Vandalism Detection in Wikidata

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

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

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

ABSTRACT

Wikidata is the new, large-scale knowledge base of the Wikimedia Foundation. Its knowledge is increasingly used within Wikipedia itself and various other kinds of information systems, imposing high demands on its integrity. Wikidata can be edited by anyone and, unfortunately, it frequently gets vandalized, exposing all information systems using it to the risk of spreading vandalized and falsified information. In this paper, we present a new machine learning-based approach to detect vandalism in Wikidata. We propose a set of 47 features that exploit both content and context information, and we report on 4 classifiers of increasing effectiveness tailored to this learning task. Our approach is evaluated on the recently published Wikidata Vandalism Corpus WDVC-2015 and it achieves an area under curve value of the receiver operating characteristic, ROC_{AUC}, of 0.991. It significantly outperforms the state of the art represented by the rule-based Wikidata Abuse Filter (0.865 ROC_{AUC}) and a prototypical vandalism detector recently introduced by Wikimedia within the Objective Revision Evaluation Service (0.859 ROC_{AUC}).

General Terms: Design, Experimentation, Evaluation
Keywords: Knowledge Base; Vandalism; Data Quality; Trust

1. INTRODUCTION

Crowdsourcing is a well-established paradigm to acquire knowledge. Many projects have emerged that invite their users to contribute, collect, and curate knowledge with the goal of building public knowledge bases. Famous examples include Wikipedia, StackExchange, Freebase, and Wikidata. Most of today's crowdsourced knowledge bases are unstructured, i.e., their content is encoded in natural language, rendering these resources hardly usable for machine processing and inference. Besides the (error-prone) automatic extraction from unstructured knowledge bases, a trend toward crowdsourcing structured knowledge directly can be observed: one of the most prominent examples used to be Freebase [5], which was recently shut down and is now superseded by Wikidata [28, 33], the knowledge base of the Wikimedia Foundation. Despite its recent launch in 2012, Wikidata has already become the largest structured crowdsourced knowledge base on the web.

Crowdsourcing knowledge at scale and in a reasonable time frame requires large crowds. This renders it impossible to rely only on experts, to vet every volunteer, or to manually review individual contributions, unless other volunteers do so. Inviting the crowd therefore means trusting them. Wikipedia and all other projects of the Wikimedia Foundation prove that the crowd can be trusted to build and maintain reliable knowledge bases under the freedom that anyone can edit anything. Despite the success of the freedom-to-edit model, all knowledge bases that rely on it are plagued by vandalism, namely "deliberate attempts to damage or compromise [their] integrity."[1] Vandalism has been around ever since knowledge crowdsourcing emerged. Its impact on *unstructured* knowledge bases can be severe, spreading false information or nonsense to anyone accessing a vandalized article. But given their importance to modern information systems, the impact of vandalism on *structured* knowledge bases can be disproportionately more so: integrating Wikidata into other information systems such as search engines and question-answering systems bears the risk of spreading vandalism to all their users. Reaching such a wide audience imposes significantly higher demands on the integrity of structured knowledge bases compared to unstructured ones.

Reviewing millions of contributions every month imposes a high workload on the community of a knowledge base. In this paper, we contribute a new machine learning-based approach for vandalism detection in Wikidata, thus freeing valuable time of volunteers and allowing them to focus their efforts on adding new content rather than on detecting and reverting damaging edits by vandals. We develop and carefully analyze features suitable for Wikidata, taking into account both content and context information of a Wikidata revision. On top of the features, we apply advanced machine-learning algorithms to obtain our final classifier: starting from a default random forest, we optimize its parameters, apply bagging, and for the first time multiple-instance learning, which exploits the dependence of consecutive edits by the same user on the same item (i.e., within an editing session). For evaluation, we use the Wikidata Vandalism Corpus WDVC-2015 [14], which comprises 24 million revisions. Our approach achieves an area under curve of the receiver operating characteristic (ROC_{AUC}) of 0.991 and therefore outperforms the state of the art in the form of the Wikidata Abuse Filter (0.865 ROC_{AUC}) and the recently introduced vandalism detector within Wikimedia's Objective Revision Evaluation Service, ORES (0.859 ROC_{AUC}).

