

# ORES: Facilitating re-mediation of Wikipedia's socio-technical problems

AARON HALFAKER, Wikimedia Foundation, USA

JONATHAN T. MORGAN, Wikimedia Foundation, USA

AMIR SARABADANI, Wikimedia Deutschland, Germany

ADAM WIGHT, Wikimedia Foundation, USA

Intelligent algorithms have a long history of making curation work in peer production tractable. From counter-vandalism to task routing, basic machine prediction allows open knowledge projects like Wikipedia to scale to the largest encyclopedia in the world. However, the ideologies and values of the community were captured in the development of these algorithms and the processes they support. Wikipedia's challenges and the community's values have changed in the last decade, but its algorithmic support systems have remained largely stagnant. The conversation about what quality control should be and what place algorithms have remains restricted to a few expert engineers. In this paper, we describe ORES: an algorithmic service designed to open up socio-technical conversations in Wikipedia to a broader set of participants. In this paper, we argue the theoretical mechanisms of social change ORES enables and we describe the phenomena around ORES from the 3 years since ORES' deployment.

CCS Concepts: • **Networks** → **Online social networks**; • **Computing methodologies** → **Supervised learning by classification**; • **Applied computing** → **Sociology**; • **Software and its engineering** → **Software design techniques**; • **Computer systems organization** → **Cloud computing**;

Additional Key Words and Phrases: Wikipedia, Reflection, Systems, Machine learning, Transparency, Fairness, Successor, Margin, Algorithms, Governance, Articulation

## ACM Reference Format:

Aaron Halfaker, Jonathan T. Morgan, Amir Sarabadani, and Adam Wight. 2018. ORES: Facilitating re-mediation of Wikipedia's socio-technical problems. *ACM CSCW* , Article Under review (November 2018), 29 pages. <https://doi.org/0000001.0000001>

## 1 INTRODUCTION

Imagine a space where algorithms are developed and deployed through inclusive, transparent, community-led documentation practices. In such a place, what kind of dynamics would we expect to see?

Authors' addresses: Aaron Halfaker, Wikimedia Foundation, 1 Montgomery Street: Suite 1600, San Francisco, CA, 94104, USA, [ahalfaker@wikimedia.org](mailto:ahalfaker@wikimedia.org); Jonathan T. Morgan, Wikimedia Foundation, 1 Montgomery Street: Suite 1600, San Francisco, CA, 94104, USA, [jmorgan@wikimedia.org](mailto:jmorgan@wikimedia.org); Amir Sarabadani, Wikimedia Deutschland, Tempelhofer Ufer 23-24, 10963, Berlin, Germany, [amir.sarabadani@wikimedia.de](mailto:amir.sarabadani@wikimedia.de); Adam Wight, Wikimedia Foundation, 1 Montgomery Street: Suite 1600, San Francisco, CA, 94104, USA, [awight@wikimedia.org](mailto:awight@wikimedia.org).

This paper is published under the Creative Commons Attribution Share-alike 4.0 International (CC-BY-SA 4.0) license. Anyone is free to distribute and re-use this work on the conditions that the original authors are appropriately credited and that any derivative work is made available under the same, similar, or a compatible license.

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

0360-0300/2018/11-ARTUnder review

<https://doi.org/0000001.0000001>

Based on a reading of the critical algorithms literature, you'd think that we're just beginning to explore what such a space would look like. Government and big, corporate "platforms" dominate the discussion of algorithmic governance issues[4][6][12][42].

What if we told you that the Wikipedia<sup>1</sup> community has been developing algorithms—of the scary subjective type—in the open for more than a decade? And, that there's an extensive literature about how the social and technical governance subsystems have developed in parallel. In this field site, we see a unique set of new problems and opportunities that have yet to be encountered by the Facebooks, Twitters, and Googles of the internet.

By examining Wikipedia's algorithmic practices, we can explore research questions that are only askable after algorithmic development and adoption have become mostly *inclusive, transparent, and community-led*.

This paper discusses Wikipedia as a state-of-the-art socio-technical space for human-process and algorithmic co-development through the case of the Objective Revision Evaluation Service (ORES)<sup>2,3</sup>, a machine learning service that supports community-curated classifiers. ORES is novel as both a critical intervention and a technical probe[23] into Wikipedia's socio-technical problems around algorithmic transparency and crowd-sourced auditing. The design rationale behind ORES draws from empirical research into Wikipedia's socio-technical problems and theoretical literatures including critical feminist theory and genre ecologies. We discuss how these design rationales play out in the operation of the system and through case studies, where we show how independent communities of volunteers have responded to these AIs.

Since ORES's deployment in early 2015, it has been widely used by Wikipedians and discussed in scholarly literatures. Before it was enabled by default, the ORES service was manually enabled by roughly half of the population of Wikipedia editors across all languages. Our announcement blog post[19] and the ORES URL have been cited and footnoted several times across research fields, used as a basis for improved modeling work (e.g. [5]), as an example of a publicly available machine prediction API (e.g. [26]), and through using the prediction models to support other analyses (e.g. [35]).

By detailing ORES and discussing new technological and social interventions that ORES elicits, we open the doors for researchers to further build on this research platform[41]. Through case studies of emergent reflection and auditing of ORES, we bring some constructive, empirically-grounded, and practical considerations to recent calls for the audit-ability of algorithms[37].

We first review related literature around open algorithmic systems, then discuss the socio-technical context of Wikipedia and the design rationale that lead us to building ORES. Next, we describe how we engineered the ORES system to match Wikipedia work practices – including innovations we've made with regards to algorithmic *openness* and *transparency*. Finally, we present case studies of interesting uses and critiques of ORES' predictions.

## 2 RELATED WORK

### 2.1 The politics of algorithms

Algorithmic systems play increasingly crucial roles in the governance of social processes [12]. In online spaces, these systems help us deal with information overload problems: What search results best balance *relevance* and *importance*? Which books are most *related* to the ones a

<sup>1</sup><https://wikipedia.org>

<sup>2</sup><https://ores.wikimedia.org>

<sup>3</sup><http://enwp.org:/mw:ORES>

user likes? In other spaces, algorithmic systems help institutions run more efficiently: Who is least *risky* to loan money to? Where should police patrol to encounter the most crime? Software algorithms are increasingly used in answering such questions that have no single right answer and where prior human decisions used as training data can be problematic [42].

Algorithms designed to support work change work practices, shifting the dynamics of that work[4, 12, 48]. Software algorithms gain political relevance on par with other process-mediating artifacts (e.g. laws[25]). This increasing relevance of algorithms in social and political life renewed focus on questions of fairness and transparency<sup>4</sup>.

There are repeated calls to address power dynamics at play in algorithmic bias through transparency and accountability of the algorithms that govern public life and access to resources[6][37][24]. The field around effective transparency and accountability mechanisms is growing. We cannot fully address the scale of concerns in this rapidly shifting, but we find inspiration in Kroll et al's discussion of the potential and limitations of auditing and transparency.[24]

This paper discusses a specific political context (Wikipedia's algorithmic quality control and socialization practices) and the development of novel algorithms for support of these processes. We implement an algorithmic intervention using the unusual strategy of deploying a set of prediction algorithms as a service, leaving decisions about appropriation to our users and other technology developers. Instead of seeking to train the single best classifier and implement it in our own designs, we embrace public auditing and re-interpretations of our model's predictions as an *intended* and *desired* outcome. Extensive work on technical and social ways to achieve fairness and accountability generally do not discuss this kind of infrastructural intervention on communities of practice and their articulation work.

## 2.2 Machine prediction in support of open production

Open peer production systems have a long history of using machine learning in service of efficiency. For Wikipedia and related Wikimedia projects, vandalism detection and quality control has been paramount for practitioners and researchers. Article quality prediction models have also been explored and applied to help Wikipedians focus their work in the most beneficial places.

**Vandalism detection.** The damage detection problem in Wikipedia is one of great scale. English Wikipedia receives about 160,000 new edits every day, which immediately go live without review. Wikipedians embrace this risk as the nature of an open encyclopedia, but work tirelessly to maintain quality. Every damaging or offensive edit puts the credibility of the community at risk, so all edits must be reviewed as soon as possible.

As an information overload problem, filtering strategies have been hugely successful in Wikipedia, and we briefly review a decade of work in this area. Potthast et al.'s seminal paper[34] describes a strategy using logistic regressions to automatically detecting vandalism. Jacobi described *ClueBot*[2], a Wikipedia editing bot deployed to automatically revert obvious vandalism. Adler et al. (2011) summarized the state of the art in feature extraction and modeling strategies, building a classifier that aggregated all features extraction strategies [1]. In some cases, researchers directly integrated their prediction models into tools for Wikipedians to use (e.g. STiki[47], a machine-supported human-computation tool). Yet most of this modeling work remains in the literature and has not been incorporated into current tools in Wikipedia.

<sup>4</sup>See also <https://www.fatml.org/> for a conference devoted to these questions

Wikipedians have built and orchestrated a complex, multi-stage filter for incoming edits that is efficient and effective. Geiger & Halfaker quantitatively describe the temporal rhythm of edit review in Wikipedia[9]: First, automated bots revert vandalism that scores very highly according to a machine prediction model. Then “vandal fighters” use human-in-the-loop tools, where they review edits scored highly by machine prediction models, but not high enough for fully-automated reversion [11]. Edits that make it past these bots and “cyborgs”[18] are routed through a system of *watchlists*<sup>5</sup> to experienced Wikipedia editors who are interested in the articles being edited. With this system in place, most damaging edits are reverted within seconds of when they are saved[9] and Wikipedia is kept clean.

**Task routing.** Task routing in Wikipedia is supported by a natural dynamic: people read what they are interested in, and when they see an opportunity to contribute, they do. This leads to a demand-driven contribution pattern where the most viewed content tends to be edited to the highest quality[22]. There are still many cases where Wikipedia remains misaligned[46], and content coverage biases creep in (e.g. for a long period of time, the coverage of Women Scientists in Wikipedia lagged far behind the rest of the encyclopedia[15]). By aligning interests with missed opportunities for contribution, these misalignments and gaps can be re-aligned and filled. Past work has explored collaborative recommender-based task routing strategies (see SuggestBot[3]) and shown good success. Recently, the maintainers of SuggestBot have developed article quality prediction models to help route attention to low quality articles[44]. Warncke-Wang and Halfaker have also used the article quality model to perform some one-off analyses to help Wikipedians critique and update their own manual quality assessments[45].

