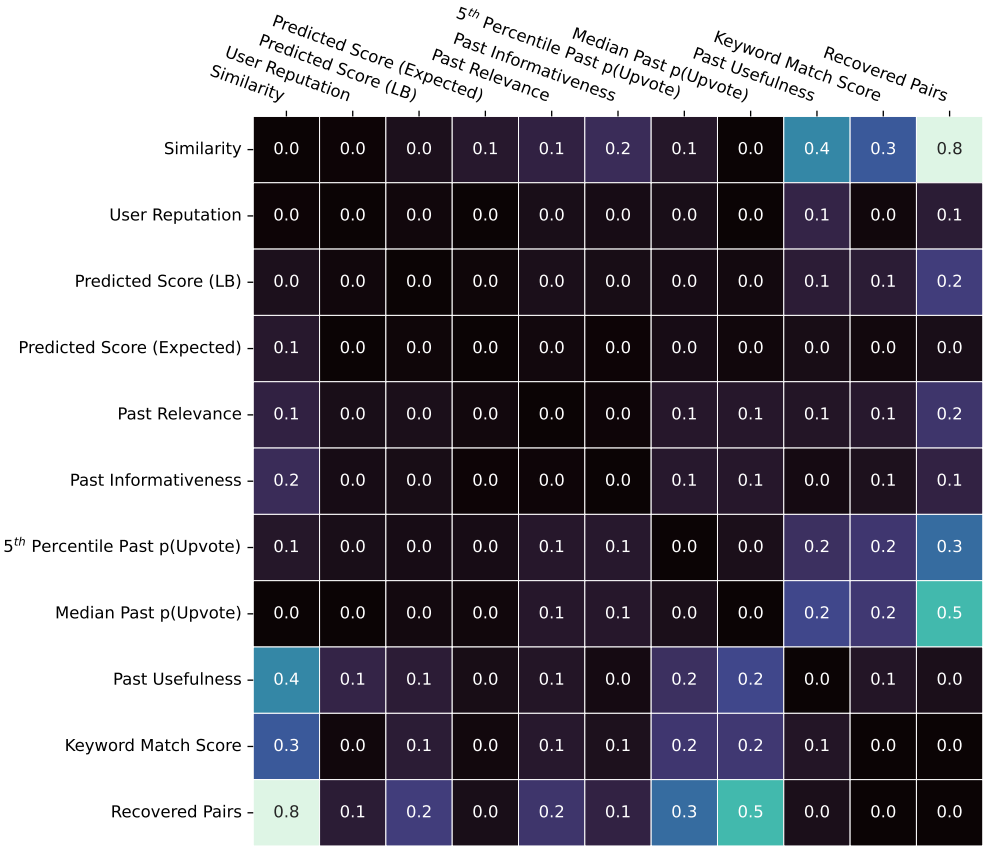



Figure 8: P-values for correlation of ranking of assignments by each metric for computer science StackExchange. Only the pairs with p-values close to 0 are statistically significant.