In what follows, Section 2 reviews related work, Section 3 introduces our vandalism model for structured knowledge bases, Section 4 overviews the data set underlying our evaluation, and Section 5 reports on the optimization of our model as well as on its evaluation. Finally, Section 6 discusses practical implications.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2983323.2983740

[1] https://www.wikidata.org/wiki/Wikidata:Vandalism

2. RELATED WORK

This section reviews the state of the art related to detecting vandalism and low-quality contributions to (un)structured knowledge bases (e.g., Freebase, OpenStreetMap, Wikipedia, and StackOverflow).

2.1 Structured Knowledge Bases

To the best of our knowledge, there are only two papers to date that specifically address the detection of low-quality contributions to structured knowledge bases: Tan et al. [27] present a machine learning approach for Freebase and Neis et al. [22] present a rule-based approach for OpenStreetMap.

Tan et al.'s approach employs two categories of features: (1) content features for triples, and (2) context features for user history and user expertise. The content features basically consist of a single categorical feature, encoded as a Boolean feature vector, which denotes the predicate of the affected subject-predicate-object triple. Interestingly, Tan et al. show that this feature has a higher effectiveness than all context features combined. The user history features include numbers of past (total, correct, incorrect) edits, and the age of the user account. Expertise is captured by representing users in a topic space based on their past contributions: a new contribution is mapped into the topic space, computing its similarity to the user's past revisions using the dot product, the cosine similarity, and the Jaccard index, which serve as features. Our model incorporates Tan et al.'s best-performing features, but we omit the past number of "incorrect" edits (i.e., vandalism in our case) as well as the user expertise feature: we found the former to overfit and Tan et al.'s analysis shows the latter to perform worse than the triple features. Going beyond Tan et al. features, we include triple features for subject and object that were neglected, and we employ spatio-temporal features by geolocating a user's IP address.

Neis et al. [22] propose a rule-based approach to detect vandalism in OpenStreetMap. Their "vandalism score" incorporates content and context information measured as (1) edit reputation, and (2) user reputation, respectively. The edit reputation measure depends on the type of action (create, modify, delete) and a set of additional rules specific to OpenStreetMap (for example, whether a geographical object was moved more than 11 meters), and the user reputation measure is basically derived from the number of geographical objects a user has created. Our model, too, incorporates features to estimate user reputation, whereas the content rules cannot be transferred.

2.2 Unstructured Knowledge Bases

A different category of related work concerns the detection of low-quality contributions to unstructured knowledge bases. In particular, detecting vandalism in Wikipedia was studied previously. The first machine learning-based approach was proposed by Potthast et al. [24]. It was based on 16 features, primarily focusing on content, detecting nonsense character sequences and vulgarity. In subsequent work, the set of content-based features was extended [21], and context features were proposed to measure user reputation [1] and spatio-temporal user information [36]. Wu et al. [37] added text-stability features. Adler et al. [2] combined most of the aforementioned features into a single model. We have reviewed all the features employed for Wikipedia vandalism to date and transfer those applicable to structured knowledge bases into our model. Regarding classifiers, the most effective approach [2] uses random forests which we use as starting point for our optimizations. None of the authors have experimented with multiple-instance learning, which significantly improved our model's performance.

Besides traditional feature engineering approaches, Itakura and Clarke [16] propose to detect vandalism solely based on edit compressibility. Wang and McKeown [34] and Ramaswamy et al. [25] utilize search engines to check the correctness of Wikipedia edits, achieving a better performance than previous approaches. However, their approaches have only been evaluated on a small dataset, and do not seem to scale well to hundreds of millions of edits. All of the approaches described so far are tailored to the English Wikipedia, whereas West and Lee [35] propose a set of language-independent features, and Tran and Christen [29] attempt transfer learning between languages (see also [31, 30]). This seems to be particularly promising when training on one of Wikipedia's large language editions and applying the classifier on a small language edition. While most of the detection approaches focus on immediate vandalism detection once a new edit arrives, Adler et al. [2] also consider historic vandalism detection, taking into account information that emerges long after an edit was made. Wikidata comprises multilingual content, so that language-independent features are of interest for our model, whereas our focus is on immediate vandalism detection.