### 2.3 The Rise and Decline: Wikipedia’s socio-technical problems

While Wikipedians have made significant success in the algorithmic systems developed to support quality control, a line of critical research has studied the unintended consequences of this complex socio-technical system, particularly on newcomer socialization. [16][28][17]. In summary, Wikipedians struggled with the issues of scaling when the popularity of Wikipedia grew exponentially between 2005 and 2007[16]. In response, they developed quality control processes and technologies that prioritized efficiency by using machine prediction models[17] and templated warning messages[16]. This transformed newcomer socialization from a primarily human and welcoming activity to one that is more dismissive and impersonal[28]. This is a good example of the values of the designers being captured in the process and supporting infrastructure they developed[17]. The efficiency of quality control work and the elimination of damage was considered extremely politically important, while the positive experience of newcomers was less politically important. The result was a sudden and sustained decline in the retention of good-faith newcomers and a decline in the overall population of Wikipedia editors[16].

After the effect of this trade-off was made clear, a number of initiatives were started in an effort to more effectively balance the needs for good community management with quality control efficiency. For example, a newcomer help space (The Teahouse[28]) was developed to provide a more welcoming and forgiving space for newcomers to ask questions and meet experienced Wikipedians. A newcomer training game was developed and tested with financial support from the Wikimedia Foundation[33]. The Wikimedia Foundation also formed a product development team that was tasked with making changes to Wikipedia’s

<sup>5</sup><http://enwp.org:/mw:Help:Watchlist>

user interfaces to increase newcomer retention<sup>6</sup>. Most of these efforts did not show gains in newcomer retention under experimental conditions. An exception is the Teahouse, where it was shown that intervening by inviting newcomers to participate in the question and answer space had a statistically significant benefit to long term newcomer retention[29].

Despite these targeted efforts and shifts in perception among some members of the Wikipedia community, the quality control process that was designed over a decade ago remains largely unchanged[17]. The quality control systems that were dominant before the publication of the seminal “Rise and Decline” study in 2013[16] remain dominant today. The regime of automatic reverting bots and semi-automated tools that conceptualize edits as “good” and “bad” remains in place and unchanged. Notably, Halfaker et al. experimented with developing a novel reversal of the damage detection tools by re-appropriating a damage detection model to highlight good editors who were running into trouble in the quality control system[17], but there has not been a significant shift in how quality control is enacted in Wikipedia.

### 3 DESIGN RATIONALE

In this section, we discuss systemic mechanisms behind Wikipedia’s socio-technical problems and how we as system builders make positive impact. Past work demonstrated how Wikipedia’s problems are systemic with no readily apparent system-level solutions. To responsibly use machine learning in addressing these problems, we examined how Wikipedia functions as a distributed system: how processes, policies, power, and software come together to make Wikipedia happen.

**Wikipedia as a genre ecology.** Unlike traditional mass-scale projects, Wikipedia’s structure and processes are not centrally planned. Wikipedia’s system is a heterogeneous assemblage of humans, practices, policies, and software. This open system and its processes are dynamic, complex, and non-deterministic.

Genre ecologies is a theoretical framework we found useful in helping us account for the totality of factors and their relationships in Wikipedia, which is essential to building a system-level understanding of state and change processes. A genre ecology is “an interrelated group of genres (artifact types and the interpretive habits that have developed around them) used to jointly mediate the activities that allow people to accomplish complex objectives.”[40]. The genre ecology framework arose from observational research on how collaborators amend and repurpose existing officially-sanctioned tools — e.g. written documents, technological interfaces — and developed their own unofficial tools to supplement or circumvent official tools. This was needed in order to account for practical contingencies and emergent needs not anticipated, a longstanding concern in CSCW. In tracing the relationships among the set of official and unofficial tools used communities of practice, this literature helps conceptualize the relationships between tool genres and individual human actors, addressing issues of distributed agency, interdependency, rule formalization, and power dynamics in sociotechnical systems[39].

Morgan & Zachry used genre ecologies to characterize the relationships between Wikipedia’s official policies and “essays:” unofficial rules, best practices, and editing advice documents that are created by editors in order to contextualize, clarify, and contradict policies[30]. In Wikipedia, essays and policies co-exist and interact. For example, the “proper” interpretation of Wikipedia’s official Civility policy<sup>7</sup> within a particular context may need to account for the

<sup>6</sup><http://enwp.org/m:Growthteam>

<sup>7</sup><http://enwp.org/WP:CIVIL>

guidance provided in related essays such as “No Angry Mastodons”<sup>8</sup>. In genre ecology terms, performing the work of enforcing civil behavior on Wikipedia is dynamically and contingently *mediated* by the guidance provided in the official policy and the guidance provided in any related essays. Unofficial genres provide interpretive flexibility in the application of official rules to local circumstances as well as challenging and re-interpreting official ideologies and objectives.

Algorithmic systems also have a role in mediating the policy, values, rules, and emergent practices in social spaces as well[25? ?]. When looking at Wikipedia’s articulation work through the genre ecology lens, bots and human-computation tools mediate the meaning of policies and how Wikipedia enacts quality controls. For example, Sinebot enforces the signature policy in certain ways but not others[7], and the “Huggle” vandal fighting tool enacts quality in Wikipedia as a task of rapidly distinguishing good from bad edits in a live stream of recent changes[17]).

**Wikipedia’s problems with automated mediation.** As discussed in section 2, through the codevelopment of collaborative work processes and technologies to support them, Wikipedians traded soft, human newcomer socialization for hard, efficient quality control. Their software focused narrowly on most important problem to them — removing damage at all costs — and formalized processes that were previously more flexible and contingent. As they scaffolded their quality control workflows and values in code, other forms of quality control and socialization were pushed to the margins. The result was a marked decline in the retention of new editors in Wikipedia and a new threat to the core values of the project.

Where does change come from in a distributed cognition system that emerged based on community needs and volunteer priorities, where problematic assumptions have been embedded in the mediation of policy and the design of software for over a decade? Or maybe more generally, how does deep change take place in a genre ecology?

**Making change is complicated by the distributed nature.** As previously discussed, several initiatives were created to improve Wikipedian socialization practices, including the Teahouse and outreach efforts like Inspire Campaigns[27], which elicited ideas from contributors on the margins of the community. However, the process of quality control has remained largely unchanged. This assemblage of mindsets, policies, practices, and software prioritizes quality/efficiency and does so effectively [9][17] but at a cost.

Instead of pursuing the tempting technical solutions to *just fix quality control*, it is not at all apparent what better quality control would look like. Even if we did, how does one cause systemic change in a decentralized system like Wikipedia? We draw from standpoint epistemology, specifically Harding and Haraway’s concept of *successors*[20][21] in reflecting on the development of new software/process/policy components. Past work has explored developing a successor view that prioritizes the standpoints of mentors in support of new editors in Wikipedia, rather than the standpoints of vandal fighters focused on the efficiency of quality control[17][8]. However, a single point rarely changes the direction of an entire conversation or the shape of an entire ecology, so change is still elusive.

From these efforts, we know there is general interest in balancing quality/efficiency and diversity/welcomingness more effectively. So where are these designers who incorporate this expanded set of values? How to we help them bring forward their alternatives? How do we

<sup>8</sup><http://enwp.org/WP:MASTODON>



help them re-mediate Wikipedia's policies and values through their lens? How do we support the development of more successors?

**Expanding the margins of the ecology.** Successors come from the margin: they represent non-dominant values and engage in the re-mediation of articulation[32]. In our view, such successors are a primary means to change in an open genre ecology like Wikipedia. For anyone looking to enact a new view of quality control into the designs of a software system, there is a high barrier to entry: the development of a realtime machine prediction model. Without exception, all of the critical, high efficiency quality control systems that keep Wikipedia clean of vandalism and other damage employ a machine prediction model for highlighting the edits that are most likely to be bad. For example, Huggle<sup>9</sup> and STiki<sup>10</sup> use a machine prediction models to highlight likely damaging edits for human reviews. ClueBot NG<sup>11</sup> uses a machine prediction model to automatically revert edits that are highly likely to be damaging. These automated tools and their users work to employ a multi-stage filter that quickly and efficiently addresses vandalism[9].

Wikipedians have long had extensive discussions and debates about the development of the thousands of relatively simple rule-based bots that are tasked with enforcing rules or supporting various tasks [7]. In contrast, there are high barriers to entry around machine learning classification models for quality control, both in knowing how they work and how to develop and operate them at Wikipedia's scale. Without these skills, it was not possible for the average Wikipedian to create an alternative view of what quality controls should be, while also accounting for efficiency and the need to scale. Notably, one of the key interventions in this area that did do so was also built by a computer scientist[17].

The result is a dominance of a certain type of individual: a computer scientist or software engineer, who, as the stereotype goes, works with an eye towards efficiency but has little interest in messy human interactions. This high barrier to entry and in-group effects has exacerbated the minimization of the margin and a supreme dominance of the authority of quality control regimes that were largely developed in 2006—long before the social costs of efficient quality control were understood.

**Lowering the barriers to entry.** Wikipedia's quality control processes are open to the development of successor systems for re-mediating quality control, but only for those with the right skills. We have two options for expanding the margins: (1) increase general literacy around machine classification techniques and operations at scale; or (2) minimize the need to deeply understand practical machine learning at scale in order to develop an effective quality control tool.

The development of ORES is the second option. By deploying a high-availability machine prediction service that supports multiple classifiers at scale, designing accessible interfaces to engage with such classifiers in various ways, and engaging in basic outreach efforts, we sought to dramatically lower the barriers to the development of successor systems. In lowering the barriers to alternative visions of what quality control and newcomer socialization in Wikipedia should look like, we also open the doors to participation of alternative views in the genre ecology around quality control. For us, we measure success not through higher rates of precision and recall, but instead through the new conversations about how algorithmic tools affect editing dynamics, as well as new types of tools that take advantage of these resources, implementing alternative visions of what Wikipedia is and ought to be.

