ORES: Lowering Barriers with Paricipatory Machine Learning in Wikipedia

Aaron Halfaker
Wikimedia Foundation
San Francisco, CA, USA
ahalfaker@wikimedia.org

R. Stuart Geiger
University of California, Berkeley
Berkeley, CA, USA
stuart@stuartgeiger.com

Jonathan T. Morgan
Wikimedia Foundation
San Francisco, CA, USA
jmorgan@wikimedia.org

Amir Sarabadani
Wikimedia Deutschland
Berlin, Germany
amir.sarabadani@wikimedia.de

Adam Wight
Wikimedia Foundation
San Francisco, CA, USA
awight@wikimedia.org

ABSTRACT

Algorithmic systems—from rule-based bots to machine learning classifiers—have a long history of supporting the essential work of content moderation and other curation work in peer production projects. From counter-vandalism to task routing, basic machine prediction has allowed open knowledge projects like Wikipedia to scale to the largest encyclopedia in the world, while maintaining quality and consistency. However, conversations about what quality control should be and what role algorithms should play have generally been led by the expert engineers who have the skills and resources to develop and modify these complex algorithmic systems. In this paper, we describe ORES: an algorithmic scoring service that supports real-time scoring of wiki edits using multiple independent classifiers trained on different datasets. ORES decouples three activities that have typically all been performed by engineers: choosing or curating training data, building models to serve predictions, and developing interfaces or automated agents that act on those predictions. This meta-algorithmic system was designed to open up socio-technical conversations about algorithmic systems in Wikipedia to a broader set of participants. In this paper, we discuss the theoretical mechanisms of social change ORES enables and detail case studies in participatory machine learning around ORES from the 3 years since its deployment.

FAT'18, Feb 2019, Atlanta, Georgia*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/0000001.0000001

KEYWORDS

Wikipedia, Reflection, Machine learning, Transparency, Fairness, Algorithms, Governance

ACM Reference Format:
Aaron Halfaker, R. Stuart Geiger, Jonathan T. Morgan, Amir Sarabadani, and Adam Wight. 2019. ORES: Lowering Barriers with Paricipatory Machine Learning in Wikipedia. In *Proceedings of ACM Conference on Fairness, Accountability, and Transparency (FAT*'18).* ACM, New York, NY, USA, Article Under review, 15 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Wikipedia—the free encyclopedia that anyone can edit—faces many challenges in maintaining the quality of its articles and sustaining the volunteer community of editors. The people behind the hundreds of different language versions of Wikipedia have long relied on automation, bots, expert systems, recommender systems, human-in-the-loop assisted tools, and machine learning to help moderate and manage content at massive scales. The issues around artificial intelligence in Wikipedia are as complex as those facing other large-scale user-generated content platforms like Facebook, Twitter, or YouTube, as well as traditional corporate and governmental organizations that must make and manage decisions at scale. And like in those organizations, Wikipedia's automated classifiers are raising new and old issues about truth, power, responsibility, openness, and representation.

Yet Wikipedia's approach to AI has long been different than in corporate or governmental contexts typically discussed in emerging fields like Fairness, Accountability, and Transparency in Machine Learning (FATML) or Critical Algorithms Studies (CAS). The volunteer community of editors has strong ideological principles of openness, decentralization, and consensus-based decision-making. The paid staff at the non-profit Wikimedia Foundation—which legally owns and operates the servers—are not

tasked with making editorial decisions about content[1]. This is instead the responsibility of the volunteer community, where a self-selected set of developers build tools, bots, and advanced technologies in broad consultation with the community. Even though Wikipedia's longstanding socio-technical system of algorithmic governance is far more open, transparent, and accountable than most platforms operating at Wikipedia's scale, ORES[2], the system we present in this paper, pushes even further on the crucial issue of who is able to participate in the development and use of advanced technologies.

ORES represents several innovations in openness in machine learning, particularly in seeing openness as a socio-technical challenge that is as much about scaffolding support as it is about open-sourcing code and data. With ORES, volunteers can curate labeled training data from a variety of sources for a particular purpose, commission the production of a machine classifier based on particular approaches and parameters, and make this classifier available via an API which anyone can query to score any edit to a page—operating in real time on the Wikimedia Foundation's servers. Currently, 78 classifiers have been produced for 37 languages—classifying edits in real-time based on criteria like "damaging / not damaging," "good faith / bad faith," or a language-specific article quality scale. ORES intentionally does not seek to produce a single classifier to enforce a gold standard of quality, nor does it prescribe particular ways in which scores and classifications will be incorporated into fully automated bots and semi-automated editing interfaces. Instead, ORES was built as a kind of cultural probe [20] to support an open-ended set of community efforts to re-imagine what machine learning in Wikipedia is and who it is for.

