Can Who-Edits-What Predict Edit Survival?

Ali Batuhan Yardım*
Bilkent University
batuhan.yardim@ug.bilkent.edu.tr

Victor Kristof
Ecole Polytechnique Fédérale de Lausanne
victor.kristof@epfl.ch

Lucas Maystre
Ecole Polytechnique Fédérale de Lausanne
lucas.maystre@epfl.ch

Matthias Grossglauser
Ecole Polytechnique Fédérale de Lausanne
matthias.grossglauser@epfl.ch

ABSTRACT

As the number of contributors to online peer-production systems grows, it becomes increasingly important to predict whether the edits that users make will eventually be beneficial to the project. Existing solutions either rely on a user reputation system or consist of a highly specialized predictor that is tailored to a specific peer-production system. In this work, we explore a different point in the solution space that goes beyond user reputation but does not involve any content-based feature of the edits. We view each edit as a game between the editor and the component of the project. We posit that the probability that an edit is accepted is a function of the editor's skill, of the difficulty of editing the component and of a user-component interaction term. Our model is broadly applicable, as it only requires observing data about *who* makes an edit, *what* the edit affects and whether the edit survives or not. We apply our model on Wikipedia and the Linux kernel, two examples of large-scale peer-production systems, and we seek to understand whether it can effectively predict edit survival: in both cases, we provide a positive answer. Our approach significantly outperforms those based solely on user reputation and bridges the gap with specialized predictors that use content-based features. It is simple to implement, computationally inexpensive, and in addition it enables us to discover interesting structure in the data.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing**; *Reputation systems*; • **Information systems** → *Web mining*; • **Computing methodologies** → Machine learning;

KEYWORDS

peer-production systems; user-generated content; collaborative filtering; ranking

*This work was done while the author was at EPFL.

KDD 2018, August 19–23, 2018, London, United Kingdom
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219979

1 INTRODUCTION

Over the last two decades, the number and scale of online peer-production systems has become truly massive, driven by better information networks and advances in collaborative software. At the time of writing, 128 643 editors contribute regularly to 5+ million articles of the English Wikipedia [34] and over 15 600 developers have authored code for the Linux kernel [7]. On GitHub, 24 million users collaborate on 25.3 million active software repositories [14].

In order to ensure that such projects advance towards their goals, it is necessary to identify whether edits made by users are beneficial. As the number of users and components of the project grows, this task becomes increasingly challenging. In response, two types of solutions are proposed. On the one hand, some advocate the use of *user reputation systems* [2, 25]. These systems are general, their predictions are easy to interpret and can be made resistant to manipulations [10]. On the other hand, a number of highly specialized methods are proposed to automatically predict the quality of edits in particular peer-production systems [12, 15]. These methods can attain excellent predictive performance [16] and usually significantly outperform predictors that are based on user reputation alone [12], but they are tailored to a particular peer-production system, use domain-specific features and rely on models that are difficult to interpret.

In this work, we set out to explore another point in the solution space. We aim to keep the generality and simplicity of user reputation systems, while reaching the predictive accuracy of highly specialized methods. We ask the question: Can one predict the outcome of contributions simply by observing *who edits what* and whether the edits eventually survive? We address this question by proposing a novel statistical model of edit outcomes. We formalize the notion of collaborative project as follows. N users can propose edits on M distinct items (components of the project, such as articles on Wikipedia or a software's modules), and we assume that there is a process for validating edits (either immediately or over time). We observe triplets (u, i, q) that describe a user $u \in \{1, \ldots, N\}$ editing an item $i \in \{1, \ldots, M\}$ and leading to outcome $q \in \{0, 1\}$; the outcome $q = 0$ represents a rejected edit, whereas $q = 1$ represents an accepted, beneficial edit. Given a dataset of such observations, we seek to learn a model of the probability p_{ui} that an edit made by user u on item i is accepted. This model can then be used to help moderators and project maintainers prioritize their efforts once new edits appear: For example, edits that are unlikely to survive could be sent out for review immediately.

Our approach borrows from probabilistic models of pairwise comparisons [24, 36]. These models learn a real-valued score for each object (user or item) such that the difference between two objects' scores is predictive of comparison outcomes. We take a similar perspective and view each edit in a collaborative project as a game between the user who tries to effect change and the item that resists change[1]. Similarly to pairwise-comparison models, our approach learns a real-valued score for each user and each item. In addition, it also learns latent features of users and items that capture interaction effects.

In contrast to quality-prediction methods specialized on a particular peer-production system, our approach is general and can be applied to any system in which users contribute by editing discrete items. It does not use any explicit content-based features: instead, it simply learns by observing triplets $\{(u, i, q)\}$. Furthermore, the resulting model parameters can be interpreted easily. They enable a principled way of *a*) ranking users by the quality of their contributions, *b*) ranking items by the difficulty of editing them and *c*) understanding the main dimensions of the interaction between users and items.

We apply our approach on two different peer-production systems. We start with Wikipedia and consider its Turkish and French editions. Evaluating the accuracy of predictions on an independent set of edits, we find that our model approaches the performance of the state of the art. More interestingly, the model parameters reveal important facets of the system. For example, we characterize articles that are easy or difficult to edit, respectively, and we identify clusters of articles that share common editing patterns. Next, we turn our attention to the Linux kernel. In this project, contributors are typically highly skilled professionals, and the edits that they make affect 394 different subsystems (kernel components). In this instance, our model's predictions are *more accurate* than a random forest classifier trained on domain-specific features. In addition, we give an interesting qualitative description of subsystems based on their difficulty score.

In short, our paper *a*) gives evidence that observing *who edits what* can yield valuable insights into peer-production systems and *b*) proposes a statistically grounded and computationally inexpensive method to do so. The analysis of two peer-production systems with very distinct characteristics demonstrates the generality of the approach.

Organization of the Paper. We start by reviewing related literature in Section 2. In Section 3, we describe our statistical model of edit outcomes and briefly discuss how to efficiently learn a model from data. In Sections 4 and 5, we investigate our approach in the context of Wikipedia and of the Linux kernel, respectively. Finally, we conclude in Section 6.

2 RELATED WORK

With the growing size and impact of online peer-production systems, the task of assessing contribution quality has been extensively studied. We review various approaches to the problem of quantifying and predicting the quality of user contributions and contrast them to our approach.