<sup>9</sup><http://enwp.org/WP:Huggle>

<sup>10</sup><http://enwp.org/WP:STiki>

<sup>11</sup>[http://enwp.org/User:ClueBot\\_NG](http://enwp.org/User:ClueBot_NG)

## 4 THE ORES SYSTEM

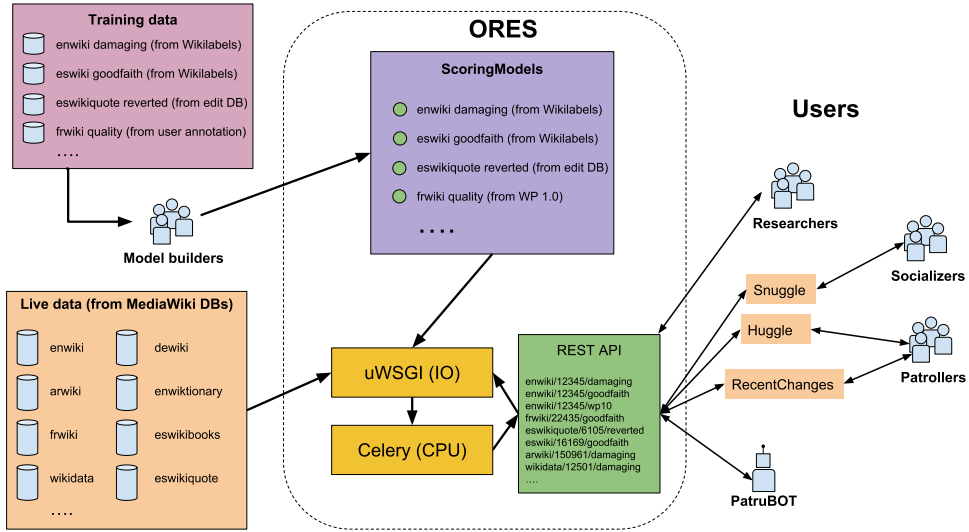


Fig. 1. ORES conceptual overview. Model builders design process for training ScoringModels from training data. ORES hosts ScoringModels and makes them available to researchers and tool developers.

ORES has been iteratively engineered to meet the needs of Wikipedia editors and the tools that support their work. In this section, we describe the architecture of the system and how architectural decisions allowed ORES to be integrated into different types of workflows.

### 4.1 Conceptual architecture

At the core, ORES is a collection of machine classifier models and an API. These models are designed and engineered by a varied set of model builders (some external researchers and others by our own research team) using varied sources of *training data*. The models that ORES hosts are engineered to support Wikipedia processes related to damage-detection, quality-assessment, and topic-routing, but the system is adaptable to a wide range of other models.

To make these models available for user, ORES implements a simple container service where the “container”, referred to as a *ScoringModel*, represents a fully trained and tested prediction model. All *ScoringModels* contain metadata about when the model was train/tested and which features are necessary for making a prediction. All predictions take the form of a JSON document. The ORES service provides access to ScoringModels via a RESTful HTTP interface and serves the predictions (JSON documents) to users. We chose this service structure because Wikimedian tool developers (our target audience) are familiar with this RESTful API/JSON workflow due to the dominant use of the MediaWiki API among tool developers.

### 4.2 Scaling & robustness

To be useful for Wikipedians and tool developers, ORES uses distributed computation strategies to provide a robust, fast, high-availability service. Reliability is a critical concern in Wikipedia’s algorithmic systems have



historically led to increased burdens for human workers and a higher likelihood that readers will see vandalism[9]. As previously discussed, we wanted to support Wikipedians who did not have the access and ability to operate a machine learning classifier at datacenter scales. The widespread use of ORES for multiple models across language versions of Wikimedia wikis could potentially involve orders of magnitude more computation than has ever been used for machine learning in Wikipedia.

This horizontal scalability is achieved in two ways: input-output (IO) workers (`uwsgi`<sup>12</sup>) and the computation (CPU) workers (`celery`<sup>13</sup>). Requests are split across available IO workers, and all necessary data is gathered using external APIs (e.g. the MediaWiki API<sup>14</sup>). The data is then split into a job queue managed by `celery` for the CPU-intensive work. This efficiently uses available resources and can dynamically scale, adding and removing new IO and CPU workers in multiple datacenters as needed. This is also fault-tolerant, as servers can fail without failing the service as a whole.

### 4.3 Real-time processing

The most common use case of ORES is real-time processing of edits. For example, those using counter-vandalism tools like Huggle monitor edits within seconds of when they are made. It is critical that ORES return these requests return in a timely manner. We implement several strategies to optimize this request pattern.

**Single score speed.** In the worst case scenario, ORES is generating a score from scratch. This is the common case when a score is requested in real-time—which invariably occurs right after the target edit or article is saved. We work to ensure that the median score duration is around 1 second. Our metrics tracking currently suggests that for the week April 6-13th, our median, 75%, and 95% score response timings are 1.1, 1.2, and 1.9 seconds respectively.

**Caching and precaching.** In order to take advantage of our users’ overlapping interests in scoring recent activity, we also maintain a basic least-recently-used (LRU) cache<sup>15</sup> using a deterministic score naming scheme (e.g. `enwiki:123456:damaging` would represent a score needed for the English Wikipedia damaging model for the edit identified by 123456). This allows requests for scores that have recently been generated to be returned within about 50ms via HTTPS. In other words, a request for a recent edit that had previously been scored is 20X faster due to this cache.

In order to make sure that scores for *all recent edits* are available in the cache for real-time use cases, we implement a “precaching” strategy that listens to a high-speed stream of recent activity in Wikipedia and automatically requests scores for a specific subset of actions (e.g. edits). With our LRU and pre-caching strategy, we attain a cache hit rate of about 80% consistently.

**De-duplication.** In real-time ORES use cases, it’s common to receive many requests to score the same edit/article right after it was saved. We use the same deterministic score naming scheme from the cache to identify scoring tasks, and ensure that simultaneous requests for that same score attach to the same result (or pending result) rather than starting a duplicate scoring job. This pattern is very advantageous in the case of precaching, because

<sup>12</sup><https://uwsgi-docs.readthedocs.io/>

<sup>13</sup><http://www.celeryproject.org/>

<sup>14</sup><http://enwp.org/mw:MW:API>

<sup>15</sup>Implemented natively by Redis, <https://redis.io>

of our network latency advantage: we can generally guarantee that the precaching request for a specific score precedes the external request for a score. All waiting requests attach to the result of a single score generation process that starts prior to receiving the external request. So even in worst-case scenarios where we're still calculating scores, we often see a better-than-expected response speed from the tool user's point of view.

#### 4.4 Batch processing

Many different types of Wikipedia's bots rely on batch processing strategies to support Wikipedia work processes[7], so ORES needs to support sudden, high-intensity querying. For example, many bots are designed to build worklists for Wikipedia editors (e.g. [3]) and recently, many of them have adopted ORES to include an article quality prediction for use in prioritization (see section 6). Work lists are either built from the sum total of all 5m+ articles in Wikipedia, or from some large subset specific to a single WikiProject (e.g. WikiProject Women Scientists claims about 6k articles<sup>16</sup>). We've observed robots submitting large batch processing jobs to ORES once per day. It's relevant to note that many researchers are also making use of ORES for various analyses, and their activity usually shows up in our logs as a similar burst of requests.

In order to most efficiently support this type of querying activity, we implemented batch optimizations in ORES by splitting IO and CPU operations into distinct stages. During the IO stage, all data is gathered to generate a set of scores. During the CPU stage, scoring jobs are split across our distributed processing system. This batch processing affords up to a 5X increase in time to scoring speed for large requests[38]. At this rate, a user can score 1 million revisions in less than 24 hours in the worst case scenario (no scores were cached)—which is unlikely for recent Wikipedia activity.

#### 4.5 Empirical access patterns

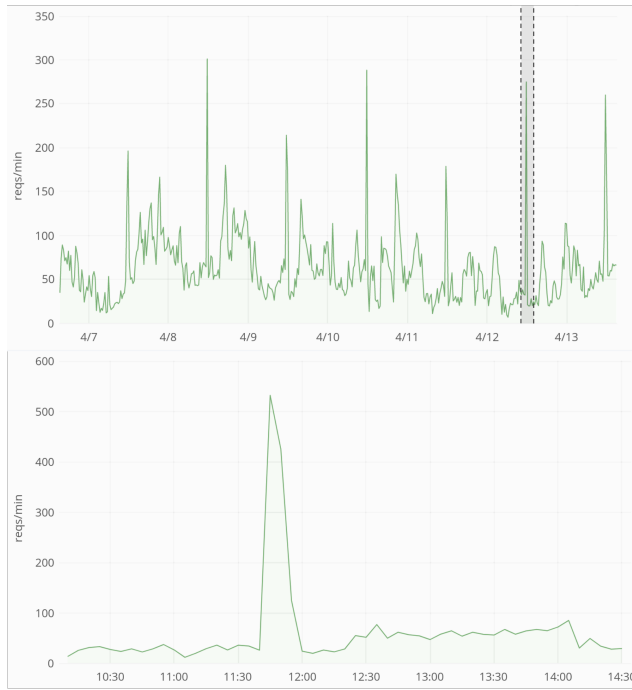
The ORES service has been online since July 2015[19]. Since then, usage has steadily risen as we've develop and deploy new models, and additional integrations are made by tool developers and researchers. Currently, ORES supports 78 different models and 37 different language-specific wikis.

Generally, we see 50 to 125 requests per minute from external tools that are using ORES' predictions (excluding the MediaWiki extension that is more difficult to track). Sometimes these external requests will burst up to 400-500 requests per second. Figure 2a shows the periodic and bursty nature of scoring requests received by the ORES service. Note that every day at about 11:40 UTC, the request rate jumps—most likely a batch scoring job such as a bot.

Figure 2b shows our rate of precaching requests coming from our own systems. This graph roughly reflects the rate of edits that are happening to all of the wikis that we support since we'll start a scoring job for nearly every edit as it happens. Note that the number of precaching requests is about an order of magnitude higher than our known external score request rate. This is expected, since Wikipedia editors and the tools they use will not request a score for every single revision. This is a computational price we pay to attain a high cache hit rate and to ensure that our users get the quickest possible response for the scores that they *do* need.

Taken together these strategies allow us to optimize the real-time quality control workflows and batch processing jobs of Wikipedians and their tools. Without serious effort to make

<sup>16</sup>As demonstrated by <https://quarry.wmflabs.org/query/14033>



(a) External requests per minute with a 4 hour block broken out to highlight a sudden burst of requests.



(b) Precaching requests per minute

Fig. 2. Request rates to the ORES service for the week ending on April 13th, 2018

sure that ORES is practically fast and highly available to real-time use cases, ORES would become irrelevant to the target audience and thus irrelevant as a work-support infrastructure. By engineering a system that conforms to the work-process needs of Wikipedians and their tools, we've built and systems intervention that has the potential gain wide adoption in Wikipedia's technical ecology.

