Overview of the Wikidata Vandalism Detection Task at WSDM Cup 2017

Stefan Heindorf[1] Martin Potthast[2] Gregor Engels[1] Benno Stein[3]

[1]Paderborn University
<last name>@uni-paderborn.de

[2]Leipzig University
<first name>.<last name>@uni-leipzig.de

[3]Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

ABSTRACT

We report on the Wikidata vandalism detection task at the WSDM Cup 2017. The task received five submissions for which this paper describes their evaluation and a comparison to state of the art baselines. Unlike previous work, we recast Wikidata vandalism detection as an online learning problem, requiring participant software to predict vandalism in near real-time. The best-performing approach achieves a ROC_{AUC} of 0.947 at a PR_{AUC} of 0.458. In particular, this task was organized as a *software submission task*: to maximize reproducibility as well as to foster future research and development on this task, the participants were asked to submit their working software to the TIRA experimentation platform along with the source code for open source release.

1. INTRODUCTION

Knowledge is increasingly gathered by the crowd. One of the most prominent examples in this regard is Wikidata, the knowledge base of the Wikimedia Foundation. Wikidata stores knowledge (better: facts) in structured form as subject-predicate-object statements that can be edited by anyone. Most of the volunteers' contributions to Wikidata are of high quality; however, there are, just like in Wikipedia, some "editors" who vandalize and damage the knowledge base. The impact of these few can be severe: since Wikidata is, to an increasing extent, integrated into information systems such as search engines and question-answering systems, the risk of spreading false information to all their users increases as well. It is obvious that this threat cannot be countered by human inspection alone: currently, Wikidata gets millions of contributions every month; the effort of reviewing them manually will exceed the resources of the community, especially as Wikidata further grows.

Encouraged by the success of algorithmic vandalism detection on Wikipedia, we started a comparable endeavor for Wikidata two years ago: we carefully compiled a corpus based on Wikidata's revision history [6] and went on by developing the first machine learning-based Wikidata vandalism detector [7]. Compared with the quality of the best vandalism detectors for Wikipedia, our results may be considered as a first step toward a practical solution.

We are working on new detection approaches ourselves but we see that progress can be made at a much faster pace if independent researchers work in parallel, generating a diversity of ideas. While research communities often form around problems of interest, this has not been the case for vandalism detection in knowledge bases—perhaps due to the novelty of the task. We hence took a proactive stance by organizing a shared task event as part of the WSDM Cup 2017 [8]. Shared tasks have proven successful as catalysts for forming communities on a number of occasions before, in particular for vandalism detection on Wikipedia: on the basis of two shared tasks, considerable interest from researchers worldwide resulted in dozens of approaches to date [14, 16].

The goal of our shared task at the WSDM Cup 2017 is to develop an effective vandalism detection model for Wikidata:

> Given a Wikidata revision, the task is to compute a quality score denoting the likelihood of this revision being vandalism (or, similarly, damaging).

The revisions had to be scored in near real-time as soon as they arrived, allowing for immediate action upon potential vandalism. Moreover, a model (detector) should hint at vandalism across a wide range of precision-recall-points to enable use cases such as (1) fully automatic reversion of damaging edits at high precision, as well as (2) pre-filtering and ranking of revisions with respect to importance of being reviewed at high recall. As our main evaluation metric we employ the area under curve of the receiver operating characteristic.

Our contributions to the WSDM Cup 2017 are as follows:

- Compilation of the Wikidata Vandalism Corpus 2016, an updated version of our previous corpus, enlarged and retrofitted for the setup of the shared task.
- Survey of the participant approaches with regard to features and model variants.
- Comparison of the participant approaches to the state of the art under a number of settings beyond the main task.
- Analysis of the combined performance of all models (detectors) as an ensemble in order to estimate the achievable performance when integrating all approaches.
- Release of an open source repository of the entire evaluation framework of the shared task, as well as release of most of the participants' code bases by courtesy of the participants.

In what follows, Section 2 briefly reviews the aforementioned related work, Section 3 introduces the developed evaluation framework including the new edition of the Wikidata vandalism corpus, Section 4 surveys the submitted approaches, and Section 5 reports on their evaluation. Finally, Section 6 reflects on the lessons learned.

2. RELATED WORK

The section gives a comprehensive overview of the literature on vandalism detection. The two subsections detail the approaches regarding dataset construction and detection technology respectively.

2.1 Corpus Construction

There are basically three strategies to construct vandalism corpora for Wiki-style projects [9], namely, (1) based on independent manual review of edits, (2) based on exploiting community feedback about edits, and (3) based on comparing item states. As expected, there is a trade-off between corpus size, label quality, and annotation costs. Below, we review state-of-the art approaches for constructing vandalism corpora under each strategy.

Annotation Based on Independent Manual Review.
The most reliable approach to construct a vandalism corpus is to manually review and annotate its edits. When facing millions of edits, however, the costs for a manual review become prohibitive, thus severely limiting corpus size. The largest manually annotated vandalism corpus to date is the PAN Wikipedia Vandalism Corpus 2010 and 2011, comprising a sample of 30,000 edits each of which having manually been annotated via crowdsourcing using Amazon's Mechanical Turk [13]. About 7% of the edits have been found to be vandalism. This approach, however, is probably not suited to Wikidata: an average worker on Mechanical Turk is much less familiar with Wikidata, and the expected ratio of vandalism in a random sample of Wikidata edits is about 0.2% (compared with 7% in Wikipedia), so that a significantly higher number of edits would have to be reviewed in order to obtain a sensible number of vandalism cases for training a model.

Annotation Based on Community Feedback.
A more scalable approach to construct a vandalism corpus is to rely on feedback about edits provided by the community for annotations. However, not all edits made to Wikidata are currently reviewed by the community, thus limiting the recall in a sample of edits to the amount of vandalism that is actually caught, and, not all edits that are rolled back are true vandalism. Nevertheless, for its simplicity, this approach was adopted to construct the Wikidata Vandalism Corpus (WDVC) 2015 [6] and 2016, whereas the latter was employed as evaluation corpus at the WSDM Cup 2017. Both corpus versions are freely available for download.[1] The corpus construction is straightforward: based on the portion of the Wikidata dump with manually created revisions, those revisions that have been reverted via Wikidata's rollback facility are labeled vandalism. The rollback facility is a special instrument to revert vandalism; it is accessible to privileged Wikidata editors only. This makes our corpus robust against manipulation by non-privileged and, in particular, anonymous Wikidata editors. As a result, we obtain a large-scale vandalism corpus comprised of more than 82 million manual revisions that were made between October 2012 and June 2016. About 200,000 revisions have been rolled back in this time (and hence are labeled vandalism). By a manual analysis we got evidence that 86% of the revisions labeled vandalism are indeed vandalism as per Wikidata's definition [6].[2] Recently, vandalism corpora for Wikipedia have also been constructed based on community feedback.

Tran and Christen [21, 22] and Tran et al. [23] label all revisions with certain keywords in the revision comment as vandalism, e.g., 'vandal' or 'rvv' (revert due to vandalism), based on Kittur et al.'s [10] approach to identify vandalism. These keywords do not work for Wikidata, since revision comments are almost always automatically generated and cannot be changed by editors.