[1]Obviously, items do not really "resist" by themselves. Instead, this notion should be taken as a proxy for the combined action of other users (e.g., project maintainers) who can accept or reject an edit depending, among others, on standards of quality.

User Reputation Systems. Reputation systems have been a long-standing topic of interest in relation to peer-production systems and, more generally, in relation to online services [25]. Adler and de Alfaro [2] propose a point-based reputation system for Wikipedia and show that reputation scores are predictive of the future quality of editing. As almost all edits to Wikipedia are immediately accepted, the authors define an *implicit* notion of edit quality by measuring how much of the introduced changes is retained in future edits. The ideas underpinning the computation of implicit edit quality are extended and refined in subsequent papers [3, 10]. This line of work leads to the development of WikiTrust [11], a browser add-on that highlights low-reputation texts in Wikipedia articles. When applying our methods to Wikipedia, we follow the same idea of measuring quality implicitly through the state of the article at subsequent revisions. We also demonstrate that by automatically learning properties of the *item* that a user edits (in addition to learning properties of the user, such as a reputation score) we can substantially improve predictions of edit quality. This was also noted recently by Tabibian et al. [28] in a setting similar to ours, but using a temporal point process framework.

Specialized Classifiers. Several authors propose quality-prediction methods tailored to a specific peer-production system. Typically, these methods consist of a machine-learned classifier trained on a large number of content-based and system-based features of the users, the items and the edits themselves. Druck et al. [12] fit a maximum entropy classifier for estimating the lifespan of a given Wikipedia edit, using a definition of edit longevity similar to that of Adler and de Alfaro [2]. They consider features based on the edit's content (such as: number of words added / deleted, type of change, capitalization and punctuation, etc.) as well as features based on the user, the time of the edit and the article. Their model significantly outperforms a baseline that only uses features of the user. Other methods use support vector machines [6], random forests [6, 17] or binary logistic regression [23], with varying levels of success. In some cases, content-based features are refined using natural-language processing, leading to substantial performance improvements. However, these improvements are made to the detriment of general applicability. For example, competitive natural language processing tools have yet to be developed for the Turkish language (we investigate the Turkish Wikipedia in Section 4). In contrast to these methods, our approach is general and broadly applicable. Furthermore, the use of black-box classifiers can hinder the interpretability of predictions, whereas we propose a statistical model whose parameters are straightforward to interpret.

Truth Inference. In crowdsourcing, a problem related to ours consists of *jointly* estimating *a*) model parameters (such as user skills or item difficulties) that are predictive of contribution quality, and *b*) the quality of each contribution, without ground truth [9]. Our problem is therefore easier, as we assume access to ground-truth information about the outcome (quality) of past edits. Nevertheless, some methods developed in the crowdsourcing context [31, 32, 37] provide models that can be applied to our setting as well. In Sections 4 and 5, we compare our models to GLAD [32].

Pairwise Comparison Models. Our approach draws inspiration from probabilistic models of pairwise comparisons. These have been studied extensively over the last century in the context of psychometrics [5, 29], item response theory [24], chess rankings [13, 36], and more. The main paradigm posits that every object i has a latent *strength* (skill or difficulty) parameter θ_i, and that the probability p_{ij} of observing object i "winning" over object j increases with the distance $\theta_i - \theta_j$. Conceptually, our model is closest to that of Rasch [24].

Collaborative Filtering. Our method also borrows from collaborative filtering techniques popular in the recommender systems community. In particular, some parts of our model are remindful of matrix-factorization techniques [19]. These techniques automatically learn low-dimensional embeddings of users and items based on ratings, with the purpose of producing better recommendations. Our work shows that these ideas can also be helpful in addressing the problem of predicting outcomes of edits in peer-production systems. Like collaborative-filtering methods, our approach is exposed to the *cold-start* problem: with no (or few) observations about a given user or item, the predictions are notably less accurate. In practice, this problem can be addressed, e.g., by using additional features of users and / or items [21, 27] or by clustering users [22].

3 STATISTICAL MODELS

In this section, we describe and explain two variants of a statistical model of edit outcomes based on *who* edits *what.* In other words, we develop models that are predictive of the outcome $q \in \{0, 1\}$ of a contribution of user u on item i. To this end, we represent the probability p_{ui} that an edit made by user u on item i is successful. In collaborative projects of interest, most users typically interact with only a small number of items. In order to deal with the sparsity of interactions, we postulate that the probabilities $\{p_{ui}\}$ lie on a low-dimensional manifold and propose two model variants of increasing complexity. In both cases, the parameters of the model have intuitive effects and can be interpreted easily.

Basic Variant. The first variant of our model is directly inspired by the Rasch model [24]. The probability that an edit is accepted is defined as

$$p_{ui} = \frac{1}{1 + \exp[-(s_u - d_i + b)]}, \tag{1}$$

where $s_u \in \mathbf{R}$ is the *skill* of user u, $d_i \in \mathbf{R}$ is the *difficulty* of item i, and $b \in \mathbf{R}$ is a global parameter that encodes the overall skew of the distribution of outcomes. We call this model variant INTERANK *basic.* Intuitively, the model predicts the outcome of a "game" between an item with inertia and a user who would like to effect change. The *skill* quantifies the ability of the user to enforce a contribution, whereas the *difficulty* quantifies how "resistant" to contributions the particular item is.

Similarly to reputation systems [2], INTERANK *basic* learns a score for each user; this score is predictive of edit quality. However, unlike these systems, our model also takes into account that some items might be more challenging to edit than others. For example, on Wikipedia, we can expect high-traffic, controversial articles to be more difficult to edit than less popular articles. As with user skills, the article difficulty can be inferred *automatically* from observed outcomes.