## 5 INNOVATIONS IN OPENNESS

Our goals in the development of ORES and the deployment of models is to keep the process—the flow of data from random samples through model training and evaluation—open for

review, critique, and iteration. In this section, we'll describe how we implemented transparent reproducibility in our model development process, and how ORES outputs a wealth of useful and nuanced information for users. By making this detailed information available to users and developers, we hope to enable flexibility and power in the evaluation and use of ORES predictions for novel purposes. In this section, we describe some of the key, novel innovations that have made ORES fit Wikipedian concerns and be flexible to re-use. The appendix also contains information about ORES' detailed prediction output (section A.1.1), how users and tools can adjust their use to model fitness (section A.1.2, and how the whole model development workflow is made inspectable and replicable (section A.1.3).

## 5.1 Collaboratively labeled data

There are two primary strategies for gathering labeled data for ORES' models: found traces and manual labels.

**Found traces.** For many models, there are already a rich set of digital traces that can be assumed to reflect a useful human judgement. For example, in Wikipedia, it's very common that damaging edits will be reverted and that good edits will not be reverted. Thus the revert action (and remaining traces) can be used to assume that the reverted edit is damaging. We have developed a re-usable script<sup>17</sup> that when given a sample of edits, will label the edits as "reverted\_for\_damage" or not based on a set of constraints: edit was reverted within 48 hours, the reverting editor was not the same person, and the edit was not restored by another editor.

However, this "reverted\_for\_damage" label is problematic in that many edits are reverted not because they are damaging but because they are involved in some content dispute. Also, the label does not differentiate damage that is a good-faith mistake from damage that is intentional vandalism. So in the case of damage prediction models, we'll only make use of the "reverted\_for\_damage" label when manually labeled data is not available.

Another case of found traces is article quality assessments—named "wp10" after the Wikipedia 1.0 assessment where the article quality assessment scale originated<sup>18</sup>. We follow the process developed by Warncke-Wang et al.[43] to extract the revision of an article that was current at the time of an assessment. Many other wikis employ a similar process of article quality labeling (e.g. French Wikipedia and Russian Wikipedia), so we can use the same script to extract their assessments with some localization<sup>19</sup>. However other wikis either do not apply the same labeling scheme consistently or at all and manual labeling is our only option.

**Manual labeling.** We hold manual labeling as the gold standard for purposes of training a model to replicate human judgement. By asking Wikipedians to demonstrate their judgement on examples from their own wikis, we can most closely tailor model predictions to match the judgements that make sense to these communities. This contrasts with found data that is much easier to come by as it is available but the implicit signals may not exactly match the intended use of the model. Manual labeling has a high up-front expense of human labor. In order to efficiently utilize valuable time investments by our collaborators, mostly volunteer Wikipedians, we've developed a system called "Wiki Labels"<sup>20</sup>. Wiki Labels

<sup>17</sup>see *autolabel* in <https://github.com/wiki-ai/editquality>

<sup>18</sup><http://enwp.org/WP:WP10>

<sup>19</sup>see *extract\_labelings* in <https://github.com/wiki-ai/articlequality>

<sup>20</sup><http://enwp.org/:m:Wikilabels>

allows Wikipedians to submit judgments of specific random samples of Wiki content using a convenient interface and logging in via their Wikipedia account.

For example, to supplement our models of edit quality, we replace the models based on found “reverted\_for\_damage” traces with manual judgments where we specifically ask labelers to distinguish “damaging”/good from “good-faith”/vandalism. Using these labels we can build two separate models which allow users to filter for edits that are likely to be good-faith mistakes[14], to just focus on vandalism, or to apply themselves broadly to all damaging edits.

## 5.2 Threshold optimization

When we first started developing ORES, we realized that operational concerns of Wikipedia’s curators need to be translated into confidence thresholds for the prediction models. For example, counter-vandalism patrollers seek catch all (or almost all) vandalism before it is allowed to stick in Wikipedia for very long. That means they have an operational concern around the *recall* of a damage prediction model. They’d also like to review as few edits as possible in order to catch that vandalism. So they have an operational concern around the *filter rate*—the proportion of edits that are not flagged for review by the model[13].

By finding the threshold of prediction likelihood that optimizes the filter-rate at a high level of recall, we can provide vandal-fighters with an effective trade-off for supporting their work. We refer to these optimizations in ORES as *threshold optimizations* and ORES provides information about these thresholds in a machine-readable format so that tools can automatically detect the relevant thresholds for their wiki/model context.

Originally, when we developed ORES, we defined these threshold optimizations in our deployment configuration. But eventually, it became apparent that our users wanted to be able to search through fitness metrics to choose thresholds that matched their own operational concerns. Adding new optimizations and redeploying quickly became a burden on us and a delay for our users. So we developed a syntax for requesting an optimization from ORES in realtime using fitness statistics from the models tests. E.g. `maximum recall @ precision >= 0.9` gets a useful threshold for a counter-vandalism bot or `maximum filter_rate @ recall >= 0.75` gets a useful threshold for semi-automated edit review (with human judgement).

```
{
  "threshold": 0.30, ...,
  "filter_rate": 0.88, "fpr": 0.097, "precision": 0.21, "recall": 0.75}

```

---

Fig. 3. Result of [https://ores.wikimedia.org/v3/scores/enwiki/?models=damaging&model\\_info=statistics.thresholds.true.'maximumfilter\\_rate@recall'=0.75](https://ores.wikimedia.org/v3/scores/enwiki/?models=damaging&model_info=statistics.thresholds.true.'maximumfilter_rate@recall'=0.75)

This result shows that, when a threshold is set on 0.299 likelihood of damaging=true, then you can expect to get a recall of 0.751, precision of 0.215, and a filter-rate of 0.88. While the precision is low, this threshold reduces the overall workload of vandal-fighters by 88% while still catching 75% of (the most egregious) damaging edits.

## 5.3 Dependency injection and interrogability

From a technical perspective, ORES is a algorithmic “scorer” container system. It’s primarily designed for hosting machine classifiers, but it is suited to other scoring paradigms as well. E.g. at one point, our experimental installation of ORES hosted a Flesch-Kincaid readability

scorer. The only real constraints are that the scorer must express its inputs in terms of “dependencies” that ORES knows how to solve and the scores must be presentable in JSON (Javascript Object Notation).

**5.3.1 Dependency injection.** One of the key features of ORES that allows scores to be generated in an efficient and flexible way is a dependency injection framework.

**Efficiency.** For example, there are several features that go into the “damaging” prediction model that are drawn from the *diff* of two versions of an articles text (words added, words removed, badwords added, etc.) Because all of these features “depend” on the *diff*, the dependency solver can make sure that only one *diff* is generated and that all subsequent features make use of that shared data.

ORES can serve multiple scores in the same request. E.g. a user might want to gather a set of scores: “edit type”, “damaging”, and “good-faith”. Again, all of these prediction models depend on features related to the edit *diff*. ORES can safely combine all of the features required for each of the requested models and extract them with their shared dependencies together. Given that feature extraction tends to take the majority of time in a real-time request request (partially for fetching data [IO], partially for computation time [CPU]), this allows the scoring for multiple, similar models to take roughly the same amount of time as scoring a single model.

**Flexibility.** When developing a model, it’s common to experiment with different feature sets. A natural progression in the life of a model in the wild involves the slow addition of new features that prove useful during experimentation. By implementing a dependency system and a dependency solver, a model can communicate to ORES which features it needs and ORES can provide those features. At no point does a model developer need to teach ORES how to gather new data. All of the information needed for the solver to solve any given feature set is wrapped up in the dependencies.

**5.3.2 Interrogability.** The flexibility provided by the dependency injection framework let us implement a novel strategy for exploring *how* ORES’ models make predictions. By exposing the features extracted to ORES users and allowing them to inject their own features, we can allow users to ask how predictions would change if the world were different. Let’s say you wanted to explore how ORES judges unregistered (anon) editors differently from registered editors. Figure 4 demonstrates two prediction requests to ORES.

Figure 4a shows that ORES’ “damaging” model concludes that the edit identified by the *revision ID* of 34234210 is not damaging with 93.9% confidence. We can ask ORES to make a prediction about the exact same edit, but to assume that the editor was unregistered (anon). Figure 4b shows the prediction if edit were saved by an anonymous editor. ORES would still conclude that the edit was not damaging, but with less confidence (91.2%). By following a pattern like this for a single edit or a set of edits, we can get to know how ORES prediction models account for anonymity.

This is a very powerful tool for examining the potential biases in judgement. Imagine being able to ask a law enforcement officer if they feel like they have probable cause for a search and then asking again how their answer would change if the suspect were black.

Interrogability isn’t only useful for checking to see where ORES’ biases originate and what effects they have on predictions. For example, Ross has started using our article quality prediction models (*wp10*) as a method for suggesting work to new editors<sup>21</sup>. By asking

<sup>21</sup><https://dashboard-testing.wikiedu.org>



```

"damaging": {
  "score": {
    "prediction": false,
    "probability": {
      "false": 0.938910157824447,
      "true": 0.06108984217555305
    }
  }
}

```

(a) Prediction with anon = false injected

```

"damaging": {
  "score": {
    "prediction": false,
    "probability": {
      "false": 0.9124151990561908,
      "true": 0.0875848009438092
    }
  }
}

```

(b) Prediction with anon = true injected

Fig. 4. Two “damaging” predictions about the same edit are listed for ORES. In one case, ORES is asked to make a prediction assuming the editor is unregistered (anon) and in the other, ORES is asked to assume the editor is registered.

ORES to score a student’s draft and then asking ORES to reconsider the predicted quality level of the article with *one more header*, *one more image*, or *one more citation*, he’s built an intelligent user interface that can exposes the internal structure of a model in order to recommend the most productive development to the article—the change that will most likely bring it to a higher quality level.

## 6 ADOPTION PATTERNS

When we designed and developed ORES, we were targeting a specific problem—expanding the set values applied to the design of quality control tools to include recent a recent understanding of the importance of newcomer socialization. We don’t have any direct control of how developers chose to use ORES. We hypothesize that, by making edit quality predictions available to all developers, we would lower the barrier to experimentation in this space. It’s clear that we lowered barriers to experimentation. After we deployed ORES, we implemented some basic tools to showcase ORES, but we observed a steady adoption of our various prediction models by external developers in current tools and through the development of new tools<sup>22</sup>.

### 6.1 Showcase tools

In order to showcase the utility of ORES, we developed a two simple tools to surface ORES predictions within MediaWiki—the wiki that powers Wikipedia: *ScoredRevisions* and the *ORES Review Tool*.