Open participation in machine learning is widely relevant to both researchers of user-generated content platforms and those working across open collaboration, social computing, machine learning, and critical algorithms studies. ORES implements several of the dominant recommendations for algorithmic system builders around transparency and community consent[5, 6, 28]. We discuss practical scoio-technical considerations for what openness, accountability, and transparency mean in a large-scale, real-world user-generated content platform. Wikipedia is also an excellent space for work on FATML topics, as the broader Wikimedia community and the non-profit Wikimedia Foundation are founded on ideals of open, public participation. All of the work presented in this paper is publicly-accessible and open sourced, from the source code and training data to the community discussions about ORES. Unlike in other nominally 'public' platforms where users often do not know their data is used for research purposes, Wikipedians have extensive discussions about using their archived activity for research, with established guidelines we followed.[3] This project is part of a longstanding engagement with the volunteer communities which involves extensive community consultation, and the case studies research have been approved by UC-Berkeley's IRB.

We first review related literature around open algorithmic systems, then discuss the socio-technical context of Wikipedia that lead us to building ORES. We discuss the operation of ORES, highlighting innovations in algorithmic *openness* and *transparency*. We present case studies of ORES that illustrate how it has broadened participation in machine learning. Finally, we conclude with a discussion of the issues raised by this work and identify future directions.

2 RELATED WORK

The politics of algorithms

Algorithmic systems play increasingly crucial roles in the governance of social processes[12]. Software algorithms are increasingly used in answering questions that have no single right answer and where prior human decisions used as training data can be problematic [2]. Algorithms designed to support work change people's work practices, shifting how, where, and by whom work is accomplished[5, 33]. Software algorithms gain political relevance on par with other process-mediating artifacts (e.g. laws[23]).

There are repeated calls to address power dynamics and bias through transparency and accountability of the algorithms that govern public life and access to resources[7, 28]. The field around effective transparency and accountability mechanisms is growing. We cannot fully address the scale of concerns in this rapidly shifting literature, but we find inspiration in Kroll et al's discussion of the potential and limitations of auditing and transparency [22] and Geiger's call to go "beyond opening up the black box[9]".

We discuss a specific socio-political context—Wikipedia's algorithmic quality control and socialization practices—and the development of novel algorithmic systems for support of these processes. We implement a meta-algorithmic intervention aligned with Wikipedians' principles and practices: deploying a set of prediction algorithms as a

[1] Except in rare cases, such as content that violates U.S. law, see http://enwp.org/WP:OFFICE

[2] https://ores.wikimedia.org and http://enwp.org/:mw:ORES

[3] See http://enwp.org/WP:NOTLAB and http://enwp.org/WP:Ethically_researching_Wikipedia

service and leaving decisions about appropriation to the volunteer community. Instead of training the single best classifier and implementing it in our own designs, we embrace public auditing, re-interpretations, and appropriations of our models' 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 socio-infrastructural intervention on communities of practice.

Machine prediction in support of open production

Open peer production systems, like all user-generated content platforms, have a long history of using machine learning for content moderation and task management. For Wikipedia and related Wikimedia projects, vandalism detection and quality control is a major goal 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 and their product at risk, so all edits must be reviewed as soon as possible [11].

As an information overload problem, filtering strategies using machine learning models have been developed to support the work of Wikipedia's patrollers (see [1] for an overview). In some cases, researchers directly integrated their prediction models into specific, purpose-designed tools for Wikipedians to use (e.g. STiki[32], a classifier-supported human-computation tool). Through the use of these machine learning models and boundary patrolling, most damaging edits are reverted within seconds of when they are saved[10].

Task routing and recommendation. Machine learning plays a major role in how Wikipedians decide what articles to work on, supplementing the standard self-selected dynamic of people contributing to topics they are interested in. Wikipedia has many well-known content coverage biases (e.g. for a long period of time, the coverage of women scientists in Wikipedia lagged far behind the rest of the encyclopedia [16]). Past work has explored collaborative recommender-based task routing strategies (see SuggestBot [4]), in which contributors are sent articles that need improvement in their areas of expertise. Such systems show strong promise to address content coverage biases, but could also inadvertently reinforce biases.

The Rise and Decline: Wikipedia's socio-technical problems

While Wikipedians have successfully deployed algorithmic quality control support systems to maintain Wikipedia, a line of critical research has studied the unintended consequences of this complex socio-technical system, particularly on newcomer socialization [17, 18, 24]. In summary, Wikipedians struggled with the issues of scaling when the popularity of Wikipedia grew exponentially between 2005 and 2007[17]. In response, they developed quality control processes and technologies that prioritized efficiency by using machine prediction models [18] and templated warning messages [17]. This transformed newcomer socialization from a primarily human and welcoming activity to one that is more dismissive and impersonal [24] and has caused in a steady decline in Wikipedia's editing population. 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.