Full Variant. Although the *basic* variant is conceptually attractive, it might prove to be too simplistic in some instances. In particular, the *basic* variant implies that if user u is more skilled than user v, then $p_{ui} > p_{vi}$ for *all* items i. In many peer-production systems, users tend to have their own specializations and interests, and each item in the project might require a particular mix of skills. For example, with the Linux kernel, an engineer specialized in file systems might be successful in editing a certain subset of software components, but might be less proficient in contributing to, say, network drivers, whereas the situation might be exactly the opposite for another engineer. In order to capture the multidimensional interaction between users and items, we add a bilinear term to the probability model (1). Letting $\boldsymbol{x}_u, \boldsymbol{y}_i \in \mathbf{R}^D$ for some dimensionality $D \in \mathbf{N}_{>0}$, we define

$$p_{ui} = \frac{1}{1 + \exp[-(s_u - d_i + \boldsymbol{x}_u^\top \boldsymbol{y}_i + b)]}. \tag{2}$$

We call the corresponding model variant INTERANK *full.* The vectors $\boldsymbol{x}_u$ and $\boldsymbol{y}_i$ can be thought of as embedding users and items as points in a latent D-dimensional space. Informally, p_{ui} increases if the two points representing a user and an item are close to each other, and it decreases if they are far from each other (e.g., if the vectors have opposite signs). If we slightly oversimplify, the parameter $\boldsymbol{y}_i$ can be interpreted as describing the set of skills needed to successfully edit item i, whereas $\boldsymbol{x}_u$ describes the set of skills displayed by user u.

The bilinear term is reminiscent of matrix-factorization approaches in recommender systems [19]; indeed, this variant can be seen as a *collaborative-filtering* method. In true collaborative-filtering fashion, our model is able to learn the latent feature vectors $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{y}_i\}$ *jointly*, by taking into consideration all edits and without any additional content-based features.

Finally, note that the skill and difficulty parameters are retained in this variant and can still be used to explain first-order effects. The bilinear term explains only the additional effect due to the user-item interaction.

3.1 Learning the Model

From (1) and (2), it should be clear that our probabilistic model assumes no data other than the identity of the user and that of the item. This makes it generally applicable to any peer-production system in which users contribute to discrete items.

Given a dataset of K independent observations $\mathcal{D} = \{(u_k, i_k, q_k) \mid k = 1, \ldots, K\}$, we infer the parameters of the model by maximizing their likelihood under $\mathcal{D}$. That is, collecting all model parameters into a single vector $\boldsymbol{\theta}$, we seek to minimize the negative log-likelihood

$$-\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{(u,i,q) \in \mathcal{D}} \left[-q \log p_{ui} - (1-q) \log(1 - p_{ui})\right], \tag{3}$$

where p_{ui} depends on $\boldsymbol{\theta}$. In the *basic* variant, the negative log-likelihood is convex, and we can easily find a global maximum by using standard methods from convex optimization. In the *full* variant, the bilinear term breaks the convexity of the objective function, and we can no longer guarantee that we will find parameters that are global minimizers. In practice, we do not observe any convergence issues but reliably find good model parameters on all datasets.

Note that (3) easily generalizes from binary outcomes ($q \in \{0, 1\}$) to continuous-valued outcomes ($q \in [0, 1]$). Continuous values can be used to represent the *fraction* of the edit that is successful.

Implementation. We implement the models in Python by using the TensorFlow library [1]. Our code is publicly available online at https://github.com/lca4/interank. In order to avoid overfitting the model to the training data, we add a small amount of ℓ_2 regularization to the negative log-likelihood. We minimize the negative log-likelihood by using stochastic gradient descent [4] with small batches of data. For INTERANK *full*, we set the number of latent dimensions to $D = 20$ by cross-validation.

Running Time. Our largest experiment consists of learning the parameters of INTERANK *full* on the entire history of the French Wikipedia (c.f. Section 4), consisting of over 65 million edits by 5 million users on 2 million items. In this case, our TensorFlow implementation takes approximately 2 hours to converge on a single machine. In most other experiments, our implementation takes only a few minutes to converge. This demonstrates that our model effortlessly scales, even to the largest peer-production systems.

3.2 Applicability

Our approach models the difficulty of effecting change through the affected item's identity. As such, it applies particularly well to peer-production systems where users *cooperate* to improve the project, i.e., where each edit is judged independently against an item's (latent) quality standards. This model is appropriate for a wide variety of projects, ranging from online knowledge bases (such as Wikipedia, c.f. Section 4) to open source software (such as the Linux kernel project, c.f. Section 5). In some peer-production systems, however, the contributions of different users *compete* against each other, such as multiple answers to a single question on a Q&A platform. In these cases, our model can still be applied, but fails to capture the fact that edit outcomes are interdependent.

4 WIKIPEDIA

Wikipedia is a popular free online encyclopedia and arguably one of the most successful peer-production systems. In this section, we apply our models to the French and Turkish editions of Wikipedia.

4.1 Background & Datasets

The French Wikipedia is one of the largest Wikipedia editions. At the time of writing, it ranks in third position both in terms of number of edits and number of users[2]. In order to obtain a complementary perspective, we also study the Turkish Wikipedia, which is roughly an order of magnitude smaller. Interestingly, both the French and the Turkish editions score very highly on Wikipedia's *depth* scale, a measure of collaborative quality [33].

The Wikimedia Foundation releases periodically and publicly a database dump containing the successive revisions to all articles[3]. In this paper, we use a dump that contains data starting from the beginning of the edition up to the fall of 2017.

[2]We chose the French edition over the English one because our computing infrastructure could not support the ≈ 15 TB needed to store the entire history of the English Wikipedia. The French edition contains roughly 5× fewer edits.

[3]See: https://dumps.wikimedia.org/.

4.1.1 Computation of Edit Quality. On Wikipedia, any user's edit is immediately incorporated into the encyclopedia[4]. Therefore, in order to obtain information about the quality of an edit, we have to consider the implicit signal given by subsequent edits to the same article. If the changes introduced by the edit are preserved, it signals that the edit was beneficial, whereas if the changes are reverted, the edit likely had a negative effect. A formalization of this idea is given by Adler and de Alfaro [2] and Druck et al. [12]; see also de Alfaro and Adler [10] for a concise explanation. In this paper, we essentially follow their approach.

Consider a particular article and denote by v_k its k-th revision (i.e., the state of the article after the k-th edit). Let $d(u, v)$ be the Levenshtein distance between two revisions [20]. We define the *quality* of edit k from the perspective of the article's state after $\ell \geq 1$ subsequent edits as

$$q_{k|\ell} = \frac{1}{2} + \frac{d(v_{k-1}, v_{k+\ell}) - d(v_k, v_{k+\ell})}{2d(v_{k-1}, v_k)}.$$

By properties of distances, $q_{k|\ell} \in [0, 1]$. Intuitively, the quantity $q_{k|\ell}$ captures the proportion of work done in edit k that remains in revision $k + \ell$. It can be understood as a *soft* measure of whether edit k has been reverted or not. We compute the unconditional quality of the edit by averaging over multiple future revisions:

$$q_k = \frac{1}{L} \sum_{\ell=1}^{L} q_{k|\ell}, \tag{4}$$

where L is the minimum between the number of subsequent revisions of the article and 10 (we empirically found that 10 revisions is enough to accurately assess the quality of an edit). Note that even though q_k is no longer binary, our models naturally extend to continuous-valued $q_k \in [0, 1]$ (c.f. Section 3.1).