**ScoredRevisions**<sup>23</sup> is a javascript-based “gadget” that runs on top of MediaWiki. When certain pages load in the MediaWiki interface (E.g. Special:RecentChanges, Special:Watchlist, etc.), the ScoredRevisions submits requests to the ORES service to score the edits present on the page. The javascript then updates the page with highlighting based on ORES predictions. Edits that are likely to be “damaging” are highlighted in red. Edits that might be damaging and are worth reviewing are highlighted in yellow. Other edits are left with the default background.

<sup>22</sup>See complete list: <http://enwp.org/:mw:ORES/Applications>

<sup>23</sup><https://github.com/he7d3r/mw-gadget-ScoredRevisions>

While this interface was excellent for displaying ORES potential, it had limited utility. First, it was severely limited by the performance of the ORES system. While ORES is reasonably fast for scoring a single edit, scoring 50-500 edits (the ranges that commonly appear on these pages) can take 30 seconds to 2 minutes. So a user is left waiting for the highlighting to appear. Because *ScoredRevisions* is only able to score edits after they are rendered, there was no way for a user to ask the system to filter edits ahead of time to only show edits that are likely to be damaging. The user needed to visually filter the long lists based on highlighted rows.

**The ORES Review Tool**<sup>24</sup> is a MediaWiki extension implemented in PHP. It uses an offline process to score all recent edits to Wikipedia and to store those scores in a table for querying and quick access. This tool implemented similar functionality to *ScoredRevisions* but because it had pre-cached ORES scores in a table, it rendered highlights for likely damaging edits as soon as the page loaded. It enabled users to filter based on likely damaging edits.

We released the ORES Review Tool as a “beta feature” on Wikimedia wikis where we were able to develop advanced edit quality models. The response was extremely positive. Over 26k editors in Wikipedia had manually enabled the ORES Review Tool by April of 2017. For reference, the total number of active editors across all languages of Wikipedia varies around 70k<sup>25</sup>, so this means that about a 3rd of active editors consciously chose to enable the feature.

## 6.2 Adoption in current tools

Many tools for counter-vandalism in Wikipedia were already available when we developed ORES. Some of them made use of machine prediction (e.g. Huggle<sup>26</sup>, STiki, ClueBot NG), but most did not. Soon after we deployed ORES, many developers that had not previously included their own prediction models in their tools were quick to adopt ORES. For example, *RealTime Recent Changes*<sup>27</sup> includes ORES predictions along-side their realtime interface and *FastButtons*<sup>28</sup>, a Portuguese Wikipedia gadget, began displaying ORES predictions next to their buttons for quick reviewing and reverting damaging edits.

Other tools that were not targeted at counter-vandalism also found ORES predictions—specific that of *article quality* (wp10)—useful. For example, *RATER*<sup>29</sup>, a gadget for supporting the assessment of article quality began to include ORES predictions to help their users assess the quality of articles and *SuggestBot*<sup>30</sup>[3], a robot for suggesting articles to an editor, began including ORES predictions in their tables of recommendations.

## 6.3 Development of new tools

Many new tools have been developed since ORES has released that may not have been developed at all otherwise. For example, the Wikimedia Foundation product department developed a complete redesign on MediaWiki’s *Special:RecentChanges* interface that implements a set of powerful filters and highlighting. They took the ORES Review Tool to it’s

<sup>24</sup>[http://enwp.org/:mw:ORES\\_review\\_tool](http://enwp.org/:mw:ORES_review_tool)

<sup>25</sup><https://stats.wikimedia.org/EN/TablesWikipediansEditsGt5.htm>

<sup>26</sup>Notably, Huggle adopted ORES prediction models soon after we deployed

<sup>27</sup><http://enwp.org/:m:RTRC>

<sup>28</sup><http://enwp.org/:pt:Wikipedia:Scripts/FastButtons>

<sup>29</sup><http://enwp.org/::en:WP:RATER>

<sup>30</sup><http://enwp.org/User:SuggestBot>

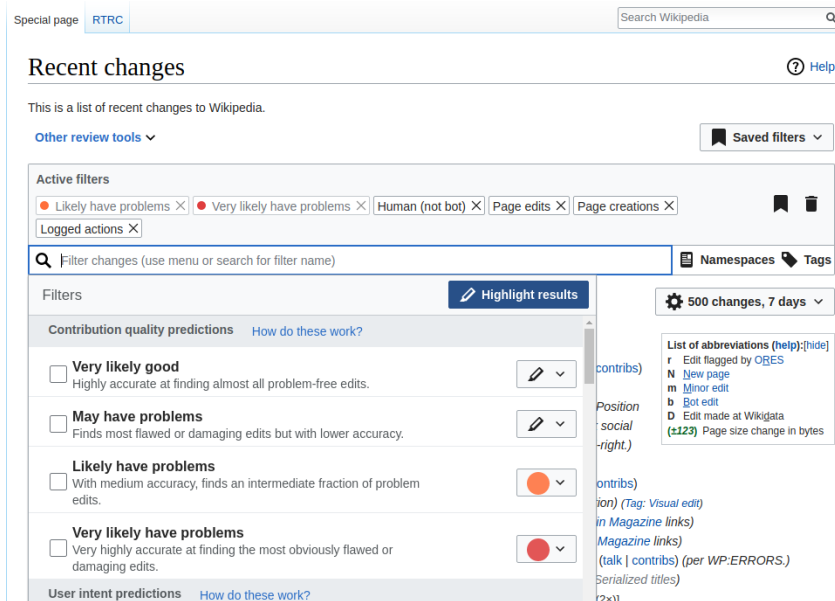


Fig. 5. A screenshot of the Edit Review Filters interface with ORES score-based filters

logical conclusion with an initiative that they referred to as Edit Review Filters<sup>31</sup>. In this interface, ORES scores are prominently featured at the top of the list of available features, and they have been highlighted as one of the main benefits of the new interface to the editing community.

When we first developed ORES, English Wikipedia was the only wiki that we are aware of that had a robot that used machine prediction to automatically revert obvious vandalism[2]. After we deployed ORES, several wikis developed bots of their own to use ORES predictions to automatically revert vandalism. For example, in PatruBOT in Spanish Wikipedia<sup>32</sup> and Dexbot in Persian Wikipedia<sup>33</sup> now automatically revert edits that ORES predicts are damaging with high confidence. These bots have been received with mixed acceptance. Because of the lack of human oversight, concerns were raised about PatruBOT's false positive rate but after consulting with the developer, we were able to help them find an acceptable threshold of confidence for auto-reverts.

One of the most noteworthy new tools is the suite of tools developed by Sage Ross to support the Wiki Education Foundation's<sup>34</sup> activities. Their organization supports classroom activities that involve editing Wikipedia. They develop tools and dashboards that help students contribute successfully and to help teachers monitor their students' work. Ross has recently published about how they interpret meaning from ORES' article quality models[36] and has integrate this prediction into their tools and dashboards to recommend work that students need to do to bring their articles up to Wikipedia's standards. See our discussion of interrogation in Section 5.

<sup>31</sup>[http://enwp.org/:mw:Edit\\_Review\\_Improvements](http://enwp.org/:mw:Edit_Review_Improvements)

<sup>32</sup><https://es.wikipedia.org/wiki/Usuario:PatruBOT>

<sup>33</sup><https://fa.wikipedia.org/wiki/User:Dexbot>

<sup>34</sup><https://wikiedu.org/>

## 7 CASE STUDIES IN REFLECTION

When we first deployed ORES, we reached out to several different wiki-communities and invited them to test out the system for use in patrolling for vandalism. In these announcements, we encouraged editors to install ScoredRevisions, the only tool that used made use of ORES' edit quality models at the time. ScoredRevisions both highlights edits that are likely to be damaging (as predicted by the model) and displays the likelihood of the prediction as a percentage.

Before long, our users began filing false-positive reports on wiki pages of their own design. In this section we will describe three cases where our users independently developed these false-positive reporting pages and how they used them to understand ORES, the roles of automated quality control in their own spaces, and to communicate with us.

### 7.1 Report mistakes (Wikidata)

Improvements [\[ edit \]](#)

Diff id ↕	Damaging ↕	Old score ↕	Score1 ↕	Score2 ↕	Score3 ↕	1st improv. ↕	2nd ↕	3rd ↕	Overall ↕
<a href="#">210649590</a>	No	91%	30%	5%	0%	+61%	+25%	+5%	+91%
<a href="#">237999679</a>	No	84%	71%	63%	60%	+13%	+8%	+3%	+24%
<a href="#">243937491</a>	No	95%	46%	74%	71%	+49%	-28%	+3%	+24%
<a href="#">251530750</a>	No	91%	55%	56%	55%	+36%	-1%	+1%	+36%
<a href="#">253584599</a>	No	99%	89%	78%	70%	+10%	+11%	+8%	+29%
<a href="#">257856652</a>	No	91%	30%	4%	1%	+61%	+26%	+3%	+90%

Fig. 6. A slice of the ORES report mistakes table in Wikidata.

When we first deployed prediction models for Wikidata, a free and open knowledge base that can be read and edited by both humans and machines<sup>35</sup>, we were breaking new ground by building a damage detection classifier based on a structured data wiki[38]. So we created a page called “Report mistakes” and invited users to tell us about mistakes that the prediction model made on that page. We left the format and structure largely up to the users.

Within 20 minutes, we received our first report from User:Mbch that ORES was reporting edits that couldn't possibly be vandalism as potentially damaging. As reports streamed in, we began to respond to them and make adjustments to the model building process to address data extraction bugs and to increase the signal so that the model differentiate damage from non-damaging edits. After a month of reports and bug fixes, we decided to build a table to represent the progress that we made in iterations on the model against the reported false-positives. See Figure 6 for a screenshot of the table. Each row represents a mis-classified edit, and each column describes the progress we made in not detecting those edits as damaging in future iterations of the model. Through this process, we learned how Wikidata editors saw damage and how our modeling and feature extraction process captured signals in ways that differed from Wikidata editors' understandings. We were also able to publicly demonstrate improvements to this community.

### 7.2 Patrolling/ORES (Italian Wikipedia)

Italian Wikipedia was one of the first wikis where we deployed basic edit quality models. Our local collaborator who helped us develop the language specific features, User:Rotpunkt,

<sup>35</sup><https://wikidata.org>

created a page for ORES<sup>36</sup> with a section for reporting false-positives (“falsi positivi”). Within several hours, Rotpunkt and a few other edits started to notice some trends in their false positive reports. First, Rotpunkt noticed that there were several counter-vandalism edits that ORES was flagging as potentially damaging, so he made a section for collecting that specific type of mistake (“annullamenti di vandalismo”). A few reports later and he added a section for “corrections to the verb for *have*” (“correzioni verbo avere”). Through this process, editors from Italian Wikipedia were essentially performing a grounded theory exploration of the general classes of errors that ORES was making.