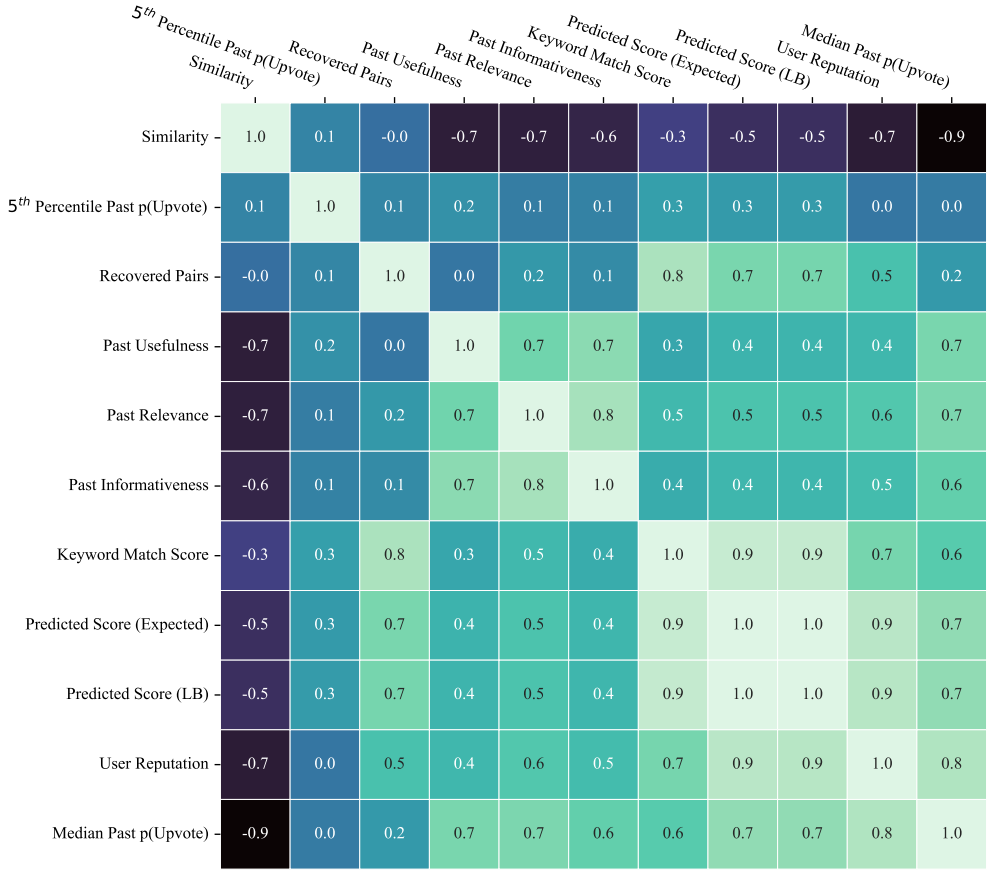



Figure 9: Correlation of ranking of assignments by each metric for biology. Higher values indicate a stronger positive correlation between rankings produced by the two metrics, while lower negative values indicate stronger negative correlation between rankings. Values close to 0 indicate less correlation overall.

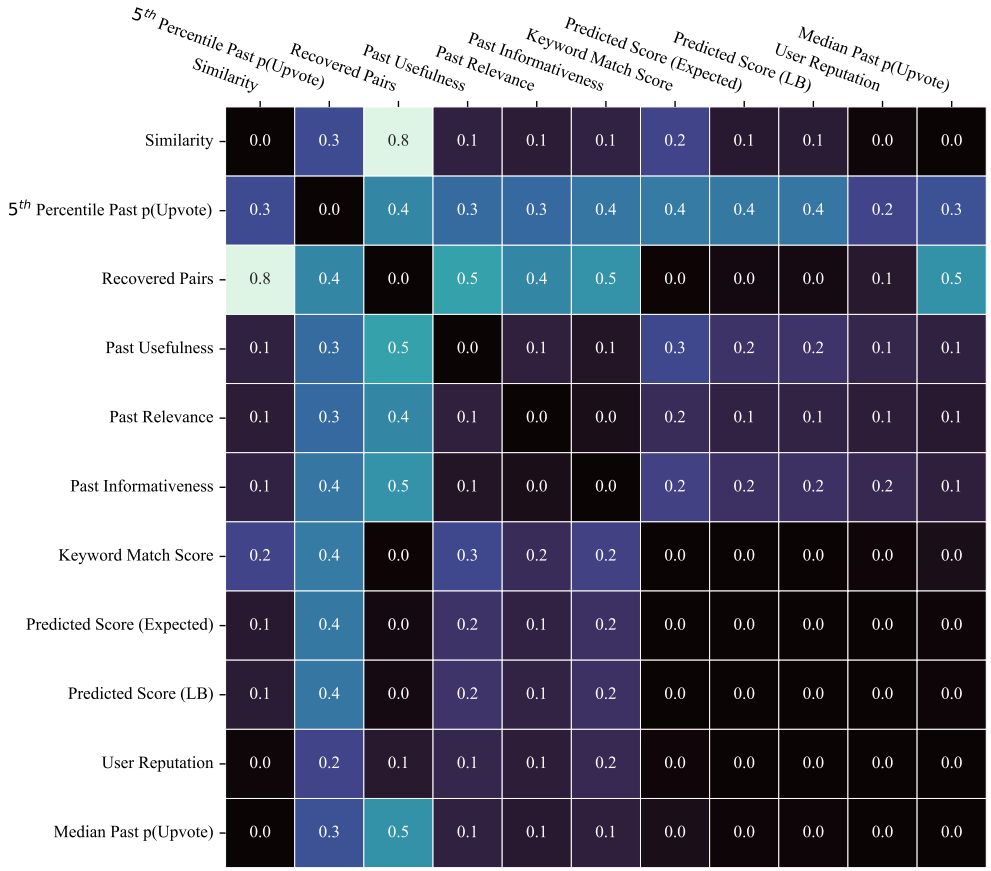



Figure 10: P-values for correlation of ranking of assignments by each metric for biology. Only the pairs with p-values close to 0 are statistically significant.