In practice, we observe that edit quality is bimodal and asymmetric. Most edits have a quality close to either 0 or 1 and a majority of edits are of high quality. The two rightmost columns of Table 1 quantify this for the French and Turkish editions.

4.1.2 Dataset Preprocessing. We consider all edits to the pages in the main namespace (i.e., articles), including those from anonymous contributors identified by their IP address[5]. Sequences of consecutive edits to an article by the same user are collapsed into a single edit in order to remove bias in the computation of edit quality [2]. To evaluate methods in a realistic setting, we split the data into a training set containing the first 90 % of edits, and we report results on an independent validation set containing the remaining 10 %. Note that the quality is computed based on subsequent revisions of an article: In order to guarantee that the two sets are truly independent, we make sure that we never use any revisions from the validation set to compute the quality of edits in the training set. A short summary of the data statistics after preprocessing is provided in Table 1.

4.2 Evaluation

In order to facilitate the comparison of our method with competing approaches, we evaluate the performance on a binary classification

[4]Except for a small minority of protected articles.

[5]Note, however, that a large majority of edits are made by registered users (82.7 % and 76.6 % for the French and Turkish editions, respectively).

Table 1: Summary statistics of Wikipedia datasets after preprocessing.

Edition	# users N	# articles M	# edits	First edit	Last edit	% edits with $q < 0.2$	% edits with $q > 0.8$
French	5 460 745	1 932 810	65 430 838	2001-08-04	2017-09-02	6.4 %	72.2 %
Turkish	1 360 076	310 991	8 768 258	2002-12-05	2017-10-01	11.6 %	60.5 %

task consisting of predicting whether an edit is of poor quality. To this end, we assign binary labels to all edits in the validation set: the label *bad* is assigned to every edit with $q < 0.5$, and the label *good* is assigned to all edits with $q \geq 0.5$. The predictions of the classifier might help Wikipedia administrators to identify edits of low quality; these edits might then be sent to domain experts for review.

As discussed in Section 3, we consider two versions of our model. The first one, INTERANK *basic*, simply learns scalar user skills and article difficulties. The second one, INTERANK *full*, additionally includes a latent embedding of dimension $D = 20$ for each user and article.

4.2.1 Competing Approaches. To set our results in context, we compare them to those obtained with four different baselines.

Average. The first approach always outputs the marginal probability of a bad edit in the training set, i.e.,

$$p = \frac{\text{\# bad edits in training set}}{\text{\# edits in training set}}$$

This is a trivial baseline, and it gives an idea of what results we should expect to achieve without any additional information on the user, article or edit.

User-Only. The second approach models the outcome of an edit using only the user's identity. In short, the predictor learns skills $\{s_u \mid u = 1, \ldots, N\}$ and a global offset b such that, for each user u, the probability

$$p_u = \frac{1}{1 + \exp[-(s_u + b)]}$$

maximizes the likelihood of that user's edits in the training set. This baseline predictor is representative of user reputation systems such as that of Adler and de Alfaro [2].

GLAD. In the context of crowdsourcing, Whitehill et al. [32] propose the GLAD model that postulates that

$$p_{ui} = \frac{1}{1 + \exp(-s_u/d_i)},$$

where $s_u \in \mathbf{R}$ and $d_i \in \mathbf{R}_{>0}$. This reflects a different assumption on the interplay between user skill and item difficulty: under their model, an item with a large difficulty value makes every user's skill more "diffuse". In order to make the comparison fair, we add a global offset parameter b to the model (similarly to INTERANK and the user-only baseline).

ORES reverted. The fourth approach is a state-of-the-art classifier developed by researchers at the Wikimedia Foundation as part of Wikipedia's Objective Revision Evaluation Service [15]. We use the two classification models specifically developed for the French and Turkish editions. Both models use over 80 content-based and system-based features extracted from the user, the article and the edit to predict whether the edit will be reverted, a target which essentially matches our operational definition of *bad* edit. Features include the number of vulgar words introduced by the edit, the length of the article and of the edit, etc. This predictor is representative of specialized, domain-specific approaches to modeling edit quality.

Table 2: Predictive performance on the *bad edit* classification task for the French and Turkish editions of Wikipedia. The best performance is highlighted in bold.

Edition	Model	Avg. log-likelihood	AUPRC
French	INTERANK *basic*	−0.339	0.399
	INTERANK *full*	**−0.336**	0.413
	Average	−0.389	0.131
	User-only	−0.346	0.313
	GLAD	−0.344	0.369
	ORES reverted	−0.469	**0.453**
Turkish	INTERANK *basic*	−0.380	0.494
	INTERANK *full*	**−0.379**	0.503
	Average	−0.461	0.168
	User-only	−0.390	0.410
	GLAD	−0.387	0.471
	ORES reverted	−0.392	**0.552**

4.2.2 Results. Table 2 presents the average log-likelihood and the area under the precision-recall curve (AUPRC) for each method. INTERANK *full* has the highest average log-likelihood of all models, meaning that its predictive probabilities are well calibrated with respect to the validation data.

Figure 1 (left and center) presents the precision-recall curves for all methods. The analysis is qualitatively similar for both Wikipedia editions. All non-trivial predictors perform similarly in the high-recall regime, but present significant differences in the high-precision regime, on which we will focus. The ORES predictor performs the best. INTERANK comes second, reasonably close behind ORES, and the *full* variant has a small edge over the *basic* variant. GLAD is next, and the user-only baseline is far behind. This shows that *a*) incorporating information about the article being edited is crucial for achieving a good performance on a large portion of the precision-recall trade-off, and *b*) modeling the outcome probability by using the *difference* between skill and difficulty (INTERANK) is better than by using the *ratio* (GLAD).