After the research about this systemic issue came out, the political importance of newcomer experience was raised substantially. But despite targeted efforts and shifts in perception among some members of the Wikipedia community [24, 26][4], the quality control processes that were designed over a decade ago remain largely unchanged [18].

3 DESIGN RATIONALE

In this section, we discuss systemic mechanisms behind Wikipedia's socio-technical problems and how we as system builders designed ORES to have impact within Wikipedia. Past work has demonstrated how Wikipedia's problems are systemic and caused in part to inherent biases in the system of quality control. To responsibly use machine learning in addressing these problems, we examined how Wikipedia functions as a distributed system, focusing on how processes, policies, power, and software come together to make Wikipedia happen.

Our goal: Lowered barriers to participation

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 real-time machine prediction model. Without exception, all of the critical, high efficiency quality control systems that keep

[4]See also a team dedicated to supporting newcomershttp://enwp.org/:m:Growthteam

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 and STiki[5] use machine prediction models to highlight likely damaging edits for human reviews. ClueBot NG[6] uses a machine prediction model to automatically revert edits that are highly likely to be damaging.

Wikipedia's quality control processes have always been open for re-consideration, but only for those with the right technical skills and capacities. We have two options for expanding the margins of this conversation[25]: (1) increase general literacy around machine classification techniques and operations at scale; or (2) minimize the need to navigate the technicalities of machine learning at scale in order to experiment with and develop novel advanced algorithmic technologies.

Our goal in the development of ORES is to explore the second option. By deploying a high-availability machine prediction service, designing accessible interfaces, and engaging in basic outreach efforts, we seek to dramatically lower the barriers to the development of new algorithmic systems that could implement radically new ideas about what should be classified, how it should be classified, and how classifications and scores should be used.

Our measure of success: More voices

We measure success not through higher rates of precision and recall (though we are, of course, interested in that as well), but instead though the new conversations about how algorithmic tools affect editing dynamics. If ORES is a successful intervention, it will enable experimentation in Wikipedia's socio-technical *conversations* about quality control. This translates into the development of novel tools and serious, critical reflection on the roles that algorithms play in mediating Wikipedia's quality and newcomer support processes. If we only see the same discussions (e.g. "How do we make vandal fighting more efficient?") and similar tools focused on quality control to the exclusion of newcomer socialization, we'll know that we have missed the mark.

4 THE ORES SYSTEM

ORES has been iteratively engineered to meet the needs of Wikipedia editors and the tools that support their work. 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 engineering team) using varied sources of *training data*. The models that ORES hosts are engineered to support Wikipedian 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 users, 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 code for feature extraction. 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. See sections A for detailed examples of ORES' outputs and descriptions of how we engineered the system to support Wikipedians' work practices and the tools they use.

5 INNOVATIONS IN OPENNESS

We developed ORES in the context of Wikipedia, an egalitarian, decentralized, and radically transparent community. So with ORES, we sought to maintain these values in our system design and model building strategies. The flow of data, from random samples through model training, evaluation, and application, is open for review, critique, and iteration. Further, we have developed novel strategies for opening ORES models to play and experimentation based on user requests. In this section, we describe some of the key, novel innovations that have made ORES fit Wikipedian concerns and be flexible to re-appropriation. The appendix also contains information about ORES' detailed prediction output (section A), how users and tools can adjust their use to model fitness (sections A and A), and how the whole model development workflow is made inspectable and replicable (section A).

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, the MediaWiki platform records a rich set of digital traces that can be assumed to reflect a useful human judgement for modeling. For example, in Wikipedia, it is very common that damaging edits will eventually be reverted and that good edits will not be reverted. Thus the revert action (and remaining traces) can be used as an endogenous label