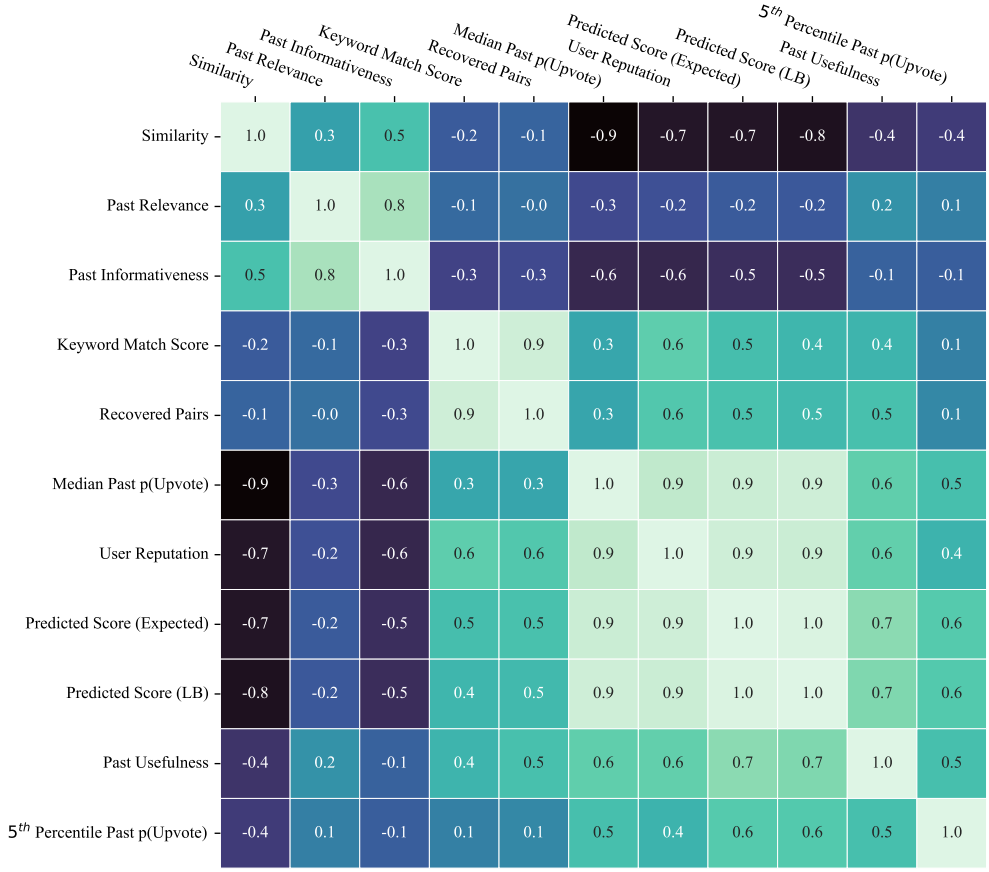



Figure 11: Correlation of ranking of assignments by each metric for chemistry. Higher values indicate a stronger positive correlation between rankings produced by the two metrics, while lower negative values indicate stronger negative correlation between rankings. Values close to 0 indicate less correlation overall.