Annotation Based on Item State.
An alternative (and still scalable) approach to build a vandalism corpus is to analyze recurring item states in order to identify so-called item "reverts" to previous states in the respective item history. In addition to community feedback this approach does also consider all other events that may have caused an item state to reappear, e.g., in case that someone just fixes an error without noticing that the error was due to vandalism. As a consequence, a higher recall can be achieved whereas, however, a lower precision must be expected. Sarabadani et al. [19] adopted this approach, and, in order to increase precision, they suggested a set of rules to annotate an edit as vandalism if and only if (1) it is a manual edit, (2) it has been reverted, (3) it does not originate from a pre-defined privileged editor group,[3] (4) it has not been propagated from a dependent Wikipedia project, and (5) it does not merge two Wikidata items. For unknown reasons, not all edits of Wikidata's history are annotated this way but only a subset of 500,000 in 2015, yielding altogether only about 20,000 vandalism edits. While the authors claim superiority of the design of their corpus over ours, their self-reported precision values are not convincing: while only 68% of the edits labeled vandalism are in fact vandalism (86% in our corpus), 92% of the edits are reported to be at least "damaging" to a greater or lesser extent. The authors have reviewed only 100 edits to substantiate these numbers (we have reviewed 1,000 edits), so that these numbers must be taken with a grain of salt.

Altogether, both corpora are suboptimal with regard to recall: within both corpora, about 1% of the edits are wrongly labeled non-vandalism, which currently amounts to an estimated 800,000 missed vandalism edits over Wikidata's entire history. A machine learning approach to vandalism detection must hence be especially robust against false negatives in the training dataset.

Tan et al. [20] compiled a dataset of low-quality triples in Freebase according to the following heuristics: Additions that have been deleted within 4 weeks after their submission are considered low-quality, as well as removals that have not been reinserted within 4 weeks. The manual investigation of 200 triples revealed a precision of about 89%. However, the usefulness of the Freebase data set is restricted by the fact that Google has shut down Freebase; the Freebase data is currently being transferred to Wikidata [12].

2.2 Vandalism Detection

The detection of vandalism and damaging edits in *structured* knowledge bases such as triple stores is a new research area. Hence, only three approaches have been published before the WSDM Cup 2017, which represent the state of the art [7, 19, 20]. All employ machine learning, using features derived from both an edit's content and its context. In what follows, we briefly review them.

The most effective approach to Wikidata vandalism detection, WDVD, was proposed by Heindorf et al. [7]. It implements 47 features, from which 27 encode an edit's *content* and 20 an edit's *context*. The content-based features cover character level features, word level features, and sentence-level features, all are computed from the automatically generated revision comment. In addition, WDVD employs features to capture predicates and objects of subject-predicate-object triples. Context-based features include user reputation, user geolocation, item reputation, and the meta data of an edit. As classification algorithm random forests along with multiple-instance learning is employed. Multiple-instance learning is applied to consecutive edits by the same user on the same item, so-called editing sessions. Typically, editing sessions are closely related, so that multiple-instance learning has a significant positive effect on the classification performance.

The second-most effective approach has been deployed within Wikimedia's Objective Revision Evaluation Service, ORES [19]. ORES operationalizes 14 features (see Table 3), most of which were introduced with WDVD, since the WDVD developers shared with Wikimedia a detailed feature list. Meanwhile, certain features have been discarded from WDVD due to overfitting but are still

[1] See http://www.wsdm-cup-2017.org/vandalism-detection.html
[2] https://www.wikidata.org/wiki/Wikidata:Vandalism

[3] These include the groups sysop, checkuser, flood, ipblock-exempt, oversight, property-creator, rollbacker, steward, sysop, translationadmin, and wikidata-staff.

found in the ORES system. Altogether, the effectiveness reported by Sarabadani et al. is significantly worse compared with WDVD [7].

Tan et al. [20] developed a classifier to detect low-quality contributions to Freebase. The only content-based features are the predicates of subject-predicate-object triples, which are used to predict low-quality contributions. Regarding context-based features, the developers employ user history features including numbers of past edits (total, correct, incorrect) and the age of the user account. Also user expertise is captured by representing users in a topic space based on their past contributions: a new contribution is mapped into the topic space and compared to the user's past revisions using the dot product, the cosine similarity, and the Jaccard index. None of the existing approaches has evaluated so far user expertise features for Wikidata.

The three approaches above build upon previous work on *Wikipedia* vandalism detection, whereas the first machine learning-based approach for this task was proposed by Potthast et al. [15]. It was based on 16 features, primarily focusing on content, detecting nonsense character sequences and vulgarity. In subsequent work, and as a result of two shared task competitions at PAN 2010 and PAN 2011 [14, 16], the original feature set was substantially extended; Adler et al. [2] integrated many of them. From the large set of approaches that have been proposed since then, those of Wang and McKeown [24] and Ramaswamy et al. [18] stick out: they are based on search engines to check the correctness of Wikipedia edits, achieving a better performance than previous approaches. On the downside, their approaches have only been evaluated on a small dataset and cannot be easily scaled-up to hundreds of millions of edits. To improve the context-based analysis it was proposed to measure user reputation [1] as well as spatio-temporal user information [25]. Again, Kumar et al. [11] stick out since they do not try to detect damaging edits but vandalizing users via their behavior. Many of these features have been transferred to Wikidata vandalism detection; however, more work will be necessary to achieve detection performance comparable to Wikipedia vandalism detectors.

3. EVALUATION FRAMEWORK

This section introduces the knowledge base Wikidata in brief,[4] the Wikidata vandalism corpus derived from it, the evaluation platform, the used performance measures, as well as the baselines.

3.1 Wikidata

Wikidata is organized around items. Each item describes a coherent concept from the real world, such as a person, a city, an event, etc. An item in turn can be divided into an item head and an item body. The item head consists of human-readable labels, descriptions, and aliases, provided for up to 375 supported language codes. The item body consists of structured statements, such as the date of birth of a person, as well as sitelinks to Wikipedia pages that cover the same topic as the item. Each time a user edits an item, a new revision is created within the item's revision history. We refer to consecutive revisions from the same user on the same item as an "editing session".

3.2 Wikidata Vandalism Corpus 2016

For the shared task we built the Wikidata Vandalism Corpus 2016 (short: WDVC-2016[5]), which is an updated version of the WDVC-2015 corpus [6]. The corpus consists of 82 million user-contributed revisions made between October 2012 to June 2016 (excluding revisions from bots) alongside 198,147 vandalism annotations on those revisions that have been reverted via the administrative rollback feature; the feature is employed at Wikidata with the purpose to revert vandalism and similarly damaging contributions. Moreover, our corpus provides meta information that is not readily available from Wikidata, such as geolocation data of all anonymous edits as well as Wikidata revision tags originating from both the Wikidata Abuse Filter and semi-automatic editing tools. Table 1 gives an overview of the corpus: participants of the shared task were provided training data and validation data, while the test data was held back until after the final evaluation.

[4]See https://www.wikidata.org/wiki/Help:Contents for a comprehensive overview.