[5] http://enwp.org/WP:STiki
[6] http://enwp.org/User:ClueBot_NG

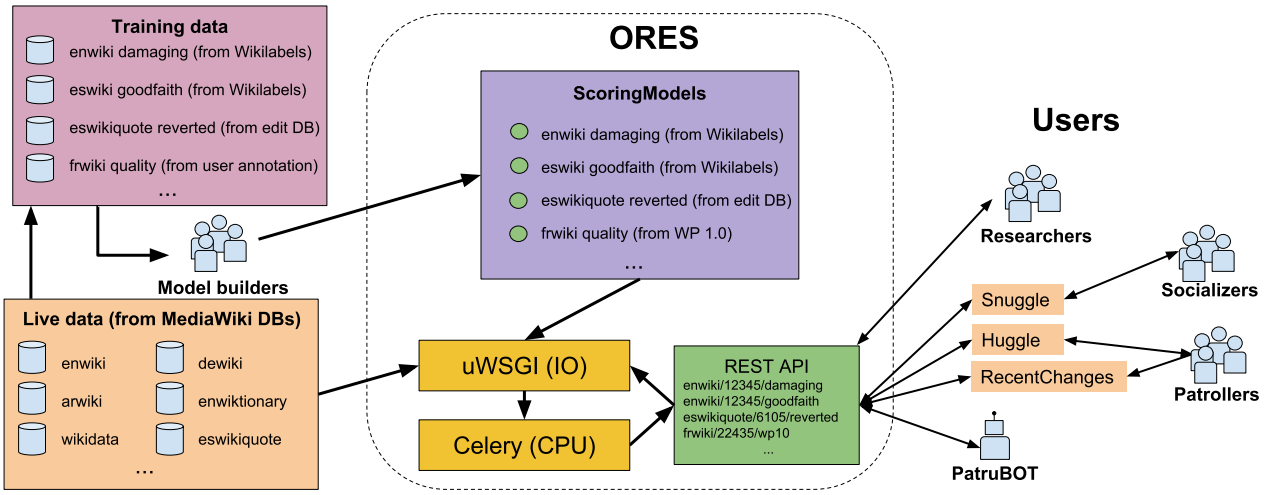


Figure 1: ORES conceptual overview. Model builders design process for training ScoringModels from training data. ORES hosts ScoringModels and makes them available to researchers and tool developers.

in training. We have developed a re-usable script[7] that when given a sample of edits, will label the edits as "reverted_for_damage" or not based on a set of constraints: the edit was reverted within 48 hours, the reverting editor was not the original editor, and the edit was not later restored by someone other than the original 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. Operationalizing quality by exclusively measuring what persists in Wikipedia reinforces Wikipedia's well-known systemic biases, which is a similar problem in using found crime data in predictive policing. 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 only make use of the "reverted_for_damage" label when manually labeled data is not available.

Manual labeling campaigns with Wiki Labels. We hold manual labeling by human Wikipedians 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, which deceptively appears to be a better option because of its apparent completeness: every edit was either reverted or not. In contrast, manual labeling has a high up-front expense of human labor. To minimize that cost, we developed a high-speed, collaborative labeling interface called "Wiki Labels[8]" to allow Wikipedians to efficiently label large datasets.

For example, to supplement our models of edit quality, we replace the models based on "reverted_for_damage" found traces with judgments from a community labeling campaign, where we specifically ask labelers to distinguish "damaging" edits from "good-faith" edits. "Good faith" is a well-established term in Wikipedian culture, with specific local meanings that are different than their broader colloquial use—similar to how Wikipedians define "consensus" or "neutrality". Using these labels we can build two separate models which allow users to filter for edits that are likely to be good-faith mistakes[15], to just focus on vandalism, or to apply themselves broadly to all damaging edits.

Dependency injection and interrogability

One of the key features of ORES that allows scores to be generated in an efficient and flexible way is a dependency injection framework. We use a dependency solver to determine what data is necessary for a scoring job and eventually compute the features used by a prediction model.

The flexibility provided by the dependency injection framework lets 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

[7] see *autolabel* in https://github.com/wiki-ai/editquality

[8] http://enwp.org/:m:Wikilabels

unregistered (anon) editors differently from registered editors. Figure 2 demonstrates two prediction requests to ORES.

Figure 2a 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 2b 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 through experience with practical examples.

Interrogability has also been used in creative new wyas beyond bias explorations. Some of our users have levered the feature injection system to expose *hypothetical* predictions to support their work. See the discussion of Ross's work recommendation tools in Section **??**.

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 a recent understanding of the importance of newcomer socialization. We do not 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. 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.[9]

When we first released ORES, there was a wave of adoption in tools that were already used by Wikipedians. Machine predictions proved useful as an addition to already-engineered systems. While this dynamic itself is fascinating, for the purposes of this paper, we focus on the development of new tools that use ORES 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 logical conclusion with an initiative that they referred to as Edit Review Improvements.[10] In this interface, ORES scores are prominently featured at the top of the list of available filters, 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[3]. After we deployed ORES, several wikis developed such bots of their own. For example, PatruBOT in Spanish Wikipedia[11] and Dexbot in Persian Wikipedia[12] now automatically revert edits that ORES predicts are damaging with high confidence.

One of the most noteworthy new applications of ORES is the suite of tools developed by Sage Ross to support the Wiki Education Foundation's[13] 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 he interprets meaning from ORES' article quality models[27] (an example of re-appropriation) and he has used the article quality model in their new editor support dashboard[14] in a novel way. Specifically, Ross's tool[15] uses our feature injection system (see Section 5) to suggest work to new editors. This system asks ORES to score a student's draft article and then asking ORES to reconsider the predicted quality level of the article with *one more header*, *one more image*, or *one more citation*. In doing so, Ross built an intelligent user interface that can expose 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.

7 CASE STUDIES IN REFLECTION

When we first deployed ORES, we reached out to several different wiki communities and invited them to test the system for use in patrolling for vandalism. Before long, our users began filing false-positive reports on wiki pages of their own design—some after our request, but mostly on their own. In this section, we 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 about model bias.