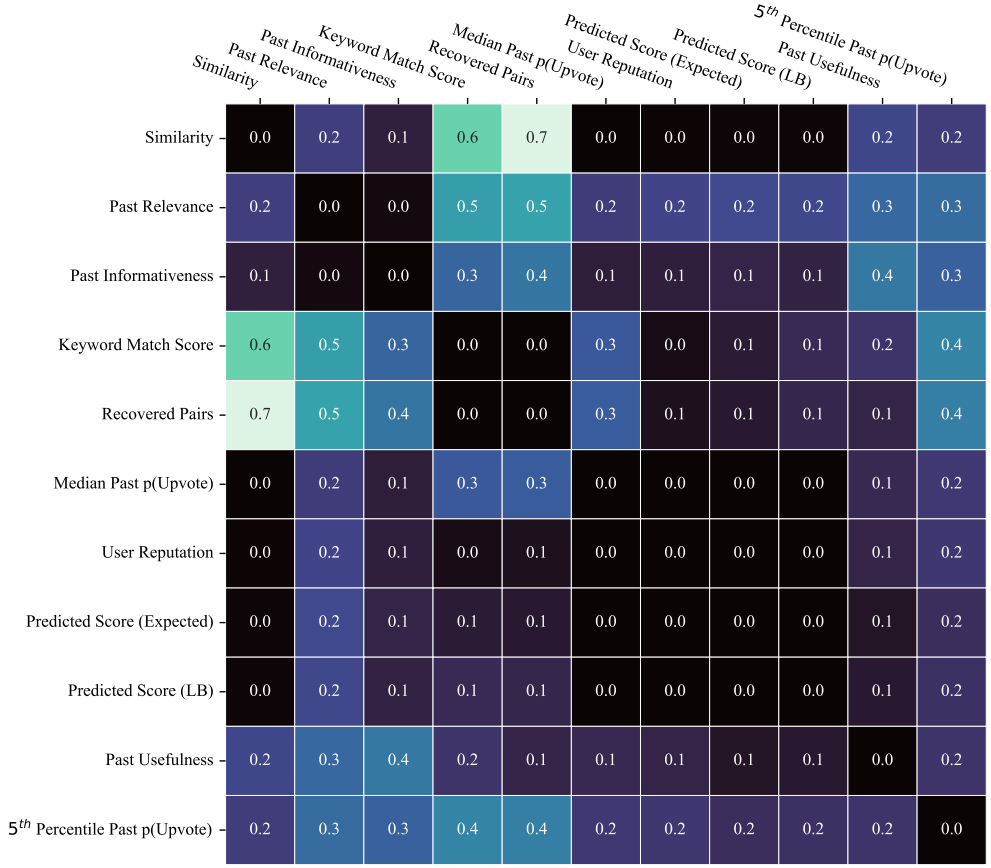



Figure 12: P-values for correlation of ranking of assignments by each metric for chemistry. Only the pairs with p-values close to 0 are statistically significant.