[5]Corpus available at http://www.wsdm-cup-2017.org/vandalism-detection.html.

Table 1: Datasets for training, validation, and test in terms of time period covered, vandalism revisions, total revisions, sessions, items, and users. Numbers are given in thousands.

Dataset	From	To	Vand.	Rev.	Sessions	Items	Users
Training	Oct 1, 2012	Feb 29, 2016	176	65,010	36,552	12,401	471
Validation	Mar 1, 2016	Apr 30, 2016	11	7,225	3,827	3,116	43
Test	May 1, 2016	Jun 30, 2016	11	10,445	3,122	2,661	41

Table 2: The Wikidata Vandalism Corpus WDVC-2016 in terms of total unique users, items, sessions, and revisions with a breakdown by item part and by vandal(ism) status (Vand.). Numbers are given in thousands.

	Entire corpus			Item head			Item body		
	Total	Vand.	Regular	Total	Vand.	Regular	Total	Vand.	Regular
Revisions	82,680	198	82,482	16,602	100	16,502	59,699	92	59,606
Sessions	43,254	119	43,142	9,835	71	9,765	33,955	49	33,908
Items	14,049	85	14,049	6,268	54	6,254	12,744	39	12,741
Users	518	96	431	247	65	186	310	35	279

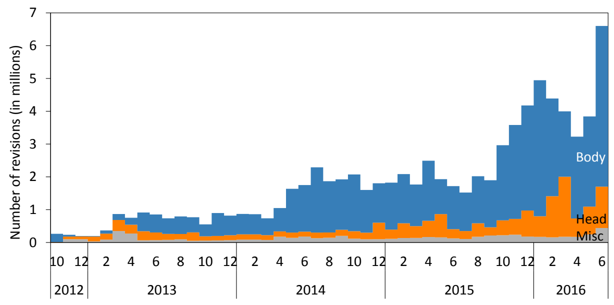
Table 2 gives an overview about the corpus in terms of content type (head vs. body) as well as revisions, sessions, items, and users. Figure 1 plots the development of the corpus over time. While the number of revisions per month is increasing (top), the number of vandalism revisions per month varies without a clear trend (bottom). We attribute the observed variations to the fast pace at which Wikidata is developed, both in terms of data acquisition and frontend development. For example, the drop in vandalism affecting item heads around April 2015 is probably related to the redesign of Wikidata's user interface around this time: with the new user interface it is less obvious to edit labels, descriptions, and aliases which might deter many drive-by vandals.[6]

3.3 Evaluation Platform

Our evaluation platform has been built to ensure (1) reproducibility of results, (2) blind evaluation, (3) ground truth protection, and, (4) to implement a realistic scenario of vandalism detection where revisions are scored in near real-time as they arrive. Reproducibility is ensured by inviting participants to submit their software instead of just its run output, employing the evaluation-as-a-service platform TIRA [17]. TIRA implements a cloud-based evaluation system, where each participant gets assigned a virtual machine in which a working version of their approach is to be deployed. The virtual machines along with the deployed softwares are remote-controlled via TIRA's web interface. Once participants manage to run their software on the test dataset hosted on TIRA, the virtual machine can be archived for later re-evaluation. In addition, TIRA ensures that participants do not get direct access to the test datasets, giving rise to blind evaluation, where the to-be-evaluated software's authors have never observed the test datasets directly.

For following reasons the task of vandalism detection in Wikis is intrinsically difficult to be organized as a shared task: the ground truth is publicly available via Wikimedia's data dumps, and the van-

[6]https://lists.wikimedia.org/pipermail/wikidata/2015-March/005703.html



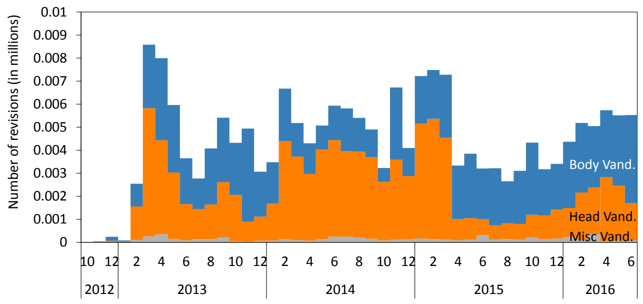


Figure 1: Overview of the Wikidata Vandalism Corpus 2016 by content type: Number of *all* revisions by content type (top) and number of *vandalism* revisions by content type (bottom).

dalism occurring at some point in the revision history of a Wikidata item will eventually be undone manually via the rollback facility, which we also use to create our ground truth. Handing out a test dataset spanning a long time frame would therefore effectively reveal the ground truth to participants. At the same time, in practice, it would be rather an unusual task to be handed a large set of consecutive revisions to be classified as vandalism or no, when in fact, every revision should be reviewed as it arrives. We therefore opted for a streaming-based setup where the participant software was supposed to connect to a data server,[7] which initiates a stream of revisions in chronological order (by increasing revision IDs), but waits sending new ones unless the software has returned vandalism scores for those already sent. This way, a software cannot easily exploit information gleaned from revisions occurring after a to-be-classified revision. However, we did not expect synchronous classification, but allowed for a backpressure window of $k = 16$ revisions, so that revision $i + k$ is sent as soon as the score for i-th revision has been reported. This allows for concurrent processing of data and the analysis, while preventing a deep look-ahead into the future of a revision. At $k = 16$, no more than 0.003% of the ground truth was revealed within the backpressure window (283 regular revisions and 76 vandalism revisions). However, as reported by participants, vandalism scores were based on previous revisions, while the backpressure window was unanimously used to gain computational speedup, and not to exploit ground truth or editing sessions.

3.4 Performance Measures

We employ the same performance measures as in our previous work [7]: area under curve of the receiver operating characteristic, ROC_{AUC}, and area under the precision-recall curve, PR_{AUC}. Both ROC_{AUC} and PR_{AUC} cover a wide range of operating points, emphasizing different characteristics. Given the large class imbalance of vandalism to non-vandalism, ROC_{AUC} emphasizes performance in the high recall range, while PR_{AUC} emphasizes performance in the high precision range. I.e., ROC_{AUC} is probably more meaningful for a semi-automatic operation of a vandalism detector, pre-filtering revisions likely to be vandalism and leaving the final judgment to human reviewers. PR_{AUC} addresses scenarios where vandalism shall be reverted fully automatically without any human intervention. The winner of the WSDM Cup was determined based on ROC_{AUC}.

For informational purposes, we also report the typical classifier performance measures for the operating point at the threshold 0.5: accuracy (ACC), precision (P), recall (R), F_1 measure (F).

3.5 Baselines and Meta Approach

WDVD. We employ the Wikidata Vandalism Detector, WDVD[7], as (strong) state-of-the-art baseline. The underlying model consists of 47 features and employs multiple-instance learning on top of bagging and random forests. The model was trained on training data ranging from May 1, 2013, to April 30, 2016. We used the same hyperparameters for this model as reported in our previous work [7]: 16 random forests, each build on 1/16 of the training dataset with the forests consisting of 8 trees, each having a maximal depth of 32 with two features per split and using the default Gini split criterion. In order to adjust WDVD to the new evaluation setup where revisions arrive constantly in a stream, we adjusted the multiple-instance learning to consider only those revisions of a session up until the current revision.