[9] See complete list: http://enwp.org/:mw:ORES/Applications
[10] http://enwp.org/:mw:Edit_Review_Improvements
[11] https://es.wikipedia.org/wiki/Usuario:PatruBOT
[12] https://fa.wikipedia.org/wiki/User:Dexbot
[13] https://wikiedu.org/
[14] https://dashboard-testing.wikiedu.org
[15] 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

Figure 2: 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.

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, created a page for ORES[16] with a section for reporting false-positives ("falsi positivi"). Within several hours, Rotpunkt and a few other editors noticed some trends. These editors began to collect false positives under different headers representing themes they were seeing. Through this process, editors from Italian Wikipedia were effectively performing an inductive, grounded theory-esque exploration ORES errors, trying to identify themes and patterns in the errors that ORES was making.

One of the themes they identified fell under the header: "corrections to the verb for *have*" ("correzioni verbo avere"). It turns out that the word "ha" in Italian translates to the English verb "to have". While in Englgh and many other langauges, "ha" is laughing and adding "ha" repeatedly is a common type of vandalism seen in all langauges of Wikipedia. We'd built a common feature in the damage model called "informal words" that captured these types of patterns. But in this case, it was clear that in Italian "ha" should not carry signal while "hahaha" still should.

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.

PatruBOT (Spanish Wikipedia)

Soon after we released support for Spanish Wikipedia, a volunteer developer made a bot to automatically revert edits using ORES's predictions for the "damaging" model (PatruBOT). This bot was not running for long before our discussion spaces were 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 about PatruBOT and its activities.

When we examined the case, we found it was one of tradeoffs between precision/recall and false positives/negatives—a common issue with machine learning applications. We concluded that PatruBOT's threshold for reverting was too sensitive. ORES reports a classification and a probability score, but it is up to the developers to decide if, for example, the bot will only auto-revert edits classified as damage with a .90, .95, .99, or higher likelihood estimate. A higher threshold will minimize the chance a good edit will be mistakenly auto-reverted, but also increase the chance that a bad edit will not be auto-reverted. Ultimately, deciding where to draw the line between false positives and false negatives is a decision for that volunteer editing community.

The Spanish Wikipedians who were concerned with these issues began a discussion about PatruBOT's activities and blocked the bot until the issue was sorted. Using wiki pages, they organized an crowdsourced evaluation of the fitness of PatruBOT's behavior[17]. This evaluation and discussion is ongoing, [18] but it shows how stakeholders do not need to have an advanced understanding in machine learning evaluation to meaningfully participate in a sophisticated discussion about how, when, why, and under what conditions such classifiers should be used.

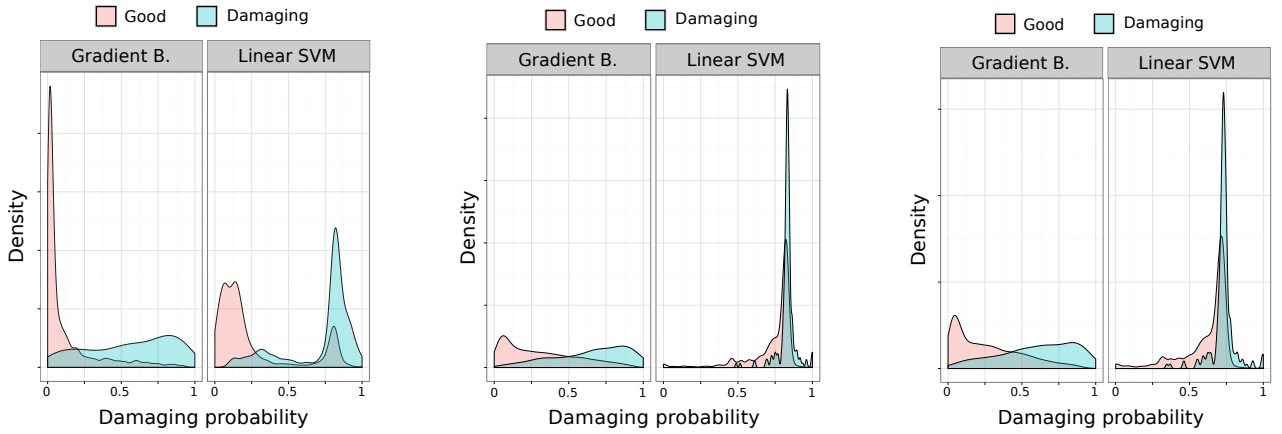
Bias against anonymous editors

Shortly after we deployed ORES, we received reports that ORES's damage detection models were overly biased against anonymous editors. At the time, we were using

[16] https://it.wikipedia.org/wiki/Progetto:Patrolling/ORES

[17] https://es.wikipedia.org/wiki/Wikipedia:Mantenimiento/Revisi%C3%B3n_de_errores_de_PatruBOT%2FAn%C3%A1lisis

[18] https://es.wikipedia.org/wiki/Wikipedia:Caf%C3%A9%2FArchivo%2FMiscel%C3%A1nea%2FActual#Parada_de_PatruBOT



(a) No injected features **(b) Everyone is anonymous** **(c) Everyone is newly registered**

Figure 3: 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.