Once there were several of these mistake-type sections and several reports within each section, Rotpunkt reached out to us to let us know what he had found. He explained to us (via our IRC channel) that many of ORES mistakes were understandable, but there were some general trends in mistakes around the Italian verb for *have*: “ha”. We knew immediately what was likely to be the issue. It turns out that “ha” in English and many other languages is laughing, an example of informal language that doesn’t belong in an encyclopedia article. While the word “ha” in Italian translates to “have” and is perfectly acceptable in articles.

Because of the work of Rotpunkt and his collaborators in Italian Wikipedia, we were able to recognize the source of this issue (a set of features intended to detect the use of *informal language* in articles) and to remove “ha” from that list for Italian Wikipedia. This is just one example of many issues we were able to address because of the grounded theory and thematic analysis performed by Italian Wikipedians.

### 7.3 PatruBOT (Spanish Wikipedia)

Soon after we released support for Spanish Wikipedia, User:jem developed a robot to automatically revert damaging edits using ORES predictions (PatruBOT). This robot was not running for long before our discussion pages started to be bombarded with confused Spanish-speaking editors asking us questions about why ORES did not like their work. We struggled to understand the origin of the complaints until someone reached out to us to tell us about PatruBOT and its activities.

We haven’t been able to find the source code for PatruBot, but from what we’ve been able to gather looking at its activity, it appears to us that PatruBOT was too sensitive and was likely reverting edits that ORES did not have enough confidence about. Generally, when running an automated counter-vandalism bot, the most immediately operational concern is around *precision* (the proportion of positive predictions that are true-positives). This is because mistakes are extra expensive when there’s no human judgement between a prediction and a revert (rejection of the contribution). The proportion of all damaging edits that are actually caught by the bot (*recall*) is a secondary concern to be optimized.

We recommend that bot developers who are interested in running an automated counter-vandalism bot use a threshold that maximizes recall at high precision (90% is a good starting point). According to our queries<sup>37</sup>, the Spanish Wikipedia damaging model can be expected to have 90% precision and catch 17% of damage if the bot only reverted edits where the likelihood estimate is above 0.959.

We reached out to the bot developer to try to help, but given the voluntary nature of their work, they were not available to discuss the issue with us. Eventually, other editors who

<sup>36</sup><https://it.wikipedia.org/wiki/Progetto:Patrolling/ORES>

<sup>37</sup>[https://ores.wikimedia.org/v3/scores/eswiki?models=damaging&model\\_info=statistics.thresholds.true.%27maximum%20recall%20@%20precision%20%3E=%200.9%27](https://ores.wikimedia.org/v3/scores/eswiki?models=damaging&model_info=statistics.thresholds.true.%27maximum%20recall%20@%20precision%20%3E=%200.9%27)

were concerned with PatruBOT's behavior organized an informal crowdsourced evaluation of the fitness of PatruBOT's behavior<sup>38</sup> where they randomly sampled 1000 reverts performed by PatruBOT and reviewed their appropriateness. At the time of writing, PatruBOT has been stopped<sup>39</sup> and the informal evaluation is ongoing.

#### 7.4 Bias against anonymous editors

Shortly after we deployed ORES, we started receiving reports that ORES' damage detection models were overly biased against anonymous editors. At the time, we were using Linear SVM<sup>40</sup> estimators to build classifiers, and we were considering making the transition towards ensemble strategies like GradientBoosting and RandomForest estimators<sup>41</sup>. We took the opportunity to look for bias in the error of estimation between anonymous editors, newly registered editors. By using our feature injection/interrogation strategy (described in Section 5), we could ask our current prediction models how they would change their predictions if the exact same edit were made by a different editor.

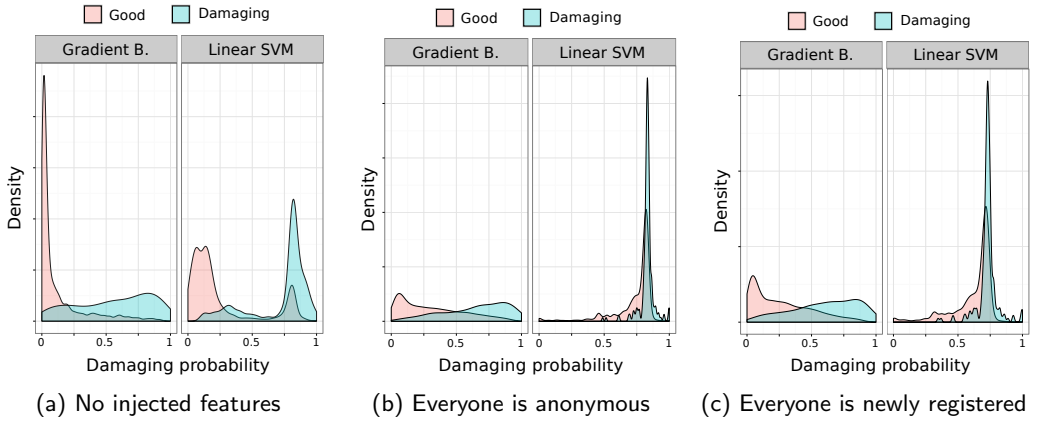


Fig. 7. The distributions of the probability of a single edit being scored as “damaging” based on injected features for the target user-class is presented. Note that when injecting user-class features (anon, newcomer), all other features are held constant.

Figure 7 shows the probability density of the likelihood of “damaging” given three different passes over the exact same test set using two of our modeling strategies. Figure 7a shows that, when we leave the features to their natural values, it appears that both models are able to differentiate effectively between damaging edits (high damaging probability) and non-damaging edits (low damaging probability) with the odd exception of a large amount of non-damaging edits with a relatively high damaging probability around 0.8 in the case of the Linear SVM model. Figures 7b and 7c show a stark difference. For the scores that go into these plots, characteristics of anonymous editors and newly registered editors were injected for all of the test edits. We can see that the GradientBoosting model can still differentiate

<sup>38</sup>[https://es.wikipedia.org/wiki/Wikipedia:Mantenimiento/Revisi%C3%B3n\\_de\\_errores\\_de\\_PatruBOT%2FAn%C3%A1lisis](https://es.wikipedia.org/wiki/Wikipedia:Mantenimiento/Revisi%C3%B3n_de_errores_de_PatruBOT%2FAn%C3%A1lisis)

<sup>39</sup>[https://es.wikipedia.org/wiki/Wikipedia:Caf%C3%A9%2FArchivo%2FMiscel%C3%A1nea%2FActual#Parada\\_de\\_PatruBOT](https://es.wikipedia.org/wiki/Wikipedia:Caf%C3%A9%2FArchivo%2FMiscel%C3%A1nea%2FActual#Parada_de_PatruBOT)

<sup>40</sup><http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>

<sup>41</sup><http://scikit-learn.org/stable/modules/ensemble.html>



damage from non-damage while the Linear SVM model flags nearly all edits as damage in both case.

Through the reporting of this issue and our subsequent analysis, we were able to identify the issue and show that an improvement to our modeling strategy mitigates the problem. Without such a tight feedback loop, we most likely wouldn't have noticed how poorly ORES' damage detection models were performing in practice. Worse, it might have caused vandal-fighters to be increasingly (and inappropriately) skeptical of contributions by anonymous editors and newly registered editors—two groups of contributors that are already met with unnecessary hostility<sup>42</sup>[16].

## 7.5 Discussion

These case studies in responses to ORES provide a window into how our team has been able to work with the locals in various communities to refine our understandings of their needs, into methods for recognizing and addressing biases in ORES' models, and into how people think about what types of automation they find acceptable in their *spaces*.

**Refining our understandings and iterating our models.** The information divide between us researchers/engineers and those members of a community is often wider than we realize. Through iteration with the Wikidata and Italian models, we learned about incorrect assumptions we would make about how edits happen (e.g. client edits in Wikidata) and how language works (e.g. “ha” is not laughing in Italian). It's likely we would never be able to fully understand the context in which damage detection models should operate before deploying the models. But these case studies demonstrate how, with a tight communication loop, many surprising and wrong assumptions that were baked into our modeling process could be identified and addressed quickly. It seems that many of the relevant issues in feature engineering and model tuning become apparent when the model is used in context to try to address a real problem (in these cases, counter-vandalism).

**Methods for recognizing and addressing bias.** The Italian Wikipedians showed us something surprising and interesting about collaborative evaluation of machine prediction: thematic analysis is very powerful. Through the collection of ORES mistakes and iteration, our Italian collaborators helped us understand general trends in the types of mistakes that ORES made. It strikes us that this a somewhat general strategy for bias detection. While our users certainly brought their own biases to their audit of ORES, they were quick to discover and come to consensus about trends in ORES' issues. Before they had performed this process and shared their results with us, we had no idea that any issues was present. After all, the fitness statistics for the damage detection model looked pretty good—probably good enough to publish a research paper! Their use of thematic analysis seems to like a powerful tool that developers will want to make sure is well supported in any crowd based auditing support technologies.

**How people think about “acceptable” automation.** In our case study, Spanish Wikipedians are in the processes of coming to agreements about what roles are acceptable for automated agents. Through observation of PatruBOT's behavior, they have decided that the *false discovery rate* (i.e.,  $1 - \text{precision}$ ) was too high by watching the bot work in practice and they started their own independent analysis to find quantitative, objective answers about what the real rate is. Eventually they may come to a conclusion about an

<sup>42</sup>[http://enwp.org/:en:Wikipedia:IPs\\_are\\_human\\_too](http://enwp.org/:en:Wikipedia:IPs_are_human_too)

acceptable *false discovery rate* or they may decide that no revert is acceptable without human intervention.

## 8 CONCLUSION AND FUTURE WORK

Designing for empowerment leads us in new directions. Rather than running an experiment on an “intervention” (e.g. [17]), we’re reducing barriers and encouraging a stalled “conversation” continue. We see this as a “hearing to speech” ala Nell Morten[31] in contrast to “speaking to be heard”. By building the exact technology or process we think is right, we would be “speaking to be heard” and forcing our own views into the technological conversation about how quality is enacted in Wikipedia. By stepping back and asking, “What is preventing others from getting involved in this conversation?” and lowering those barriers, we run the risk that the conversation will not go in the directions that we value. It is a conscious choice we make to attempt to empower others rather than to assert ourselves, but we do not make this choice purely out of altruism. It is impractical for one individual or small group to drive a conversation in Wikipedia in a productive direction. Halfaker et al.’s Snuggle intervention[17] tried that strategy with limited success in past work. We think it’s time to hear to speech and see what others would like to contribute from their own standpoints.