FILTER. As a second baseline, we employ so-called revision tags, which are created on Wikidata due to two main mechanisms: (1) The Wikidata Abuse Filter automatically tags revisions according to a collection of human-generated rules. (2) Revisions created by semi-automatic editing tools such as the Wikidata Game are tagged with the authentication method used by the semi-automatic editing tool. In general, tags assigned by the abuse filter are a strong signal for vandalism while tags from semi-automatic editing tools are a signal for non-vandalism. We trained a random forest with scikit-learn's default hyperparameters on the training data from May 1, 2013 to April 30, 2016. The revision tags were provided to all participants as part of the meta data. This baseline has also been incorporated into WDVD as feature `revisionTags`.

ORES. We reimplemented the ORES approach [19] developed for Wikidata vandalism detection by the Wikimedia Foundation and we apply it to the Wikidata Vandalism Corpus 2016.[8] Essentially, this approach consists of a subset of WDVD's features plus some additional features that were previously found to overfit [7]. It uses a random forest and was trained on the training data ranging from May 1, 2013 to April 30, 2016. We use the original hyperparameters by Sarabadani et al.:[9] 80 decision trees considering 'log2' features per split using the 'entropy' criterion.[10] While Sarabadani et al. [19] experimented with balancing the weights of the training examples, we do not do so for the ORES baseline since it has no effect on performance in terms of ROC_{AUC} and decreases performance in terms of PR_{AUC}.

META. Given the vandalism scores returned by participant approaches and our baselines, the question arises what the detection performance would be if all these approaches were combined into one. To get an estimation of the possible performance, we employ a simple meta approach whose score for each to-be-classified revision corresponds to the mean of all 8 approaches. As it turns out, the meta approach slightly outperforms the other approaches.

[7] https://github.com/wsdm-cup-2017/wsdmcup17-data-server

[8] Compared with our previous paper [7], we employ an updated version of ORES that was recently published by Sarabadani et al. [19].

[9] https://github.com/wiki-ai/wb-vandalism/blob/sample_subsets/Makefile

[10] The difference in performance of our reimplementation of ORES compared with Sarabadani et al. [19] is explained by the different datasets and evaluation metrics: while we split the dataset by time, Sarabadani et al. split the dataset randomly, causing revisions from the same editing session to appear both in the training as well as test dataset.

Table 3: Overview of feature groups, features, and their usage by WSDM Cup participants. Features that were newly introduced by participants and not previously used as part of WDVD are marked with an astersik (*). Features computed in the same way as for WDVD as well as new features are marked with ✓, features computed similarly to WDVD are marked with (✓), features for which it is unclear from their respective paper whether they are included are marked with ?, and features not utilized are marked with –.

	Feature group	Feature	Submitted Approach					Baseline		
			Buffaloberry [3]	Conkerberry [5]	Honeyberry [26]	Loganberry [28]	Riberry [27]	WDVD [7]	FILTER [7]	ORES [19]
Content features	Character features	`lowerCaseRatio`	✓	–	✓	✓	✓	✓	–	–
		`upperCaseRatio`	✓	–	✓	✓	✓	✓	–	–
		`nonLatinRatio`	✓	–	✓	✓	✓	✓	–	–
		`latinRatio`	✓	–	✓	✓	✓	✓	–	–
		`alphanumericRatio`	✓	–	✓	✓	✓	✓	–	–
		`digitRatio`	✓	–	✓	✓	✓	✓	–	–
		`punctuationRatio`	✓	–	✓	✓	✓	✓	–	–
		`whitespaceRatio`	✓	–	✓	✓	✓	✓	–	–
		`longestCharacterSequence`	✓	–	✓	✓	✓	✓	–	–
		`asciiRatio`	–	–	–	✓	✓	✓	–	–
		`bracketRatio`	✓	–	–	✓	✓	✓	–	–
		misc features from WDVD	–	–	–	–	✓	–	–	–
		`symbolRatio` *	✓	–	–	–	–	–	–	–
		`mainAlphabet` *	✓	–	–	–	–	–	–	–
	Word features	`languageWordRatio`	✓	–	✓	✓	✓	✓	–	–
		`containsLanguageWord`	✓	–	✓	✓	✓	✓	–	–
		`lowerCaseWordRatio`	✓	–	✓	✓	✓	✓	–	–
		`longestWord`	✓	–	✓	✓	✓	✓	–	–
		`containsURL`	✓	–	✓	✓	✓	✓	–	–
		`badWordRatio`	✓	–	–	✓	✓	✓	–	–
		`proportionOfQidAdded`	–	–	–	?	–	✓	–	✓
		`upperCaseWordRatio`	✓	–	✓	✓	✓	✓	–	–
		`proportionOfLinksAdded`	–	–	–	?	–	✓	–	✓
		`proportionOfLanguageAdded`	–	–	–	–	–	–	–	✓
		misc features from WDVD	–	–	–	–	✓	–	–	–
		`bagOfWords` *	–	✓	–	–	–	–	–	–
	Sentence features	`commentTailLength`	✓	–	–	✓	✓	✓	–	–
		`commentSitelinkSimilarity`	(✓)	–	–	✓	✓	✓	–	–
		`commentLabelSimilarity`	(✓)	–	–	✓	✓	✓	–	–
		`commentCommentSimilarity`	–	–	–	?	–	✓	–	–
		`languageMatchProb` *	✓	–	–	–	–	–	–	–
		`hasIdentifierChanged`	–	–	–	–	–	–	–	✓
	Statement features	`propertyFrequency`	–	(✓)	–	✓	✓	✓	–	(✓)
		`itemValueFrequency`	–	(✓)	–	✓	✓	✓	–	–
		`literalValueFrequency`	–	(✓)	–	✓	✓	✓	–	–
Context features	User features	`userCountry`	✓	✓	✓	–	✓	✓	–	–
		`userTimeZone`	✓	✓	✓	–	✓	✓	–	–
		`userCity`	✓	✓	✓	–	✓	✓	–	–
		`userCounty`	✓	✓	✓	–	✓	✓	–	–
		`userRegion`	✓	✓	✓	–	✓	✓	–	–
		`cumUserUniqueItems`	–	–	–	–	✓	✓	–	–
		`userContinent`	✓	✓	✓	–	✓	✓	–	–
		`isRegisteredUser`	✓	✓	✓	✓	✓	✓	–	✓
		`userFrequency`	✓	✓	–	–	✓	✓	–	–
		`isPrivilegedUser`	✓	–	–	✓	✓	✓	–	(✓)
		misc features from WDVD	–	–	–	–	✓	–	–	–
		`userIPSubnets` *	–	✓	–	–	–	–	–	–
		`userVandalismFraction` *	–	–	–	✓	✓	–	–	–
		`userVandalismCount` *	–	–	–	✓	–	–	–	–
		`userUniqueItems` *	–	–	–	–	✓	–	–	–
		`userAge`	–	–	–	–	–	–	–	✓
	Item features	`logCumItemUniqueUsers`	–	–	–	–	–	✓	–	–
		`logItemFrequency`	–	–	–	–	–	✓	–	–
		`isHuman`	–	–	–	–	–	–	–	✓
		`isLivingPerson`	–	–	–	–	–	–	–	✓
		misc features from WDVD	–	–	–	–	✓	–	–	–
		`itemFrequency` *	✓	(✓)	–	–	✓	–	–	–
		`itemVandalismFraction` *	–	–	–	✓	✓	–	–	–
		`itemVandalismCount` *	–	–	–	✓	–	–	–	–
		`itemUniqueUsers` *	–	–	–	–	✓	–	–	–
	Revision features	`revisionTags`	(✓)	✓	✓	?	✓	✓	✓	–
		`revisionLanguage`	(✓)	✓	✓	✓	✓	✓	–	–
		`revisionAction`	✓	✓	✓	✓	✓	✓	–	(✓)
		`commentLength`	–	–	✓	✓	✓	✓	–	–
		`isLatinLanguage`	–	–	✓	✓	✓	✓	–	–
		`revisionPrevAction`	–	–	–	?	–	✓	–	–
		`revisionSubaction`	✓	✓	✓	✓	✓	✓	–	(✓)
		`positionWithinSession`	–	–	–	?	–	✓	–	–
		`numberOfIdentifiersChanged`	–	–	–	–	–	–	–	✓
		misc features from WDVD	–	–	–	–	✓	–	–	–
		`isMinorRevision` *	✓	–	–	–	✓	–	–	–
		`changeCount` *	✓	✓	–	–	✓	–	–	(✓)
		`superItem` *	✓	–	–	–	✓	–	–	–
		`revisionSize` *	✓	–	–	–	✓	–	–	–
		`hourOfDay` *	✓	–	–	–	–	–	–	–
		`dayOfWeek` *	–	–	–	–	✓	–	–	–
		`revisionPrevUser` *	✓	–	–	–	–	–	–	–
		`hashTag` *	–	✓	(✓)	–	–	–	–	–
		`isSpecialRevision` *	–	–	✓	–	–	–	–	–