Linear SVM[19] estimators to build classifiers, and we were considering making the transition towards ensemble strategies like GradientBoosting and RandomForest estimators.[20] We took the opportunity to look for bias in the error of estimation between anonymous editors and 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.

Figure 3 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 3a 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 3b and 3c 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 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 weakness of our estimator and show that an improvement to our modeling strategy mitigates the problem. Without such a tight feedback loop, we most likely would not have noticed how poorly ORES's 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[21] [17].

8 CONCLUSION AND FUTURE WORK

ORES as a socio-technical system has helped us 1) refine our understandings of volunteers' needs across wiki communities, 2) identify and address biases in ORES's models, and 3) reflect on how people think about what types of automation they find acceptable in their *spaces*. Through our participatory design process with various Wikipedian communities, we've arrived at several innovations in open machine learning practice that represent advancements in the field.

As we stated in Section 3, we measure success in 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. We have demonstrated through discussion of adoption patterns and case studies in reflection around the use of algorithmic systems that something fundamental is *working*. ORES is

[19] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[20] http://scikit-learn.org/stable/modules/ensemble.html
[21] http://enwp.org/:en:Wikipedia:IPs_are_human_too

being heavily adopted. The meaning of ORES models is being re-appropriated. Both the models and the technologies that use the models are being collaboratively audited by their users and those who are affected.

Participatory machine learning

In a world dominated by for-profit social computing and user-generated content platforms—often marketed by their corporate owners as "communities"[13]—Wikipedia is an anomaly. While the non-profit Wikimedia Foundation has only a fraction of the resources as Facebook or Google, the unique principles and practices in the broad Wikipedia/Wikimedia movement are a generative constraint. ORES emerged out of this context, operating at the intersection of a pressing need to deploy efficient machine learning at scale for content moderation, but to do so in ways that enable volunteers to develop and deploy advanced technologies on their own terms. Our approach is in stark contrast to the norm in machine learning research and practice, which involves a more top-down mode of developing the most precise classifiers for a known ground truth, then wrap those classifiers in a complete technology for end-users, who must treat them as black boxes.

The more wiki-inspired approach to what we call "participatory machine learning" imagines classifiers to be just as provisional and open to skeptical reinterpretation as the content of Wikipedia's encyclopedia articles. And like Wikipedia articles, we suspect some classifiers will be far better than others based on how volunteers develop and curate them, for various definitions of "better" that are already being actively debated. Our case studies briefly indicate how volunteers have collectively engaged in sophisticated discussions about how they ought to use machine learning. ORES' fully open, reproducible, and auditable code and data pipeline—from training data to models to scored predictions—enables a wide range of new collaborative practices. ORES is a more socio-technical approach to issues in FATML, where attention is often placed on technical solutions, like interactive visualizations for model interpretability or mathematical guarantees of operationalized definitions of fairness. Our approach is specific to the particular practices and values of Wikipedia, and we have shown how ORES has been developed to fit into this context.

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 collaborative auditing 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. We are inspired by much of the concern that has surfaced for looking into biases in ORES' prediction models (e.g. anon bias and the Italian "ha") and over what role algorithms should have in directly reverting human actions (e.g. PatruBOT and Dexbot).

Eliciting this type of critical reflection and empowering users to engage in their own choices about the roles of algorithmic systems in their social spaces has typically been more of a focus from the Critical Algorithms Studies literature (e.g. [2, 21]. This literature also emphasizes a need to see algorithmic systems as dynamic and constantly under revision by developers [30]—work that is invisible in most platforms, but foregrounded in ORES. In these case studies, we see that given ORES' open API and Wikipedia's collaborative wiki pages, Wikipedians will 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.

Future work

Observing ORES in practice suggests avenues of future work toward crowd-based auditing tools. As our case studies suggest, auditing of ORES' predictions and mistakes has become a very popular activity. Even though we did not design interfaces for discussion and auditing, some Wikipedians have used unintended affordances of wiki pages and MediaWiki's template system to organize processes for flagging false positives and calling them to our attention. This process has proved invaluable for improving model fitness and addressing critical issues of bias against disempowered contributors. 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 what machine prediction gets right and wrong, we can make it easier for tools that use ORES to also allow for reporting mistakes and for others to infer trends. For example, a database of ORES mistakes could be queried 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 (e.g. PatruBOT).