Besides Wikis, question and answer sites such as StackOverflow, Yahoo Answers, and Quora comprise unstructured knowledge, too. Agichtein et al. [3] were the first to develop machine learning features to distinguish high-quality from low-quality content on question and answer sites. Other approaches predict the quality of questions [18] and answers [26]. Most of the features used for these tasks are similar to those employed for Wikipedia vandalism detection; the remainder are specific to question and answer sites.

3. WIKIDATA VANDALISM MODEL

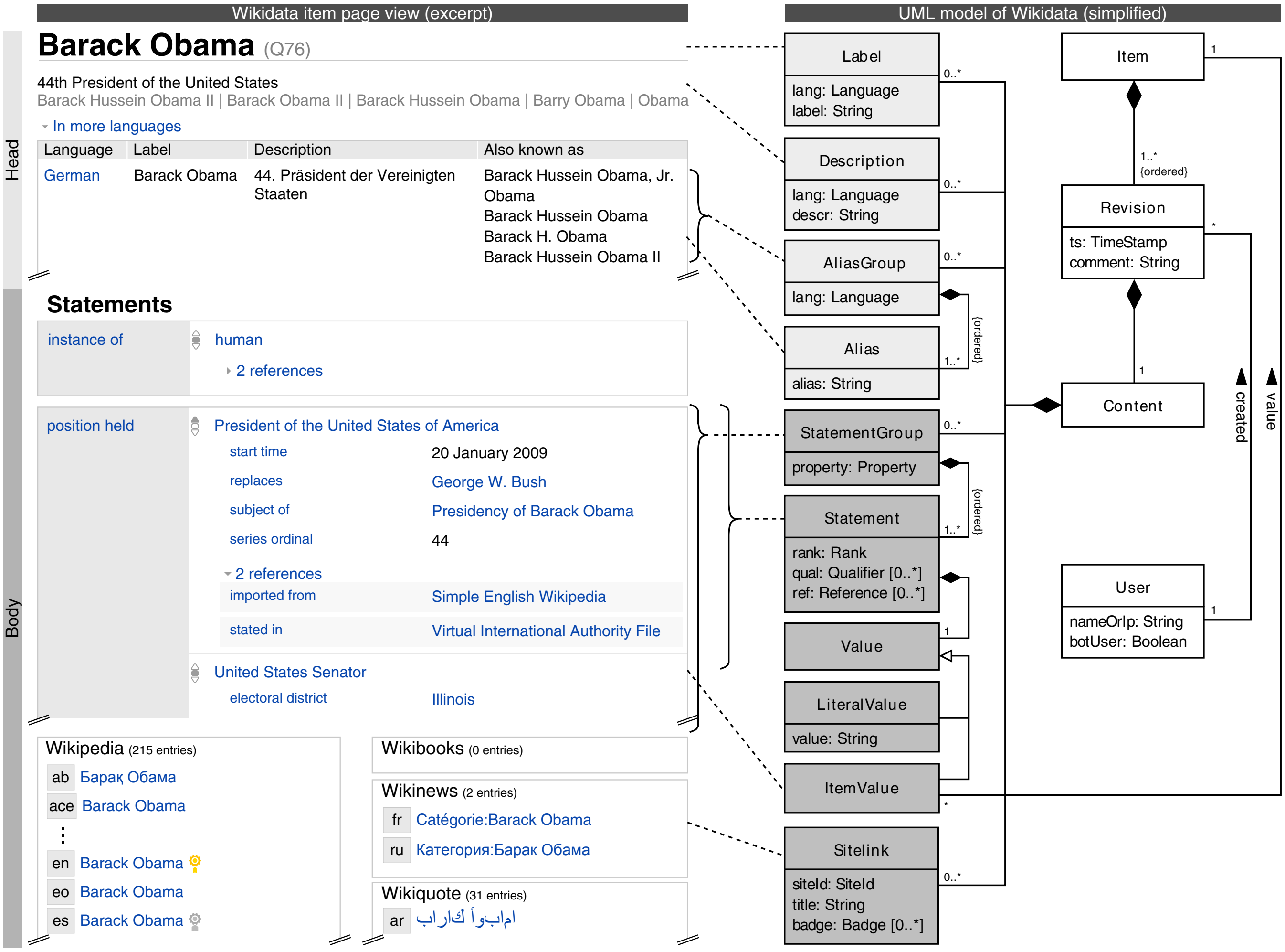
Our goal is to develop an effective vandalism detection model for Wikidata to determine whether newly arriving revisions are vandalism or not. Our model shall hint at vandalism across a wide range of precision-recall points to enable different use cases, e.g., fully automatic reversion of vandalism at high precision, but also prefiltering revisions in need of manual review at high recall.