4. SURVEY OF SUBMISSIONS

This section surveys the features and learning algorithms employed by the participants. All of them chose to build their own model—which eventually are based on our WDVD approach [7], whose code base had been published to ensure reproducibility. On the one hand, the availability of this code base leveled the playing field among participants since it enabled everyone to achieve at least state-of-the-art performance. On the other hand, the availability may have stifled creativity and innovation among participants since all approaches follow a similar direction, and no one investigated different classification paradigms. However, all participants attempted to improve over our approach (which was one of the baselines) by developing new features and experimenting with learning variants.

4.1 Features

Table 3 gives a comprehensive overview of the features used in the submitted approaches. The feature set is complete; in particular, it unifies those features that are the same or closely related across participants. The table divides the features into two main groups, content features and context features. The content features in turn are subdivided regarding the granularity level at which characteristics are quantified, whereas the context features are subdivided regarding contextual entities in connection with a to-be-classified revision. Since the features have been extensively described in our previous work and the participant notebooks we omit a detailed description here. Instead, the feature names have been chosen to convey their intended semantics and are in accordance with the corresponding implementations found in our code base. For in-depth information we refer to our paper covering WDVD [7], the FILTER baseline, our reimplementation of ORES, as well as the notebook papers submitted by the participants [3, 5, 26, 27, 28].

We would like to point out certain observations that can be gained from the overview: Buffaloberry [3] used many of the WDVD features but also contributed a number of additional features on top of that. Conkerberry [5] used an interesting bag of words model that basically consists of the feature values computed by many of the WDVD features, all taken as words. Loganberry [28] did not exploit the information we provided as meta data, such as geolocation, etc. With two exceptions, Honeyberry [26] used almost exclusively WDVD features. Riberry [27] used on top of the WDVD features those that we previously found to overfit (denoted as "misc. features from WDVD" in the table), which may explain their poor overall performance, corroborating our previous results.

4.2 Learning Algorithms

Table 4 overviews the employed learning algorithms and organizes them wrt. achieved performance. The best-performing approach by Buffaloberry employs XGBoost and multiple-instance learning. The second-best approach by Conkerberry employs a linear SVM, encoding all WDVD features as a bag of words model. This results in an effectiveness comparable to the WDVD baseline in terms of $\mathrm{ROC}_{\mathrm{AUC}}$, but not in terms of $\mathrm{PR}_{\mathrm{AUC}}$. The third approach by Loganberry also employs XGBoost, however, in contrast to the first approach, no multiple-instance learning was conducted. The fourth approach, Honeyberry, created an ensemble of various algorithms following a stacking strategy. In contrast to the first approach, the authors put less emphasis on feature engineering. Their final submission contained a bug reducing the performance of their approach, which was fixed only after the submission deadline. The bugfix caused their performance to jump to an $\mathrm{ROC}_{\mathrm{AUC}}$ of 0.928, thus virtually achieving the third place in the competition. The fifth approach of Riberry performed poorly, probably due to overfitting features. The baselines employ a parameter-optimized random forest.

Table 4: Overview of the employed learning algorithms per submission. The rows are sorted wrt. the achieved evaluation scores, starting with the best.

Submission	Learning Algorithms							
	XGBoost	Linear SVM	Logistic Regression	Random Forest	Extra Trees	GBT	Neural Networks	Multiple-Instance
META	✓	✓	✓	✓	✓	✓	✓	✓
Buffaloberry	✓	–	–	–	–	–	–	✓
Conkerberry	–	✓	–	–	–	–	–	–
WDVD (baseline)	–	–	–	✓	–	–	–	✓
Honeyberry	–	–	✓	✓	✓	✓	✓	–
Loganberry	✓	–	–	–	–	–	–	–
Riberry	–	–	–	✓	–	✓	–	–
ORES (baseline)	–	–	–	✓	–	–	–	–
FILTER (baseline)	–	–	–	✓	–	–	–	–

5. EVALUATION

This section presents an in-depth evaluation of the submitted approaches, including overall performance and performance achieved regarding different data subsets, such as head content vs. body content, registered users vs. unregistered users, and performance variation over time. Furthermore, we combine the five submitted approaches within an ensemble to get a first idea about the performance that could be achieved if these approaches were integrated.

5.1 Official Competition Ranking

The competition phase of the WSDM Cup 2017 officially ended on December 30, 2016, resulting in the following ranking:

1. Buffaloberry by Crescenzi et al. [3]
2. Conkerberry by Grigorev [5]
3. Loganberry by Zhu et al. [28]
4. Honeyberry by Yamazaki et al. [26]
5. Riberry by Yu et al. [27]

We congratulate the winners! Working versions of these approaches have been successfully deployed within TIRA and evaluated using our aforementioned evaluation framework; three participants also shared their code bases as open source.[11]

To provide a realistic overview of the state of the art at the time of writing, we report the results that were most recently achieved. This is particularly relevant for the authors of the Honeyberry vandalism detector, who found and fixed an error in their approach shortly after the deadline, moving their approach up one rank. Moreover, we include the performances achieved by our own approach as well as that of our two baselines and of the meta approach. The following ranking hence slightly differs from the official one.