When considering our design rationale, ORES is not quite a successor system. It doesn’t directly enact an alternative vision of how quality control should function in Wikipedia. It is a system for making the construction of successor technologies easier. By solving for efficiency and letting others design for process, a new set of expanded standpoints can more easily be expressed. Under another view, ORES is maybe a successor in that it enacts an alternative view of how algorithms should be made available to the populations they govern. By keeping algorithms integrated into specific tools and keeping the operational details of the algorithm hidden, past developers enacted values of control and focused use of their creations. By opening ORES as an algorithmic system, encouraging experimentation, and providing tools for interrogation of the algorithms themselves, we enact a set of values that embrace experimentation and the progress of mediation.

We do intend to help Wikipedia’s social process form in ways that align with our values—the more complete balance of efficient quality control and newcomer support. We want to see Wikipedia work better. By “better” we mean that more people who want to contribute will feel welcome to do so and will find their place in Wikipedia. In this paper we provide no direct evaluation of newcomer retention, nor do we look for evidence of improved newcomer socialization. Instead, we’re targeting the early precursors of social change: ecological health and critical reflection.

### 8.1 Ecological health

In section 6, we show clear evidence of a boom in the ecosystem of quality control technologies. Through the adoption of ORES, new tools have quickly gained relevance. Many of these tools re-mediate the way that quality control happens in Wikipedia. For example, consider Sage Ross’s use of ORES’ interrogability to support newcomers in developing articles that Wikipedians would find acceptable that was discussed in section 6. By using a quality model to direct newcomers behavior, socialization may be improved at no cost to efficiency—maybe even an improvement from a reviewer’s point of view since the system would result in more structurally complete drafts. This addition to the technological conversation around quality control represents a substantial shift from the focus on the post-publication boundary[10] to a focus on pre-review training for newcomers. While Sage’s interface has yet to be widely

Reference  
ecological  
health  
from  
design  
rationale  
here

adopted, it represents a key example of a new concept in the technological conversation in Wikipedia.

While ORES has been live for 3 years at the time of writing, the ecological intervention is still young. "ORES" has only recently become commonly known by Wikimedia contributors and we're still building out basic support for the different Wikimedia language/project communities. In order to see a fundamental shift in the Wikimedia's technical ecology, we'll likely need to continue observing for 1-2 more years still. But the early signs of ecological health (heavy adoption and increased diversity of quality management technologies) are what we hope to see—long term and continual re-mediation of quality and community management policies, processes, and technologies.

## 8.2 Critical reflection

In section 7, we show evidence of critical reflection on the current processes and the role of algorithms in quality control. We believe that the case studies that we describe both show that this reflection is taking place and that the wide proliferation of tools that provide surprising alternative uses of ORES suggest that Wikipedians feel a renewed power over their quality control processes. Even if we are wrong about the direction of change that Wikipedia needs for quality control and newcomer socialization, we're inspired by much of the concern that has surfaced for looking into biases in ORES' prediction models (e.g. anons and the Italian "ha") and over what role algorithms should have in directly reverting human actions (e.g. PatruBOT and DexBot). Because of ORES' open API and Wikipedia's collaborative wiki pages, it was relatively straightforward for Wikipedians to audit ORES' predictions and collaborate with each other to build information about trends in ORES' mistakes and how they expected their own processes to function.

## 8.3 Future work

One of the clear lines of future work that observing ORES in the world drives us toward is improved crowd-based auditing tools. As our case studies suggest, auditing of ORES' predictions and mistakes has become a very popular activity. Despite the difficulty for a user of finding a deep wiki page and using templates to flag false positives, Wikipedians have managed to organize several similar processes for flagging false positives and calling them to our attention. To better facilitate this process, future system builders should implement structured means to refute, support, discuss, and critique the predictions of machine models. With a structured way to report false positives and other real-human judgments, we can make it easy for tools that use ORES to also allow for reporting mistakes. We can also make it easier to query a database of ORES mistakes in order to build the kind of thematic analyses that Italian Wikipedians showed us. By supporting such an activity, we are working to transfer more power from ourselves and to our users. Should one of our models develop a nasty bias, our users will be more empowered to coordinate with each other, show that the bias exists and where it causes problems, and either get the model's predictions turned off or even shut down ORES.

We look forward to what those who work in the space of critical algorithm studies will do with ORES. As of writing, most of the studies and critiques of *subjective algorithms*[42] focus on large for-profit organizations like Google and Facebook—organizations that can't afford to open up their proprietary algorithms due to competition. Wikipedia is one of the largest and most important information resources in the world. The algorithms that ORES makes available are part of the decision process for allowing some people to contribute

Consider discussing the take-aways to system-builders and critical algorithms folks

and preventing other. This is a context where *algorithms matter to humanity*, and we are openly experimenting with the kind of transparent and open processes that the *fairness and transparency* researchers are advocating. Yet we have new problems and new opportunities. There is a large body of work exploring how biases manifest and how unfairness can play out in algorithmically mediated social contexts. ORES would be an excellent place to expand the literature within a real and important field site.

Finally, we also see potential in allowing Wikipedians, the denizens of Wikipedia, to freely train, test, and use their own prediction models. Currently, ORES is only suited to deploy models that are developed by someone with a strong modeling and programming background. That doesn't need to be the case. We have been experimenting with demonstrating ORES model building processes using Jupyter Notebooks<sup>4344</sup> and have found that beginning programmers can understand the work involved. This is still not the holy grail of crowd developed machine prediction—where all of the incidental complexities involved in programming are removed from the process of model development and evaluation. Future work exploring strategies for allowing end-users to build models that are deployed by ORES would surface the relevant HCI issues involved and the changes the technological conversations that such a margin-opening intervention might provide.

## 9 ACKNOWLEDGEMENTS

REDACTED FOR REVIEW

## REFERENCES

- [1] B Thomas Adler, Luca De Alfaro, Santiago M Mola-Velasco, Paolo Rosso, and Andrew G West. 2011. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 277–288.
- [2] Jacobi Carter. 2008. ClueBot and vandalism on Wikipedia. <http://www.acm.uiuc.edu/~carter11/ClueBot.pdf>
- [3] 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 the 12th international conference on Intelligent user interfaces*. ACM, 32–41.
- [4] Kate Crawford. 2016. Can an algorithm be agonistic? Ten scenes from life in calculated publics. *Science, Technology, & Human Values* 41, 1 (2016), 77–92.
- [5] Quang-Vinh Dang and Claudia-Lavinia Ignat. 2016. Quality assessment of wikipedia articles without feature engineering. In *Digital Libraries (JCDL), 2016 IEEE/ACM Joint Conference on*. IEEE, 27–30.
- [6] Nicholas Diakopoulos. 2015. Algorithmic accountability: Journalistic investigation of computational power structures. *Digital Journalism* 3, 3 (2015), 398–415.
- [7] R Stuart Geiger. 2011. The lives of bots. (2011), 78–79.
- [8] R Stuart Geiger. 2014. Successor Systems: The Role of Reflexive Algorithms in Enacting Ideological Critique. *AoIR Selected Papers of Internet Research* 4 (2014).
- [9] R Stuart Geiger and Aaron Halfaker. 2013. When the levee breaks: without bots, what happens to Wikipedia's quality control processes?. In *Proceedings of the 9th International Symposium on Open Collaboration*. ACM, 6.
- [10] R Stuart Geiger, Aaron Halfaker, Maryana Pinchuk, and Steven Walling. 2012. Defense Mechanism or Socialization Tactic? Improving. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*.
- [11] R Stuart Geiger and David Ribes. 2010. The work of sustaining order in wikipedia: the banning of a vandal. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM, 117–126.

<sup>43</sup><http://jupyter.org>

<sup>44</sup>e.g. [https://github.com/wiki-ai/editquality/blob/master/ipynb/reverted\\_detection\\_demo.ipynb](https://github.com/wiki-ai/editquality/blob/master/ipynb/reverted_detection_demo.ipynb)

- [12] Tarleton Gillespie. 2014. The relevance of algorithms. *Media technologies: Essays on communication, materiality, and society* 167 (2014).
- [13] Aaron Halfaker. 2016. Notes on writing a Vandalism Detection paper. <http://socio-technologist.blogspot.com/2016/01/notes-on-writing-wikipedia-vandalism.html>
- [14] Aaron Halfaker. 2017. Automated classification of edit quality (worklog, 2017-05-04). [https://meta.wikimedia.org/wiki/Research\\_talk:Automated\\_classification\\_of\\_edit\\_quality/Work\\_log/2017-05-04](https://meta.wikimedia.org/wiki/Research_talk:Automated_classification_of_edit_quality/Work_log/2017-05-04)
- [15] Aaron Halfaker. 2017. Interpolating Quality Dynamics in Wikipedia and Demonstrating the Keilana Effect. In *Proceedings of the 13th International Symposium on Open Collaboration*. ACM, 19.
- [16] Aaron Halfaker, R Stuart Geiger, Jonathan T Morgan, and John Riedl. 2013. The rise and decline of an open collaboration system: How Wikipedias reaction to popularity is causing its decline. *American Behavioral Scientist* 57, 5 (2013), 664–688.
- [17] Aaron Halfaker, R Stuart Geiger, and Loren G Terveen. 2014. Snuggle: Designing for efficient socialization and ideological critique. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 311–320.
- [18] Aaron Halfaker and John Riedl. 2012. Bots and cyborgs: Wikipedia’s immune system. *Computer* 45, 3 (2012), 79–82.
- [19] 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/>
- [20] Donna Haraway. 1988. Situated knowledges: The science question in feminism and the privilege of partial perspective. *Feminist studies* 14, 3 (1988), 575–599.
- [21] Sandra G Harding. 1987. *Feminism and methodology: Social science issues*. Indiana University Press.
- [22] Benjamin Mako Hill and Aaron Shaw. 2014. Consider the redirect: A missing dimension of Wikipedia research. In *Proceedings of The International Symposium on Open Collaboration*. ACM, 28.
- [23] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, et al. 2003. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 17–24.
- [24] Joshua A Kroll, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. 2016. Accountable algorithms. *U. Pa. L. Rev.* 165 (2016), 633.
- [25] Lawrence Lessig. 1999. *Code: And other laws of cyberspace*. Basic Books.
- [26] Włodzimierz Lewoniewski, Krzysztof Wecel, and Witold Abramowicz. 2017. Relative Quality and Popularity Evaluation of Multilingual Wikipedia Articles. In *Informatics*, Vol. 4. Multidisciplinary Digital Publishing Institute, 43.
- [27] Jonathan T. Morgan. 2015. What we learned from the Inspire campaign to increase gender diversity on Wikimedia. <https://blog.wikimedia.org/2015/05/28/what-we-learned-from-the-inspire-campaign/>
- [28] Jonathan T Morgan, Siko Bouterse, Heather Walls, and Sarah Stierch. 2013. Tea and sympathy: crafting positive new user experiences on wikipedia. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 839–848.
- [29] Jonathan T. Morgan and Aaron Halfaker. 2018. Evaluating the impact of the Wikipedia Teahouse on newcomer socialization and retention. *in press* (2018).
- [30] Jonathan T Morgan and Mark Zachry. 2010. Negotiating with angry mastodons: the wikipedia policy environment as genre ecology. In *Proceedings of the 16th ACM international conference on Supporting group work*. ACM, 165–168.
- [31] Nelle Morton. 1985. *The Journey is Home*. Beacon Press.
- [32] Gabriel Mugar. 2017. Preserving the Margins: Supporting Creativity and Resistance on Digital Participatory Platforms. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (2017), 83.
- [33] Sneha Narayan, Jake Orlowitz, Jonathan T Morgan, and Aaron Shaw. 2015. Effects of a Wikipedia Orientation Game on New User Edits. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*. ACM, 263–266.
- [34] Martin Potthast, Benno Stein, and Robert Gerling. 2008. Automatic vandalism detection in Wikipedia. In *European conference on information retrieval*. Springer, 663–668.
- [35] Amira Rezguia and Kevin Crowston. 2017. Stigmergic Coordination in Wikipedia. (2017). <https://crowston.syr.edu/sites/crowston.syr.edu/files/Stigmergic%20coordination%20in%20wikipedia%20to%20distribute.pdf>