We also note that in the validation set, approximately 20 % (15 %) of edits are made by users (respectively, on articles) that are never encountered in the training set (the numbers are similar in both

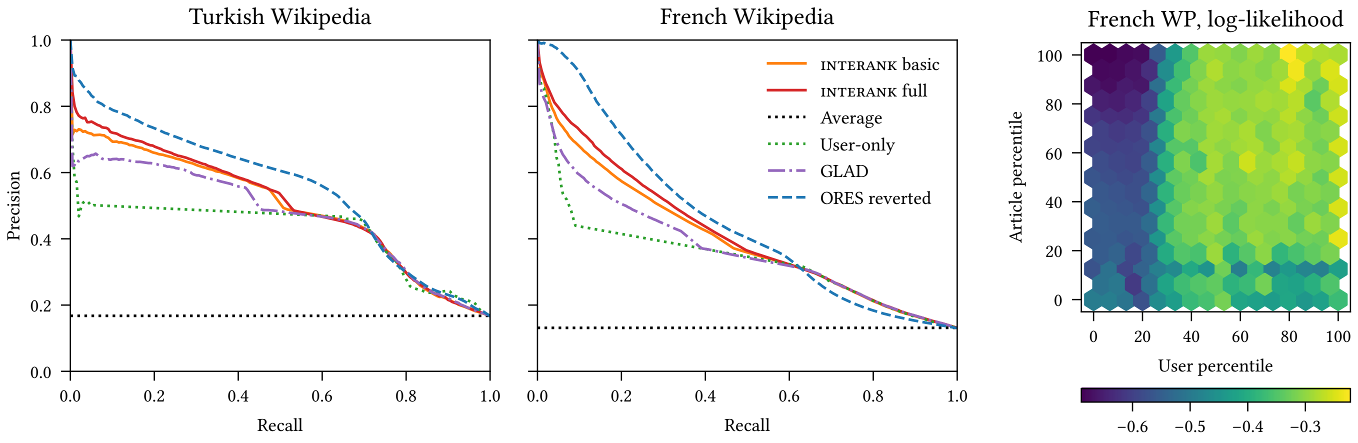

Figure 1: Precision-recall curves on the *bad edit* classification task for the Turkish and French editions of Wikipedia (left and center). Average log-likelihood as a function of the number of observations of the user and item in the training set (right).

editions). In these cases, INTERANK reverts to average predictions, whereas content-based methods can take advantage of other features of the edit to make an informed prediction. In order to explore this *cold-start* effect in more detail, we group users and articles into bins based on the number of times they appear in the training set, and we compute the average log-likelihood of validation examples separately for each bin. Figure 1 (right) presents the results for the French edition; the results for the Turkish edition are similar. Clearly, predictions for users and articles present in the training set are significantly better. In a practical deployment, several methods can help to address this issue [21, 22, 27]. A thorough investigation of ways to mitigate the cold-start problem is beyond the scope of this paper.

In summary, we observe that our model, which incorporates the articles' identity, is able to bridge the gap between user-only prediction approach and a specialized predictor (ORES reverted). Furthermore, modeling the interaction between user and article (INTERANK *full*) is beneficial and helps further improve predictions, particularly in the high-precision regime.

4.3 Interpretation of Model Parameters

The parameters of INTERANK models, in addition to being predictive of edit outcomes, are also very interpretable. In the following, we demonstrate how they can surface interesting characteristics of the peer-production system.

4.3.1 Controversial Articles. Intuitively, we expect an article i whose difficulty parameter d_i is large to deal with topics that are potentially controversial. We focus on the French Wikipedia and explore a list of the ten most controversial articles given by Yasseri et al. [35]. In this 2014 study, the authors identify controversial articles by using an ad-hoc methodology. Table 3 presents, for each article identified by Yasseri et al., the percentile of the corresponding difficulty parameter d_i learned by INTERANK *full*. We analyze these articles approximately four years later, but the model still identifies them as some of the most difficult ones. Interestingly, the article on Sigmund Freud, which has the lowest difficulty parameter of the list, has become a *featured* article since Yasseri et al.'s analysis—a distinction awarded only to the most well-written and neutral articles.

Table 3: The ten most controversial articles on the French Wikipedia according to Yasseri et al. [35]. For each article i, we indicate the percentile of its corresponding parameter d_i.

Rank	Title	Percentile of d_i
1	Ségolène Royal	99.840 %
2	Unidentified flying object	99.229 %
3	Jehovah's Witnesses	99.709 %
4	Jesus	99.953 %
5	Sigmund Freud	97.841 %
6	September 11 attacks	99.681 %
7	Muhammad al-Durrah incident	99.806 %
8	Islamophobia	99.787 %
9	God in Christianity	99.712 %
10	Nuclear power debate	99.304 %
	median	99.710 %

4.3.2 Latent Factors. Next, we turn our attention to the parameters $\{\boldsymbol{y}_i\}$. These parameters can be thought of as an embedding of the articles in a latent space of dimension $D = 20$. As we learn a model that maximizes the likelihood of edit outcomes, we expect these embeddings to capture latent article features that explain edit outcomes. In order to extract the one or two directions that explain most of the variability in this latent space, we apply principal component analysis [4] to the matrix $Y = [\boldsymbol{y}_i]$.

In Table 4, we consider the Turkish Wikipedia and list a subset of the 20 articles with the highest and lowest coordinates along the first principal axis of Y. We observe that this axis seems to distinguish articles about popular culture from those about "high culture" or timeless topics. This discovery supports the hypothesis that users have a propensity to successfully edit *either* popular culture *or* high-culture articles on Wikipedia, but not *both.*

Table 4: A selection of articles of the Turkish Wikipedia among the top-20 highest and lowest coordinates along the first principal axis of the matrix Y.

Direction	Titles
Lowest	Harry Potter's magic list, List of programs broadcasted by Star TV, Bursaspor 2011-12 season, Kral Pop TV Top 20, Death Eater, Heroes (TV series), List of programs broadcasted by TV8, Karadayı, Show TV, List of episodes of Kurtlar Vadisi Pusu.
Highest	Seven Wonders of the World, Thomas Edison, Cell, Mustafa Kemal Atatürk, Albert Einstein, Democracy, Isaac Newton, Mehmed the Conqueror, Leonardo da Vinci, Louis Pasteur.