5.2 Evaluation Results

Table 5 and Figure 2 give an overview of the evaluation results of the vandalism detection task at WSDM Cup 2017.

5.2.1 Overall Performance

The evaluation results shown in Table 5 are ordered by $\mathrm{ROC}_{\mathrm{AUC}}$. The meta approach ($\mathrm{ROC}_{\mathrm{AUC}}$ 0.950) outperforms all other approaches, followed by the winner Buffaloberry ($\mathrm{ROC}_{\mathrm{AUC}}$ 0.947). The least effective approach in terms of $\mathrm{ROC}_{\mathrm{AUC}}$ is the FILTER

[11] https://github.com/wsdm-cup-2017

Table 5: Evaluation results of the WSDM Cup 2017 on the test dataset. Performance values are reported in terms of accuracy (Acc), precision (P), recall (R), F-measure, area under the precision-recall curve (PR_{AUC}), and area under curve of the receiver operating characteristic (ROC_{AUC}), as well as with regard to the four data subsets. The darker a cell, the better the performance.

Approach	Overall performance						Item head		Item body		Registered user		Unregistered user	
	Acc	P	R	F	PR_{AUC}	ROC_{AUC}	PR_{AUC}	ROC_{AUC}	PR_{AUC}	ROC_{AUC}	PR_{AUC}	ROC_{AUC}	PR_{AUC}	ROC_{AUC}
META	0.9991	0.668	0.339	0.450	0.475	0.950	0.648	0.996	0.387	0.926	0.082	0.829	0.627	0.944
Buffaloberry	0.9991	0.682	0.264	0.380	0.458	0.947	0.634	0.997	0.364	0.921	0.053	0.820	0.613	0.938
Conkerberry	0.9990	0.675	0.099	0.173	0.352	0.937	0.512	0.989	0.281	0.911	0.004	0.789	0.538	0.915
WDVD (baseline)	0.9991	0.779	0.147	0.248	0.486	0.932	0.668	0.996	0.388	0.900	0.086	0.767	0.641	0.943
Honeyberry	0.7778	0.004	0.854	0.008	0.206	0.928	0.364	0.993	0.101	0.893	0.002	0.760	0.308	0.819
Loganberry	0.9285	0.011	0.767	0.022	0.337	0.920	0.429	0.961	0.289	0.892	0.020	0.758	0.487	0.895
Riberry	0.9950	0.103	0.483	0.170	0.174	0.894	0.328	0.932	0.113	0.878	0.002	0.771	0.378	0.795
ORES (baseline)	0.9990	0.577	0.199	0.296	0.347	0.884	0.448	0.973	0.298	0.836	0.026	0.627	0.481	0.897
FILTER (baseline)	0.9990	0.664	0.073	0.131	0.227	0.869	0.249	0.908	0.182	0.840	0.021	0.644	0.387	0.771

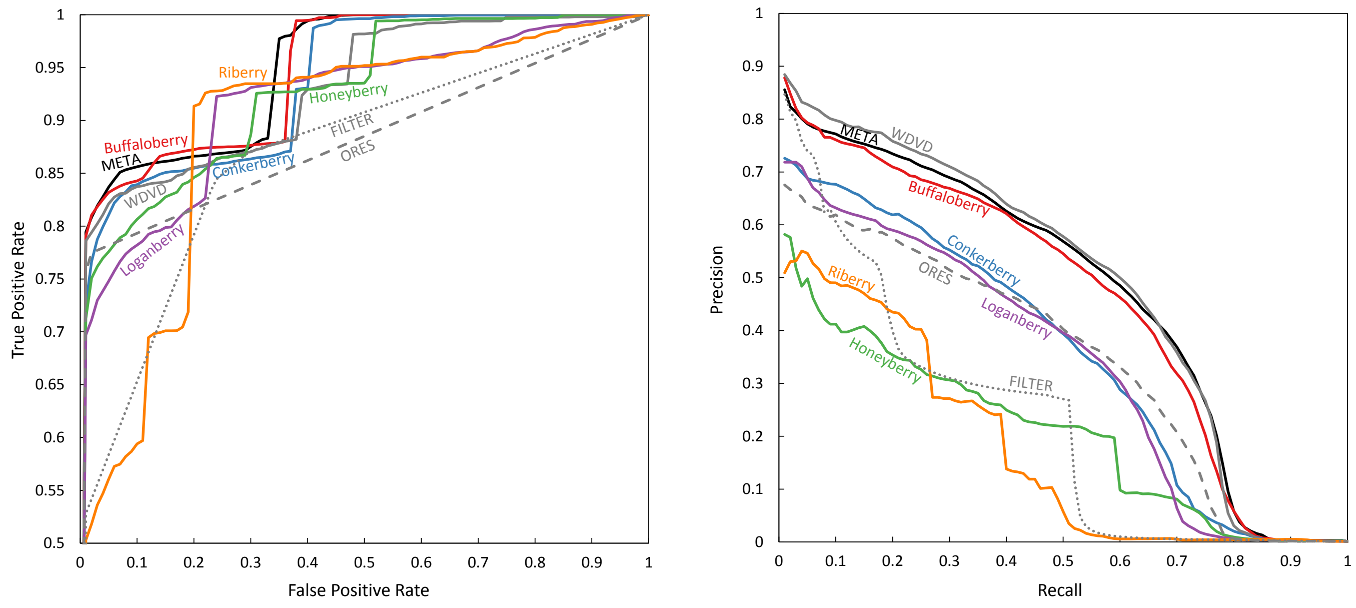


Figure 2: ROC and precision recall curves approaches.

baseline, achieving 0.869. Note that in our previous work [7] we reported higher values for ROC_{AUC}, which, however, were obtained with a previous version of the dataset. We discuss the apparent differences below.

The PR_{AUC} scores are lower and more diverse than the ROC_{AUC} scores, which is a consequence of the extreme class imbalance. The WDVD baseline outperforms all approaches in terms of PR_{AUC}, including the meta classifier; the ORES baseline outperforms all but two participants. The ranking among the participants changes only slightly: Loganberry and Honeyberry switch places. The main reason for high PR_{AUC} scores are features that can signal vandalism with a pretty high precision. For example, WDVD, Buffaloberry, Conkerberry, ORES, and Loganbery are all able to pick up on bad words (`badWordRatio` and `bagOfWords`) or contain detailed user information (`userFrequency`, `userAge`, `userVandalismFraction`, `userVandalismCount`), whereas FILTER and Honeyberry lack the respective features. The performance of Riberry can be explained by the inclusion of features that have previously been found to overfit [7].

While PR_{AUC} and ROC_{AUC} are computed on continuous scores, we also computed accuracy, precision, recall, and F-measure on binary scores at a threshold of 0.5. Honeyberry and Loganberry achieve poorest precision but highest recall. Conkerberry and FILTER achieve poorest recall but high precision. The META approach and Buffaloberry manage the best trade-off in terms of the F-measure. WDVD achieves highest precision at a non-negligible recall. Accuracy correlates with precision due to the high class imbalance. Recall that, since the winner of the competition was determined based on ROC_{AUC}, the teams had only little incentive to optimize the other scores. For a real-world applications of classifiers it might be beneficial to calibrate the raw scores to represent more accurate probability estimates and to set the threshold depending on the use case, i.e., adjusting Acc, P, R, and the F-measure.