Our model is based on a total of 47 features that quantify the characteristics of Wikidata vandalism, as well as Wikidata edits in general. Table 1 lists all features by category. We present features both for Wikidata's actual content as well as contextual meta-information. The features were developed and selected in a rigorous evaluation setup involving datasets for training, validation, and test, whereas the test dataset was only used once at the end. We also briefly report on features that failed to perform in our experiments and have hence been omitted from the final model. Before going into details, we briefly review Wikidata's terminology and its underlying data model as a prerequisite to understand many of our features.

3.1 Wikidata Terminology and Data Model

Figure 1 gives an overview of how Wikidata is constructed. At its core, Wikidata is organized around items. Each item represents one coherent concept, e.g., a person, a city, or a chemical element. The content of an item divides into head and body parts. The head of an item consists of labels, descriptions, and groups of aliases (i.e., variations on the label), all of which may be present in up to 291 supported languages. Although the fields allowed have a clear surface structure, the field values can be any plain text and they are not treated as structured statements.

The body of an item consists of statements and so-called sitelinks. Statements encode all of the structured knowledge of Wikidata, and they basically are subject-predicate-object triples, where, as per Wikidata's terminology, an *item* corresponds to the subject, a *property* to the predicate, and a *value* to the object. Property-value-pairs are called *claim*. Statements are grouped by property (e.g., "instance of" and "position held" in Figure 1). The set of supported properties is a controlled vocabulary, where additions and removals are discussed within the community; at the time of writing, Wikidata



Wikidata item page view (excerpt)

Head

Barack Obama (Q76)

44th President of the United States

Barack Hussein Obama II | Barack Obama II | Barack Hussein Obama | Barry Obama | Obama

In more languages

Language	Label	Description	Also known as
German	Barack Obama	44. Präsident der Vereinigten Staaten	Barack Hussein Obama, Jr. Obama Barack Hussein Obama Barack H. Obama Barack Hussein Obama II

Body

Statements

instance of	human	
	2 references	
position held	President of the United States of America	
	start time	20 January 2009
	replaces	George W. Bush
	subject of	Presidency of Barack Obama
	series ordinal	44
	2 references	
	imported from	Simple English Wikipedia
	stated in	Virtual International Authority File
	United States Senator	
	electoral district	Illinois

Wikipedia (215 entries)

ab	Барак Обама
ace	Barack Obama
⋮	
en	Barack Obama
eo	Barack Obama
es	Barack Obama

Wikibooks (0 entries)

Wikinews (2 entries)

fr	Catégorie:Barack Obama
ru	Категория:Барак Обама

Wikiquote (31 entries)

ar	باراك أوباما

Figure 1: Wikidata at a glance. Example of a Wikidata item page[2] with commonly found content elements (left), and Wikidata's data model as UML class diagram (right). Wikidata items divide into head (top, light gray classes) and body (bottom, dark gray classes).