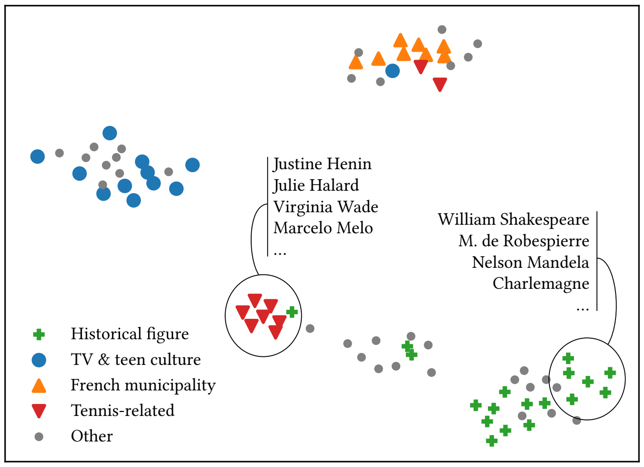

Figure 2: t-SNE visualization of 80 articles of the French Wikipedia with highest and lowest coordinates along the first and second principal axes of the matrix Y.

Finally, we consider the French Wikipedia. Once again, we apply principal component analysis to the matrix Y and keep the first two dimensions. We select the 20 articles with the highest and lowest coordinates along the first two principal axes[6]. A two-dimensional t-SNE plot [30] of the 80 articles selected using PCA is displayed in Figure 2. The plot enables identifying meaningful clusters of related articles, such as articles about tennis players, French municipalities, historical figures, and TV or teen culture. These articles are representative of the latent dimensions that separate editors the most: a user skilled in editing pages about ancient Greek mathematicians might be less skilled in editing pages about *anime*, and vice versa.

5 LINUX KERNEL

In this section, we apply the INTERANK model to the Linux kernel project, a well-known open-source software project. In contrast to Wikipedia, most contributors to the Linux kernel are highly skilled professionals who dedicate a significant portion of their time and efforts to the project.

5.1 Background & Dataset

The Linux kernel has fundamental impact on technology as a whole. In fact, the Linux operating system runs 90 % of the cloud workload and 82 % of the smartphones [7]. To collectively improve the source code, developers submit bug fixes or new features in the form of a *patch* to collaborative repositories. Review and integration time depend on the project's structure, ranging from a few hours or days for Apache Server [26] to a couple of months for the Linux kernel [18]. In particular for the Linux kernel, developers submit patches to subsystem mailing lists, where they undergo several rounds of reviews. After suggestions are implemented and if the code is approved, the patch can be committed to the subsystem maintainer's software repository. Integration conflicts are spotted at this stage by other developers monitoring the maintainer's repository and any issues must be fixed by the submitter. If the maintainer is satisfied with the patch, she commits it to Linus Torvalds' repository, who decides to include it or not with the next Linux release.

5.1.1 Dataset Preprocessing. We use a dataset collected by Jiang et al. [18] which spans Linux development activity between 2005 and 2012. It consists of 670 533 patches described using 62 features derived from e-mails, commits to software repositories, the developers' activity and the content of the patches themselves. Jiang et al. scraped patches from the various mailing lists and matched them with commits in the main repository. In total, they managed to trace back 75 % of the commits that appear in Linus Torvalds' repository to a patch submitted to a mailing list. A patch is labeled as *accepted* ($q = 1$) if it eventually appears in a release of the Linux kernel, and *rejected* ($q = 0$) otherwise. We remove data points with empty subsystem and developer names, as well as all subsystems with no accepted patches. Finally, we chronologically order the patches according to their mailing list submission time.

After preprocessing, the dataset contains $K = 619\,419$ patches proposed by $N = 9672$ developers on $M = 394$ subsystems. 34.12 % of these patches are accepted. We then split the data into training set containing the first 80 % of patches and a validation set containing the remaining 20 %.

5.1.2 Subsystem-Developer Correlation. Given the highly complex nature of the project, one could believe that developers tend to specialize in few, independent subsystems. Let $X_u = \{X_{ui}\}_{i=1}^{M}$ be the collection of binary variables X_{ui} indicating whether developer u has an accepted patch in subsystem i. We compute the sample Pearson correlation coefficient $r_{uv} = \rho(X_u, X_v)$ between X_u and X_v. We show in Figure 3 the correlation matrix $\boldsymbol{R} = [r_{uv}]$ between developers patching subsystems. Row $\boldsymbol{r}_u$ corresponds to developer u, and we order all rows according to the subsystem each developer u contribute to the most. We order the subsystems in decreasing order by the number of submitted patches, such that larger subsystems appear at the top of the matrix $\boldsymbol{R}$. Hence, the

[6]Interestingly, the first dimension has a very similar interpretation to that obtained on the Turkish edition: it can also be understood as separating popular culture from high culture.

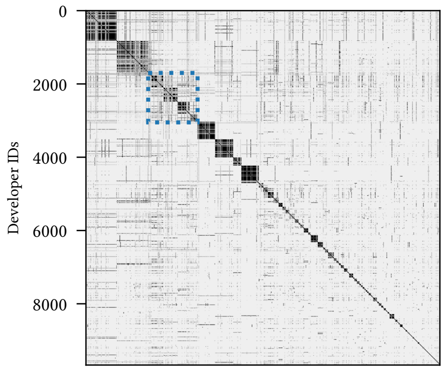

Figure 3: Correlation matrix R between developers ordered according to the subsystem they contribute to the most. The blocks on the diagonal correspond to subsystems. Core subsystems form a strong cluster (blue square).

blocks on the diagonal roughly correspond to subsystems and their size represents the number of developers involved with the subsystem. As shown by the blocks, developers tend to specialize into one subsystem. However, as the numerous non-zero off-diagonal entries reveal, they still tend to contribute substantially to other subsystems. Finally, as highlighted by the dotted, blue square, subsystems number three to six on the diagonal form a cluster. In fact, these four subsystems (`include/linux`, `arch/x86`, `kernel` and `mm`) are core subsystems of the Linux kernel.

5.2 Evaluation

We consider the task of predicting whether a patch will be integrated into a release of the kernel. Similarly to Section 4, we use INTERANK *basic* and INTERANK *full* with $D = 20$ latent dimensions to learn the developers' skills, the subsystems' difficulty, and the interaction between them.

5.2.1 Competing Approaches. Three baselines that we consider—*average*, *user-only* and *GLAD*—are identical to those described in Section 4.2.1. In addition, we also compare our model to a random forest classifier trained on domain-specific features similar to the one used by Jiang et al. [18]. In total, this classifier has access to 21 features for each patch. Features include information about the developer's experience up to the time of submission (e.g., number of accepted commits, number of patches sent), the e-mail thread (e.g., number of developers in copy of the e-mail, size of e-mail, number of e-mails in thread until the patch) and the patch itself (e.g., number of lines changed, number of files changed). We optimize the hyperparameters of the random forest using a grid-search. As the model has access to domain-specific features about each edit, it is representative of the class of specialized methods tailored to the Linux kernel peer-production system.