5.2.2 Head Content vs. Body Content

Table 5 contrasts the approaches' performances on different content types, namely, item heads vs. item bodies. Regardless of the metric, all approaches perform significantly better on item heads. We explain this by the fact that vandalism on item heads typically happens at the lexical level (and hence can be detected more easily), e.g., by inserting bad words or wrong capitalization, whereas vandalism on item bodies typically happens at the semantic level, e.g., by inserting wrong facts. In particular, the character and word features focus on textual content as is found in the item head, but there are not many features for structural content. Future work might focus on transferring techniques that are used for the Google knowledge vault [4] to Wikidata, such as link prediction techniques to check the

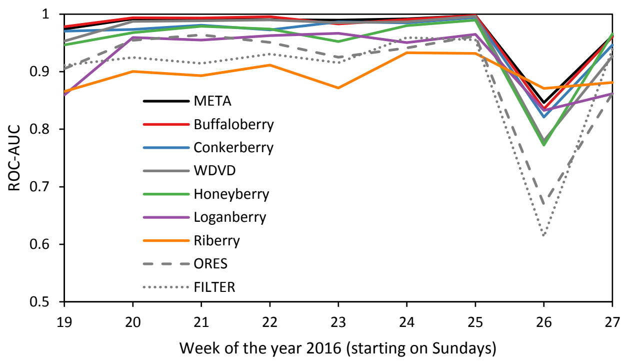


Figure 3: Performance over time on the test dataset.

correctness of links between items, and Gaussian mixture models to check the correctness of attribute values.

5.2.3 *Registered Users vs. Unregistered Users*

Table 5 also contrasts the approaches' performances regarding revisions originated from registered users vs. revisions from unregistered users. All approaches perform significantly better on revisions from unregistered users, which is in particular reflected by the $\mathrm{PR}_{\mathrm{AUC}}$ scores. Spot-checks suggest that vandalism of unregistered users appears to be rather obvious, whereas vandalism of registered users appears to be more sophisticated and elusive. Moreover, some reverted edits of registered users may also be considered honest mistakes. Telling apart honest mistakes from a (detected) vandalism attempt may be difficult.

5.2.4 *Performance over Time*

Figure 3 shows the performance of the approaches on the test set over time. Over the first seven weeks (calendar weeks 19 to 25) the approaches' performances remain relatively stable. All approaches were only trained with data obtained up until calendar week 18. Since no drop in the performances is observed, no major changes in the kinds of vandalism seem to have happened in this time frame. However, in calendar week 26 a major performance drop can be observed. The outlier is caused by a single, highly reputable user using some automatic editing tool on June 19, 2016, to create 1,287 slightly incorrect edits (which were rollback-reverted later). Since only 11,043 edits were labeled vandalism in the entire test set of two months, these 1,287 edits within a short time period have a significant impact on the overall performance.

We were curious to learn about the impact of this data artifact and recomputed the detection performances, leaving out the series of erroneous edits from the user in question. Table 6 shows the overall performance the approaches would have achieved then: while the absolute performance increases, the ranking among the participants is not affected.

Probably one cannot anticipate such an artifact in the test data, but, with hindsight, we consider it a blessing rather than a curse: it points to the important question of how to deal with such cases in practice. Machine learning-based vandalism detectors become unreliable when the characteristics of the stream of revisions to be classified suddenly changes—errors in both directions, false positives and false negatives, can be the consequence. Ideally, the developers of detectors envision possible exceptional circumstances and provide a kind of exception handling; e.g., flagging a user with suspicious behavior by default for review, regardless whether the respective series of edits is considered damaging or not.

Table 6: Evaluation results of the WSDM Cup 2017 on the test dataset without the erroneous edits.

Approach	**Overall performance**					
	Acc	P	R	F	$\mathrm{PR}_{\mathrm{AUC}}$	$\mathrm{ROC}_{\mathrm{AUC}}$
META	0.9992	0.668	0.384	0.487	0.536	0.988
Buffaloberry	0.9992	0.682	0.298	0.415	0.517	0.988
WDVD (baseline)	0.9992	0.779	0.167	0.274	0.548	0.980
Conkerberry	0.9991	0.675	0.113	0.193	0.398	0.980
Honeyberry	0.7779	0.004	0.967	0.008	0.233	0.972
Loganberry	0.9286	0.011	0.867	0.022	0.383	0.939
FILTER (baseline)	0.9991	0.664	0.082	0.146	0.257	0.938
ORES (baseline)	0.9991	0.577	0.225	0.324	0.392	0.935
Riberry	0.9951	0.103	0.546	0.173	0.196	0.902

6. REFLECTIONS ON THE WSDM CUP

The WSDM Cup 2017 had two tasks for which a total of 140 teams registered, 95 of which ticked the box for participation in the vandalism detection task (multiple selections allowed). This is a rather high number compared with other shared task events. We attribute this success to the facts that the WSDM conference is an A-ranked conference, giving the WSDM Cup a high visibility, that the vandalism detection task was featured on Slashdot,[12] and that we attracted sponsorship from Adobe, which allowed us to award cash prizes to the three winning participants of each task. However, only 35 registered participants actually engaged when being asked for their operating system preferences for their virtual machine on TIRA, 14 of which managed to produce at least one run, whereas the remainder never used their assigned virtual machines at all. In the end, five teams made a successful submission by running their software without errors on the test dataset.

Why did so many participants decided to drop out on this task? We believe that the comparably huge size of the dataset as well as difficulties in setting up their approach in our evaluation platform are part of the reason: each approach had to process gigabytes of data by implementing a client-server architecture, and all of that had to be deployed on a remote virtual machine. The requirement to submit working software, however, may not have been the very main cause since the retention rate of our companion task was much higher. Rather, the combination of dataset size, real-time client-server processing environment, and remote deployment is a likely cause. Note that the vandalism detection task itself demanded for this scale of operations, since otherwise it would have been easy to cheat, which is particularly a problem when cash prizes are involved. Finally, the provided baseline systems were already competitive, so that the failure to improve upon them may have caused additional dropouts.

The WSDM Cup taught us an important lesson about the opportunities and limitations of shared tasks in general and about evaluation platforms and rule enforcement in particular. On the one hand, competitions like ours are important to rally followers for a given task and to create standardized benchmarks. On the other hand, shared tasks are constrained to a relatively short period of time and create a competitive environment between teams. I.e., it becomes important to implement a good trade-off in the evaluation setup in order to prevent potential cheating and data leaks, while, at the same time, placing low hurdles on the submission procedure. A means towards this end might be standardized evaluation platforms that are widely used for a large number of shared tasks. While there are already platforms like TIRA or Kaggle, we are not aware of a widely used evaluation platform for time series data, serving teams with one test example after the other and providing a sandboxing mechanism

[12]https://developers.slashdot.org/story/16/09/10/1811237

of the submitted softwares to prevent data leaks. Moreover, there definitely is a trade-off between enforcing strict rules on the one side and scientific progress on the other. For example, only two teams had made a successful submission by the original deadline, while other teams were still struggling with running their approaches. In this case, we erred on the side of scientific progress by additional submissions in favor of overly strict rules and gave all teams a short deadline extension of 8 days, accepting some discussion about the deadline's extensions fairness.