- [36] Sage Ross. 2016. Visualizing article history with Structural Completeness. <https://wikiedu.org/blog/2016/09/16/visualizing-article-history-with-structural-completeness/>
- [37] Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort. 2014. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry* (2014), 1–23.
- [38] Amir Sarabadani, Aaron Halfaker, and Dario Taraborelli. 2017. Building automated vandalism detection tools for Wikidata. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 1647–1654.
- [39] Clay Spinuzzi. 2003. *Tracing genres through organizations: A sociocultural approach to information design*. Vol. 1. Mit Press.
- [40] Clay Spinuzzi and Mark Zachry. 2000. Genre ecologies: An open-system approach to understanding and constructing documentation. *ACM Journal of Computer Documentation (JCD)* 24, 3 (2000), 169–181.
- [41] Loren Terveen, Joseph A Konstan, and Cliff Lampe. 2014. Study, Build, Repeat: Using Online Communities as a Research Platform. In *Ways of Knowing in HCI*. Springer, 95–117.
- [42] Zeynep Tufekci. 2015. Algorithms in our midst: Information, power and choice when software is everywhere. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 1918–1918.
- [43] Morten Warncke-Wang. 2017. English Wikipedia Quality Assessment Dataset. In *Fileset*. Figshare.
- [44] Morten Warncke-Wang, Dan Cosley, and John Riedl. 2013. Tell me more: an actionable quality model for Wikipedia. In *Proceedings of the 9th International Symposium on Open Collaboration*. ACM, 8.
- [45] Morten Warncke-Wang and Aaron Halfaker. 2017. Screening WikiProject Medicine articles for quality. [https://meta.wikimedia.org/wiki/Research:Screening\\_WikiProject\\_Medicine\\_articles\\_for\\_quality](https://meta.wikimedia.org/wiki/Research:Screening_WikiProject_Medicine_articles_for_quality)
- [46] Morten Warncke-Wang, Vivek Ranjan, Loren G Terveen, and Brent J Hecht. 2015. Misalignment Between Supply and Demand of Quality Content in Peer Production Communities.. In *ICWSM*. 493–502.
- [47] Andrew G West, Sampath Kannan, and Insup Lee. 2010. STiki: an anti-vandalism tool for Wikipedia using spatio-temporal analysis of revision metadata. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*. ACM, 32.
- [48] Shoshana Zuboff. 1988. *In the age of the smart machine: The future of work and power*. Vol. 186. Basic books New York.

## A APPENDIX

### A.1 ORES system details

In this section, we describe some of the details of the ORES system.

**A.1.1 Score documents.** The predictions made by through ORES are human- and machine-readable. In general, our classifiers will report a specific prediction along with a set of probability (likelihood) for each class. By providing detailed information about a prediction, we allow users to re-purpose the prediction for their on use. Consider article quality (wp10) prediction output in Figure 9.

```
"wp10": {
  "score": {
    "prediction": "Start",
    "probability": {
      "FA": 0.0032931301528326693, "GA": 0.005852955431273448,
      "B": 0.060623380484537165, "C": 0.01991363271632328,
      "Start": 0.7543301344435299, "Stub": 0.15598676677150375
    }
  }
}
```

---

Fig. 8. Result of <https://ores.wikimedia.org/v3/scores/enwiki/34234210/wp10>



A developer making use of a prediction like this may choose to present the raw prediction “Start” (one of the lower quality classes) to users or to implement some visualization of the probability distribution across predicted classes (75% Start, 16% Stub, etc.). They might even choose to build an aggregate metric that weights the quality classes by their prediction weight (e.g. Ross’s student support interface[36] or the *weighted sum* metric from [15]).

**A.1.2 Model information.** In order to use a model effectively in practice, a user needs to know what to expect from model performance. E.g. how often is it that when an edit is predicted to be “damaging” it actually is? (*precision*) or what proportion of damaging edits should I expect will be caught by the model? (*recall*) The target metric of an operational concern depends strongly on the intended use of the model. Given that our goal with ORES is to allow people to experiment with the use and reflection of prediction models in novel ways, we sought to build an general model information strategy.

```
"damaging": {
  "type": "GradientBoosting",
  "version": "0.4.0",
  "environment": {"machine": "x86_64", ...},
  "params": {"center": true, "init": null, "label_weights": {"true": 10},
    "labels": [true, false], "learning_rate": 0.01, "min_samples_leaf": 1,
    ...},
  "statistics": {
    "counts": {"labels": {"false": 18702, "true": 743},
      "n": 19445,
      "predictions": {"false": {"false": 17989, "true": 713},
        "true": {"false": 331, "true": 412}},
    "precision": {"labels": {"false": 0.984, "true": 0.34},
      "macro": 0.662, "micro": 0.962},
    "recall": {"labels": {"false": 0.962, "true": 0.555},
      "macro": 0.758, "micro": 0.948},
    "pr_auc": {"labels": {"false": 0.997, "true": 0.445},
      "macro": 0.721, "micro": 0.978},
    "roc_auc": {"labels": {"false": 0.923, "true": 0.923},
      "macro": 0.923, "micro": 0.923},
    ...
  }
}
```

Fig. 9. Result of <https://ores.wikimedia.org/v3/scores/enwiki/?model.info&models=damaging>

The output captured in Figure 9 shows a heavily trimmed JSON (human- and machine-readable) output of *model\_info* for the “damaging” model in English Wikipedia. Note that many fields have been trimmed in the interest of space with an ellipsis (“...”). What remains gives a taste of what information is available. Specifically, there is structured data about what kind of model is being used, how it is parameterized, the computing environment used for training, the size of the train/test set, the basic set of fitness metrics, and a version number so that secondary caches know when to invalidate old scores. A developer using an ORES model in their tools can use these fitness metrics to make decisions about whether or

not a model is appropriate and to report to users what fitness they might expect at a given confidence threshold.

*A.1.3 Explicit pipelines.* We have designed the process of training and deploying ORES prediction models to be repeatable and reviewable. Consider the following code that represents a common pattern from our model-building Makefiles:

```
datasets/enwiki.human_labeled_revisions.20k_2015.json:
    ./utility fetch_labels \
        https://labels.wmflabs.org/campaigns/enwiki/4/ > $$

datasets/enwiki.labeled_revisions.w_cache.20k_2015.json: \
    datasets/enwiki.labeled_revisions.20k_2015.json
cat $< | \
revscoring extract \
    editquality.feature_lists.enwiki.damaging \
    --host https://en.wikipedia.org \
    --extractor $(max_extractors) \
    --verbose > $$

models/enwiki.damaging.gradient_boosting.model: \
    datasets/enwiki.labeled_revisions.w_cache.20k_2015.json
cat $^ | \
revscoring cv_train \
    revscoring.scoring.models.GradientBoosting \
    editquality.feature_lists.enwiki.damaging \
    damaging \
    --version=$(damaging_major_minor).0 \
    (... model parameters ...)
    --center --scale > $$
```

---

Fig. 10. Makefile rules for the English damage detection model from <https://github.com/wiki-ai/editquality>

Essentially, this code helps someone determine where the labeled data comes from (manually labeled via the Wiki Labels system). It makes it clear how features are extracted (using the `revscoring extract` utility and the `feature_lists.enwiki.damaging` feature set). Finally, this dataset of extracted features is used to cross-validate and train a model predicting the “damaging” label and a serialized version of that model is written to a file. A user could clone this repository, install the set of requirements, and run `make enwiki_models` and expect that all of the data-pipeline would be reproduced, and an exactly equivalent model obtained.

By explicitly using public resources and releasing our utilities and Makefile source code under an open license (MIT), we have essentially implemented a turn-key process for replicating our model building and evaluation pipeline. A developer can review this pipeline for issues knowing that they are not missing a step of the process because all steps are captured in the Makefile. They can also build on the process (e.g. add new features) incrementally and restart the pipeline. In our own experience, this explicit pipeline is extremely useful for identifying

the origin of our own model building bugs and for making incremental improvements to ORES' models.

At the very base of our Makefile, a user can run `make models` to rebuild all of the models of a certain type. We regularly perform this process ourselves to ensure that the Makefile is an accurate representation of the data flow pipeline. Performing complete rebuild is essential when a breaking change is made to one of our libraries. The resulting serialized models are saved to the source code repository so that a developer can review the history of any specific model and even experiment with generating scores using old model versions.

Received April 2018