5.2.2 Results. Table 5 displays the average log-likelihood and area under the precision-recall curve (AUPRC). INTERANK *full* performs best in terms of both metrics. In terms of AUPRC, it outperforms the random forest classifier by 4.4 %, GLAD by 5 %, and the *user-only* baseline by 7.3 %.

Table 5: Predictive performance on the *accepted patch* classification task for the Linux kernel. The best performance is highlighted in bold.

Model	Avg. log-likelihood	AUPRC
INTERANK *basic*	-0.589	0.525
INTERANK *full*	**-0.588**	**0.527**
Average	-0.640	0.338
User-only	-0.601	0.491
GLAD	-0.598	0.502
Random forest	-0.599	0.505

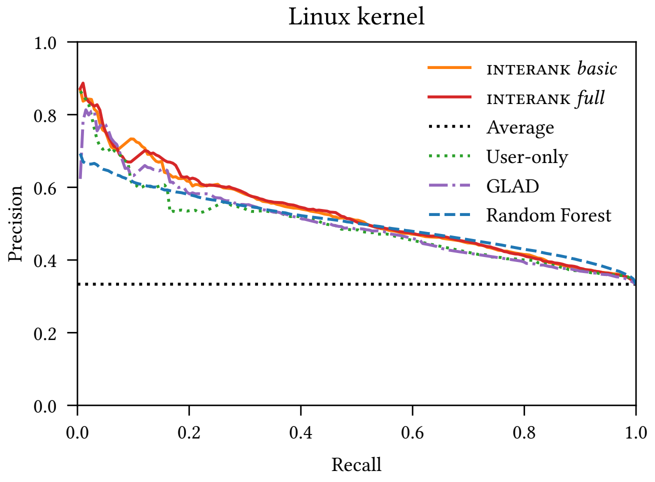

Figure 4: Precision-recall curves on the bad edit classification task for the Linux kernel. INTERANK (solid orange and red) outperforms the user-only baseline (dotted green), the random forest classifier (dashed blue), and GLAD (dash-dotted purple).

We show the precision-recall curves in Figure 4. Both INTERANK *full* and INTERANK *basic* perform better than the four baselines. Notably, they outperform the random forest in the high-precision regime, even though the random forest uses content-based features about developers, subsystems and patches. In the high-recall regime, the random forest attains a marginally better precision. The *user-only* and GLAD baselines perform worse than all non-trivial models.

5.3 Interpretation of Model Parameters

We show in Table 6 the top-five and bottom-five subsystems according to difficulties $\{d_i\}$ learned by INTERANK *full*. We note that even though patches submitted to difficult subsystems have in general low acceptance rate, INTERANK enables a finer ranking by taking into account *who* is contributing to the subsystems. This effect is even more noticeable with the five subsystems with smallest difficulty value.

The subsystems i with largest d_i are *core* components, whose integrity is crucial to the system. For instance, the `usr` subsystem, providing code for RAM-related instructions at booting time, has

Table 6: Top-five and bottom-five subsystems according to their difficulty d_i.

Difficulty	Subsystem	% Acc.	# Patch	# Dev.
+2.664	`usr`	1.88 %	796	70
+1.327	`include`	7.79 %	398	101
+1.038	`lib`	15.99 %	5642	707
+1.013	`drivers/clk`	34.34 %	495	81
+0.865	`include/trace`	17.73 %	547	81
-1.194	`drivers/addi-data`	78.31 %	272	8
-1.080	`net/tipc`	43.11 %	573	44
-0.993	`drivers/ps3`	44.26 %	61	9
-0.936	`net/nfc`	73.04 %	204	26
-0.796	`arch/mn10300`	45.40 %	359	63

barely changed in the last seven years. On the other hand, the subsystems i with smallest d_i are *peripheral* components serving specific devices, such as digital signal processors or gaming consoles. These components can arguably tolerate a higher rate of bugs, and hence they evolve more frequently.

Jiang et al. [18] establish that a high prior subsystem churn (i.e., high number of previous commits to a subsystem) leads to lower acceptance rate. We approximate the number of commits to a subsystem as the number of patches submitted multiplied by the subsystem's acceptance rate. The first quartile of subsystems according to their increasing difficulty, i.e., the least difficult subsystems, has an average churn of 687. The third quartile, i.e., the most difficult subsystems, has an average churn of 833. We verify hence that higher churn correlates with difficult subsystems. This corroborates the results obtained by Jiang et al.

As shown in Figure 4, if false negatives are not a priority, INTERANK will yield a substantially higher precision. In other words, if the task at hand requires that the patches classified as accepted are actually the ones integrated in a future release, then INTERANK will yield more accurate results. For instance, it would be efficient in supporting Linus Torvalds in the development of the Linux kernel by providing him with a restricted list of patches that are likely to be integrated in the next release of the Linux kernel.

6 CONCLUSION

In this paper, we have introduced INTERANK, a model of edit outcomes in peer-production systems. Predictions generated by our model can be used to prioritize the work of project maintainers by identifying contributions that are of high or low quality.

Similarly to user reputation systems, INTERANK is simple, easy to interpret and applicable to a wide range of domains. Whereas user reputation systems are usually not competitive with specialized edit quality predictors tailored to a particular peer-production system, INTERANK is able to bridge the gap between the two types of approaches, and it attains a predictive performance that is competitive with the state of the art—without access to content-based features.

We have demonstrated the performance of the model on two peer-production systems exhibiting different characteristics. Beyond predictive performance, we can also use model parameters to gain insight into the system. On Wikipedia, we have shown that the model identifies controversial articles, and that latent dimensions learned by our model display interesting patterns related to cultural distinctions between articles. On the Linux kernel, we have shown that inspecting model parameters enables to identify core subsystems (large difficulty parameters) from peripheral components (small difficulty parameters).

Future Work. In the future, we would like to investigate the idea of using the latent embeddings learned by our model in order to recommend items to edit. Ideally, we could match items that need to be edited with users that are most suitable for the task. For Wikipedia, an ad-hoc method called "SuggestBot" was proposed by Cosley et al. [8]. We believe it would be valuable to propose a method that is applicable to peer-production systems in general.