7. CONCLUSION AND OUTLOOK

This paper gives an overview of the five vandalism detection approaches submitted to the WSDM Cup 2017. The approaches were evaluated on the new Wikidata Vandalism Corpus 2016, which has been specifically compiled for the competition. Under a semi-automatic detection scenario, where newly arriving revisions are ranked for manual review, the winning approach from Buffaloberry Crescenzi et al. [3] performs best. Under a fully automatic detection scenario, where the decision whether or not to revert a given revision is left with the classifier, the baseline approach WDVD by Heindorf et al. [7] still performs best. Combining all approaches within a meta classifier yields a small improvement; however, the feature set seems to be the performance-limiting factor.

All approaches build upon the WDVD baseline, proposing only few additional features. I.e., for the future it is interesting to develop and explore fundamentally different feature sets. E.g., building upon the work on knowledge graphs, technology for link prediction and value range prediction should be investigated. Building upon the work of other user-generated content, also psychologically motivated features capturing a user's personality and state of mind appear promising.

Acknowledgments

We thank Adobe for sponsoring the WSDM Cup, and Wikimedia Germany for supporting our task. Our special thanks go to all participants for their devoted work.

References

[1] B. Adler, L. de Alfaro, and I. Pye. Detecting Wikipedia Vandalism using WikiTrust. In *CLEF*, pages 22–23, 2010.

[2] B. T. Adler, L. de Alfaro, S. M. Mola-Velasco, P. Rosso, and A. G. West. Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features. In *CICLing*, pages 277–288, 2011.

[3] R. Crescenzi, M. Fernandez, F. A. G. Calabria, P. Albani, D. Tauziet, A. Baravalle, and A. S. D'Ambrosio. A Production Oriented Approach for Vandalism Detection in Wikidata—The Buffaloberry Vandalism Detector at WSDM Cup 2017. In *WSDM Cup*, 2017.

[4] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *KDD*, pages 601–610. ACM, 2014.

[5] A. Grigorev. Large-Scale Vandalism Detection with Linear Classifiers— The Conkerberry Vandalism Detector at WSDM Cup 2017. In *WSDM Cup*, 2017.

[6] S. Heindorf, M. Potthast, B. Stein, and G. Engels. Towards Vandalism Detection in Knowledge Bases: Corpus Construction and Analysis. In *SIGIR*, pages 831–834. ACM, 2015.

[7] S. Heindorf, M. Potthast, B. Stein, and G. Engels. Vandalism Detection in Wikidata. In *CIKM*, pages 327–336. ACM, 2016.

[8] S. Heindorf, M. Potthast, H. Bast, B. Buchhold, and E. Haussmann. WSDM Cup 2017: Vandalism Detection and Triple Scoring. In *WSDM*, pages 827–828. ACM, 2017.

[9] J. Kiesel, M. Potthast, M. Hagen, and B. Stein. Spatio-Temporal Analysis of Reverted Wikipedia Edits. In *ICWSM*, pages 122–131. AAAI Press, 2017.

[10] A. Kittur, B. Suh, B. A. Pendleton, and E. H. Chi. He Says, She Says: Conflict and Coordination in Wikipedia. In *CHI*, pages 453–462. ACM, 2007.

[11] S. Kumar, F. Spezzano, and V. S. Subrahmanian. VEWS: A Wikipedia Vandal Early Warning System. In *KDD*, pages 607–616. ACM, 2015.

[12] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From Freebase to Wikidata: The Great Migration. In *WWW*, pages 1419–1428, 2016.

[13] M. Potthast. Crowdsourcing a Wikipedia Vandalism Corpus. In *SIGIR*, pages 789–790, 2010.

[14] M. Potthast and T. Holfeld. Overview of the 2nd International Competition on Wikipedia Vandalism Detection. In *CLEF*, 2011.

[15] M. Potthast, B. Stein, and R. Gerling. Automatic Vandalism Detection in Wikipedia. In *ECIR*, pages 663–668, 2008.

[16] M. Potthast, B. Stein, and T. Holfeld. Overview of the 1st International Competition on Wikipedia Vandalism Detection. In *CLEF*, 2010.

[17] M. Potthast, T. Gollub, F. Rangel, P. Rosso, E. Stamatatos, and B. Stein. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In *CLEF*, pages 268–299. Springer, 2014.

[18] L. Ramaswamy, R. Tummalapenta, K. Li, and C. Pu. A Content-Context-Centric Approach for Detecting Vandalism in Wikipedia. In *COLLABORATECOM*, pages 115–122, 2013.

[19] A. Sarabadani, A. Halfaker, and D. Taraborelli. Building Automated Vandalism Detection Tools for Wikidata. In *WWW Companion*, pages 1647–1654, 2017.

[20] C. H. Tan, E. Agichtein, P. Ipeirotis, and E. Gabrilovich. Trust, but Verify: Predicting Contribution Quality for Knowledge Base Construction and Curation. In *WSDM*, pages 553–562. ACM, 2014.

[21] K. Tran and P. Christen. Cross Language Prediction of Vandalism on Wikipedia Using Article Views and Revisions. In *PAKDD (2)*, volume 7819 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 2013.

[22] K. Tran and P. Christen. Cross-Language Learning from Bots and Users to Detect Vandalism on Wikipedia. *IEEE Trans. Knowl. Data Eng.*, 27(3):673–685, 2015.

[23] K. Tran, P. Christen, S. Sanner, and L. Xie. Context-Aware Detection of Sneaky Vandalism on Wikipedia Across Multiple Languages. In *PAKDD (1)*, volume 9077 of *Lecture Notes in Computer Science*, pages 380–391. Springer, 2015.

[24] W. Y. Wang and K. R. McKeown. "Got You!": Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-semantic Modeling. In *COLING*, pages 1146–1154, 2010.

[25] A. G. West, S. Kannan, and I. Lee. Detecting Wikipedia Vandalism via Spatio-temporal Analysis of Revision Metadata. In *EUROSEC*, pages 22–28, 2010.

[26] T. Yamazaki, M. Sasaki, N. Murakami, T. Makabe, and H. Iwasawa. Ensemble Models for Detecting Wikidata Vandalism with Stacking—Team Honeyberry Vandalism Detector at WSDM Cup 2017. In *WSDM Cup*, 2017.

[27] T. Yu, Y. Zhao, X. Wang, Y. Xu, H. Shao, Y. Wang, X. Ma, and D. Dey. Vandalism Detection Midpoint Report—The Riberry Vandalism Detector at WSDM Cup 2017. University of Illinois at Urbana-Champaign Student Report, not published, 2017.

[28] Q. Zhu, H. Ng, L. Liu, Z. Ji, B. Jiang, J. Shen, and H. Gui. Wikidata Vandalism Detection—The Loganberry Vandalism Detector at WSDM Cup 2017. In *WSDM Cup*, 2017.