We also look forward to what those in critical algorithm studies can do with ORES. Most of the studies and critiques of *subjective algorithms*[31] focus on for-profit organizations that are strongly resistant to opening up. 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 that leads to some people's contributions remaining and others being removed. This is a context where *algorithms matter to humanity*, and we are openly experimenting with the kind of transparent and open processes that *fairness and transparency in machine learning* 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 without our engineering team involved in the process. Currently, ORES is only suited to deploy models that are trained and tested 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[22][23] 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 to 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] Solon Barocas, Sophie Hood, and Malte Ziewitz. 2013. Governing algorithms: A provocation piece. *SSRN. Paper presented at Governing Algorithms conference.* (2013). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2245322

[3] Jacobi Carter. 2008. ClueBot and vandalism on Wikipedia. https://web.archive.org/web/20120305082714/http://www.acm.uiuc.edu/~carter11/ClueBot.pdf

[4] 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.

[5] Kate Crawford. 2016. Can an algorithm be agonistic? Ten scenes from life in calculated publics. *Science, Technology, & Human Values* 41, 1 (2016), 77–92.

[6] Nicholas Diakopoulos. 2015. Algorithmic accountability: Journalistic investigation of computational power structures. *Digital Journalism* 3, 3 (2015), 398–415.

[7] Nicholas Diakopoulos and Michael Koliska. 2017. Algorithmic Transparency in the News Media. *Digital Journalism* 5, 7 (2017), 809–828. https://doi.org/10.1080/21670811.2016.1208053

[8] R Stuart Geiger. 2011. The lives of bots. In *Critical Point of View: A Wikipedia Reader*. Institute of Network Cultures, Amsterdam, 78–93. http://stuartgeiger.com/lives-of-bots-wikipedia-cpov.pdf

[9] R. Stuart Geiger. 2017. Beyond opening up the black box: Investigating the role of algorithmic systems in Wikipedian organizational culture. *Big Data & Society* 4, 2 (2017), 2053951717730735. https://doi.org/10.1177/2053951717730735

[10] 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.

[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.

[12] Tarleton Gillespie. 2014. The relevance of algorithms. *Media technologies: Essays on communication, materiality, and society* 167 (2014).

[13] Tarleton Gillespie. 2018. *Custodians of the internet : platforms, content moderation, and the hidden decisions that shape social media*. Yale University Press, New Haven.

[14] Aaron Halfaker. 2016. Notes on writing a Vandalism Detection paper. http://socio-technologist.blogspot.com/2016/01/notes-on-writing-wikipedia-vandalism.html

[15] 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

[16] 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.

[17] Aaron Halfaker, R Stuart Geiger, Jonathan T Morgan, and John Riedl. 2013. The rise and decline of an open collaboration system: How Wikipedia's reaction to popularity is causing its decline. *American Behavioral Scientist* 57, 5 (2013), 664–688.

[18] 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*

[22] http://jupyter.org

[23] e.g. https://github.com/wiki-ai/editquality/blob/master/ipython/reverted_detection_demo.ipynb

factors in computing systems. ACM, 311–320.
[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] 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.
[21] Rob Kitchin. 2017. Thinking critically about and researching algorithms. *Information, Communication & Society* 20, 1 (2017), 14–29. https://doi.org/10.1080/1369118X.2016.1154087
[22] 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.
[23] Lawrence Lessig. 1999. *Code: And other laws of cyberspace*. Basic Books.
[24] 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.
[25] 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.
[26] 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.
[27] Sage Ross. 2016. Visualizing article history with Structural Completeness. https://wikiedu.org/blog/2016/09/16/visualizing-article-history-with-structural-completeness/
[28] 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.
[29] 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.
[30] Nick Seaver. 2017. Algorithms as culture: Some tactics for the ethnography of algorithmic systems. *Big Data & Society* 4, 2 (2017). https://doi.org/10.1177/2053951717738104
[31] 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.
[32] 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.
[33] Shoshana Zuboff. 1988. *In the age of the smart machine: The future of work and power*. Vol. 186. Basic books New York.

A APPENDIX

Score documents

The predictions made by 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 5.


```
"wp10": {
  "score": {
    "prediction": "Start",
    "probability": {
      "FA": 0.00329313015, "GA": 0.0058529554,
      "B": 0.06062338048, "C": 0.01991363271,
      "Start": 0.754330134, "Stub": 0.1559867667
    }
  }
}
```


Figure 4: 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 classed (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[27] or the *weighted sum* metric from [16]).

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.

The output captured in Figure 5 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

```
"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},
    ...
  }
}
```


Figure 5: Result of https://ores.wikimedia.org/v3/scores/enwiki/?model_info&models=damaging

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.

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 to catch all (or almost all) vandalism before it stays in Wikipedia for very long. That means they have an operational concern around the *recall* of a damage prediction model. They would 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[14].