ACKNOWLEDGMENTS

We are grateful to Yujuan Jiang for providing the Linux data and to Aaron Halfaker for helping us understand ORES. We thank Patrick Thiran, Brunella Spinelli, Vincent Etter, Ksenia Konyushkova, Holly Cogliati-Bauereis and the anonymous reviewers for careful proofreading and constructive feedback.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of OSDI'16*. Savannah, GA, USA.

[2] B. Thomas Adler and Luca de Alfaro. 2007. A Content-Driven Reputation System for the Wikipedia. In *Proceedings of WWW'07*. Banff, AB, Canada.

[3] B. Thomas Adler, Luca de Alfaro, Ian Pye, and Vishwanath Raman. 2008. Measuring Author Contributions to the Wikipedia. In *Proceedings of WikiSym'08*. Porto, Portugal.

[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[5] Ralph Allan Bradley and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* 39, 3/4 (1952), 324–345.

[6] Amit Bronner and Christof Monz. 2012. User Edits Classification Using Document Revision Histories. In *Proceedings of EACL 2012*. Avignon, France.

[7] Jonathan Corbet and Greg Kroah-Hartman. 2017. *2017 Linux Kernel Development Report*. Technical Report. The Linux Foundation.

[8] Dan Cosley, Dan Frankowski, Loren Terveen, and John Riedl. 2007. SuggestBot: Using Intelligent Task Routing to Help People Find Work in Wikipedia. In *Proceedings of IUI'07*. Honolulu, HI, USA.

[9] Alexander Philip Dawid and Allan M Skene. 1979. Maximum Likelihood Estimation of Observer Error-rates using the EM Algorithm. *Applied Statistics* 28, 1 (1979), 20–28.

[10] Luca de Alfaro and B. Thomas Adler. 2013. Content-Driven Reputation for Collaborative Systems. In *Proceedings of TGC 2013*. Buenos Aires, Argentina.

[11] Luca de Alfaro, Ashutosh Kulshreshtha, Ian Pye, and B. Thomas Adler. 2011. Reputation Systems for Open Collaboration. *Commun. ACM* 54, 8 (2011), 81–87.

[12] Gregory Druck, Gerome Miklau, and Andrew McCallum. 2008. Learning to Predict the Quality of Contributions to Wikipedia. In *Proceedings of WikiAI 2008*. Chicago, IL, USA.

[13] Arpad Elo. 1978. *The Rating Of Chess Players, Past & Present*. Arco Publishing.

[14] GitHub. 2017. The State of the Octoverse 2017. https://octoverse.github.com/ Accessed: 2017-10-27.

[15] Aaron Halfaker and Dario Taraborelli. 2015. Artificial intelligence service "ORES" gives Wikipedians X-ray specs to see through bad edits. https://blog.wikimedia.org/2015/11/30/artificial-intelligence-x-ray-specs/ Accessed: 2017-10-27.

[16] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In *Proceedings of CIKM'16*. Indianapolis, IN, USA.

[17] Sara Javanmardi, David W. McDonald, and Cristina V. Lopes. 2011. Vandalism Detection in Wikipedia: A High-Performing, Feature-Rich Model and its Reduction Through Lasso. In *Proceedings of WikiSym'11*. Mountain View, CA, USA.

[18] Yujuan Jiang, Bram Adams, and Daniel M. German. 2013. Will My Patch Make It? And How Fast? Case Study on the Linux Kernel. In *Proceedings of MSR 2013*. San Francisco, CA, USA.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[20] Joseph B. Kruskal. 1983. An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules. *SIAM Rev.* 25, 2 (1983), 201–237.

[21] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. 2008. Addressing Cold-Start Problem in Recommendation Systems. In *Proceedings of ICUIMC'08*. Suwon, Korea.

[22] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. 2012. Finding a Needle in a Haystack of Reviews: Cold Start Context-Based Hotel Recommender System. In *Proceedings of RecSys'12*. Dublin, Ireland.

[23] Martin Potthast, Benno Stein, and Robert Gerling. 2008. Automatic Vandalism Detection in Wikipedia. In *Proceedings of ECIR 2008*. Glasgow, Scottland.

[24] Georg Rasch. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests*. Danmarks Pædagogiske Institut.

[25] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. 2000. Reputation systems. *Commun. ACM* 43, 12 (2000), 45–48.

[26] Peter C. Rigby, Daniel M. German, and Margaret-Anne Storey. 2008. Open Source Software Peer Review Practices: A Case Study of the Apache Server. In *Proceedings of ICSE'08*. Leipzig, Germany.

[27] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and Metrics for Cold-Start Recommendations. In *Proceedings of SIGIR'02*. Tampere, Finland.

[28] Behzad Tabibian, Isabel Valera, Mehrdad Farajtabar, Le Song, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. 2017. Distilling Information Reliability and Source Trustworthiness from Digital Traces. In *Proceedings of WWW'17*. Perth, WA, Australia.

[29] Louis L. Thurstone. 1927. A Law of Comparative Judgment. *Psychological Review* 34, 4 (1927), 273–286.

[30] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[31] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The Multidimensional Wisdom of Crowds. In *Advances in Neural Information Processing Systems 23*. Vancouver, BC, Canada.

[32] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Advances in Neural Information Processing Systems 22*. Vancouver, BC, Canada.

[33] Wikipedia. 2017. Wikipedia article depth. https://meta.wikimedia.org/wiki/Wikipedia_article_depth Accessed: 2017-10-30.

[34] Wikipedia. 2017. Wikipedia:Wikipedians. https://en.wikipedia.org/wiki/Wikipedia:Wikipedians Accessed: 2017-10-27.

[35] Taha Yasseri, Anselm Spoerri, Mark Graham, and János Kertész. 2014. The most controversial topics in Wikipedia: A multilingual and geographical analysis. In *Global Wikipedia: International and Cross-Cultural Issues in Online Collaboration*, Pnina Fichman and Noriko Hara (Eds.). Scarecrow Press.

[36] Ernst Zermelo. 1928. Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift* 29, 1 (1928), 436–460.

[37] Denny Zhou, Sumit Basu, Yi Mao, and John C Platt. 2012. Learning from the Wisdom of Crowds by Minimax Entropy. In *Advances in Neural Information Processing Systems 25*. Lake Tahoe, CA, USA.