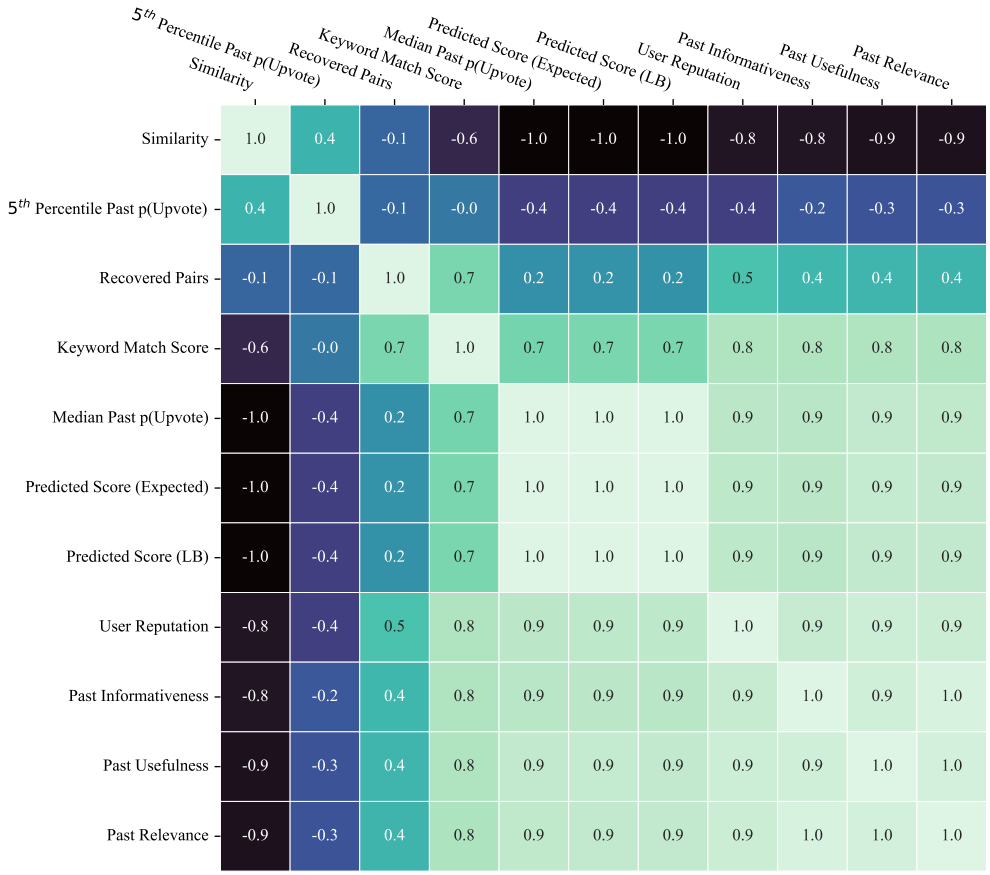



Figure 13: Correlation of ranking of assignments by each metric for academia. Higher values indicate a stronger positive correlation between rankings produced by the two metrics, while lower negative values indicate stronger negative correlation between rankings. Values close to 0 indicate less correlation overall.

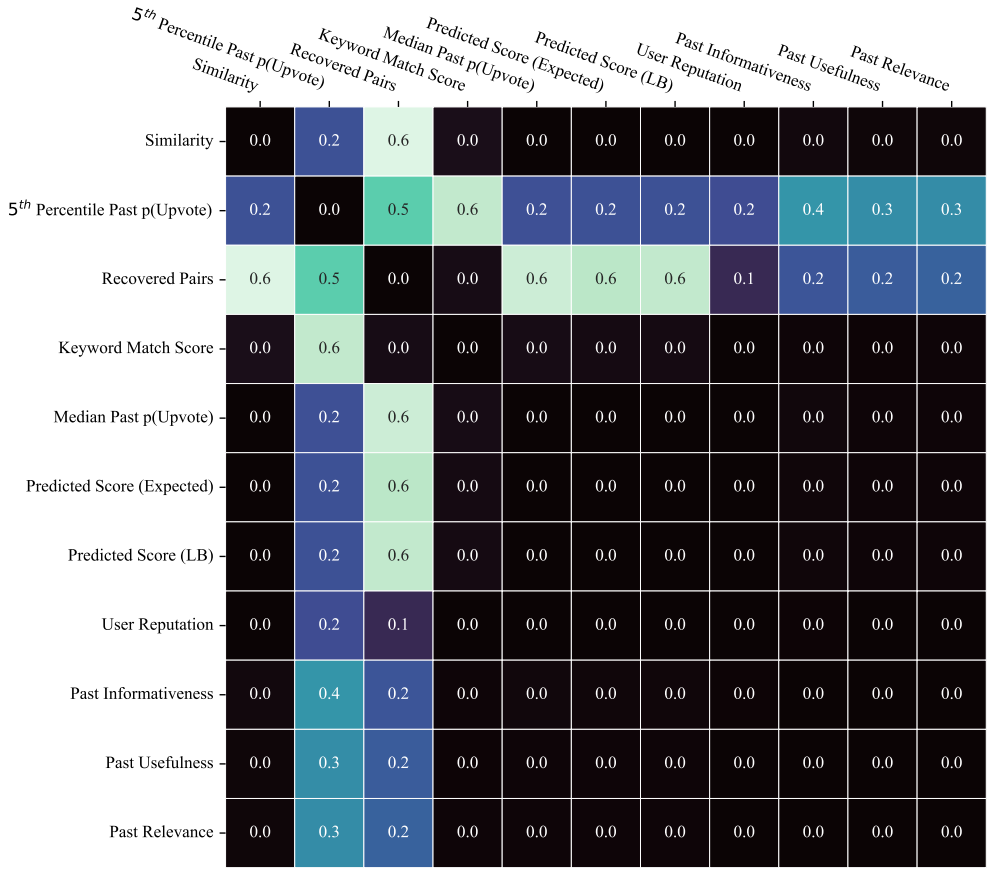



Figure 14: P-values for correlation of ranking of assignments by each metric for academia. Only the pairs with p-values close to 0 are statistically significant.