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. In response, 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}
```


Figure 6: Result of 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, a user 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.

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:

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 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.

System design

In this section we describe how the system was designed in order to meet the needs of Wikipedian work practices and the tools that support them.

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 Wikipedian quality control work. Interruptions in Wikipedia's algorithmic systems have historically led to increased burdens for human workers and a higher likelihood that readers will see vandalism[10]. Further, ORES needs to scale to be able to be used in multiple different tools across different language Wikipedias where its predecessors only needed to scale for use in a single tool.

This horizontal scaleability is achieved in two ways: input-output (IO) workers (uwsgi[24]) and the computation (CPU) workers (celery[25]). Requests are split across available IO workers, and all necessary data is gathered using external APIs (e.g. the MediaWiki API[26]). 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 taking down the service as a whole.

Real-time processing. The most common use case of ORES is real-time processing of edits to Wikipedia immediately after they are saved. 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 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 so that counter-vandalism efforts are not substantially delayed(c.f. [**?**]). Our metrics tracking currently suggests that for the week April 6-13th, 2018, 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[27] 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

[24] https://uwsgi-docs.readthedocs.io/
[25] http://www.celeryproject.org/
[26] http://enwp.org/:mw:MW:API
[27] Implemented natively by Redis, https://redis.io

```
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 > $@
```

Figure 7: Makefile rules for the English damage detection model from https://github.com/wiki-ai/editquality

requests scores for a specific subset of actions (e.g. edits). With our LRU and precaching strategy, we consistently attain a cache hit rate of about 80%.

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 are de-duplicated. This allows our service to trivially scale to support many different robots and tools on the same wiki.

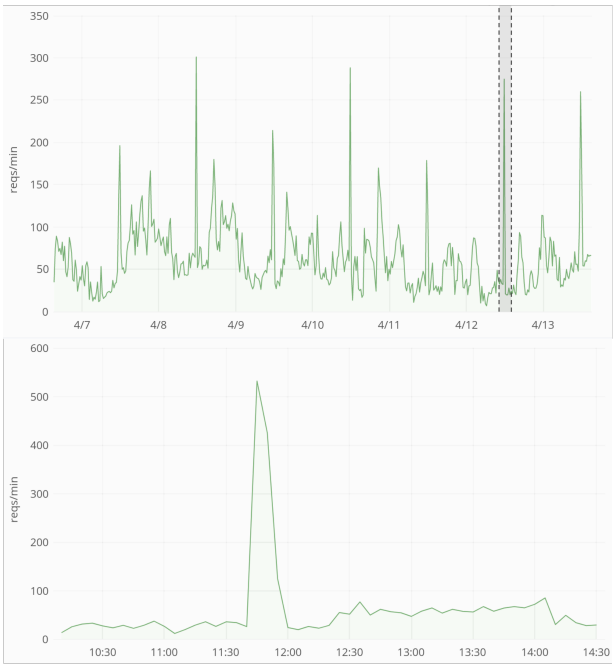
Batch processing. Many different types of Wikipedia's bots rely on periodic, batch processing strategies to support Wikipedian work processes[8]. For example, many bots are designed to build worklists for Wikipedia editors (e.g. [4]) on a daily or weekly basis, and many of these tools have adopted ORES to include an article quality prediction for use in prioritization of work (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[28]). 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 for all relevant scoring jobs in batch queries. 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[29]. At this rate, a user can request 10s of million of scores in less than 24 hours in the worst case scenario (no scores were cached) without substantially affecting the service for others.

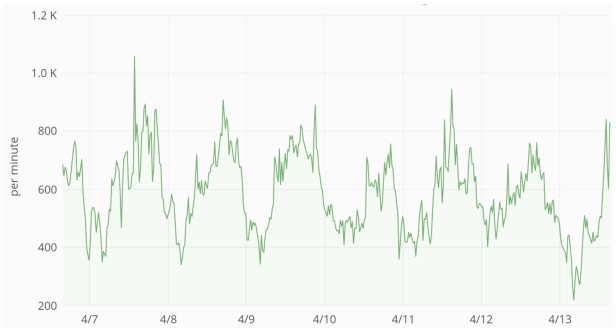
Empirical access patterns. The ORES service has been online since July 2015[19]. Since then, usage has steadily risen as we've developed and deployed new models and additional integrations are made by tool developers and researchers. Currently, ORES supports 78 different models and 37 different language-specific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

[28] 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

Figure 8: Request rates to the ORES service for the week ending on April 13th, 2018

to 400-500 requests per second. Figure 8a shows the periodic and "bursty" nature of scoring requests received by the ORES service. For example, every day at about 11:40 UTC, the request rate jumps—most likely a batch scoring job such as a bot.

Figure 8b shows the 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 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 boundary-lowering intervention. By engineering a system that conforms to the work-process needs of Wikipedians and their tools, we've built a systems intervention that has the potential gain wide adoption in Wikipedia's technical ecology.

Received April 2018; revised July 2018; revised August 2018