supports 2126 properties. The value of a statement can be plain text, but more often it refers to another item (e.g., "human", "President of the United States of America", and "United States Senator" in Figure 1), thereby inducing a knowledge graph. Statements may have one of a three-valued rank: normal is the default, preferred should be assigned to statements that comprise the most current information, and deprecated to statements with erroneous or outdated information. To date, however, ranks other than normal are seldom used. A statement may contain supplementary information called *qualifiers* and *references*, where the former encode additional information about a statement, and the latter refer to its source. Both are essentially nested statements: for example, in Figure 1 the statement $s =$ ("Barack Obama", "position held", "President of the United States of America") is the subject of qualifier $q = (s,$ "start time", "20 January 2009"). The property vocabulary for qualifiers and references, and the range of possible values are the same as for statements, whereas some properties and values are more often used for qualifiers than others and vice versa. Lastly, a sitelink is a special kind of statement that refers to other Wikimedia sites which deal with the item in question. For display, sitelinks are grouped by site ID (e.g., "Wikipedia", "Wikibooks", etc. in Figure 1). Wikidata also shows site-specific badges that indicate outstanding articles (i.e., good articles get a silver badge , featured articles gold badge).

Items are edited by users, which are identified either by their user name or, in case of anonymous edits, by their IP address at the time of editing. Every submitted edit results in a new revision that is immediately displayed to end users and stored in the item's revision history. For each revision, a time stamp and a comment are recorded. In almost all cases, the comment is automatically generated to summarize the changes made as follows:

```
comment ::== <action> <subaction>? <param>+ <tail>,
```

where `<action>` is one of 24 actions of the Wikidata API (e.g., "set claim" or "set label"), `<subaction>?` is optionally one of "add", "update", and "remove", `<param>+` is a list of required parameters, and `<tail>` states the particular change made (e.g., "instance of (P31): human (Q5)" when creating a new claim, or "Barack Obama" when adding a label). As per Wikidata's web interface, a revision usually contains only one edited content element compared to its previous revision, rendering comments short and specific. The only exceptions to this rule include revisions that undo previous edits (i.e., called undo, restore, and rollback), and a small number of edits that have been submitted directly via the Wikidata API. As a result, the homogeneity and informativeness of comments allows for computing many features directly from them.

[2] Link to the item page at Wikidata: https://www.wikidata.org/wiki/Q76

3.2 Content Features

To quantify vandalism in an item's head, namely labels, descriptions, and aliases, we employ 24 features at the level of characters, words, and sentences (see Table 1).

Character level features. At character level, vandalism typically includes unexpected character sequences, such as random keystrokes, character repetitions, wrong or excessive capitalization, missing spaces, and the use of special characters (e.g., for emoticons). To quantify character usage, we compute the ratio of ten character classes to all characters within the comment tail of a revision, each serving as one feature: `upperCaseRatio`, `lowerCaseRatio`, `alphanumericRatio`, `digitRatio`, `punctuationRatio`, `bracketRatio`, `asciiRatio`, `whitespaceRatio`, `latinRatio`, `nonLatinRatio`.[3] In addition, the `longestCharacterSequence` of the same character serves as a feature. Features that did not make it into our final model include the ratio of character classes for specific alphabets, such as Arabic, Cyrillic, Han, and Hindi, as well as character n-grams for n in [2, 4] as bag-of-words feature, because their contribution to overall detection performance was negligible.

Word level features. At word level, vandalism typically includes wrong capitalization, URLs, profane or offensive words. Other damage originates from unaccustomed users who misunderstand the suggestion "enter a description in English" in Wikidata's web interface and submit the word "English." To quantify word usage, we compute the ratio of four word classes to all words within the comment tail of a revision, each serving as one feature: `lowerCaseWordRatio`, `upperCaseWordRatio`, `badWordRatio`, `languageWordRatio`. Regarding the former two, the ratio of words starting with a lower case or upper case letter are computed, respectively. Regarding the latter two, the `badWordRatio` is based on a dictionary of 1383 offensive English words [32], and the `languageWordRatio` on a list of regular expressions for language names and variations thereof [38]. Despite seeming redundant, incorporating the Boolean feature `containsLanguageWord` is beneficial. The Boolean feature `containsURL` checks for URLs using a regular expression, and one feature encodes the length of the `longestWord`. Furthermore, we include two word level features from the ORES baseline that compute the proportion of item IDs (i.e., Q-IDs) and that of links to all IDs and links present in the previous revision of an item: `proportionOfQidAdded` and `proportionOfLinksAdded`. These features encode whether adding a new item ID or a new URL is rather an exception than the rule. We further experimented with a basic bag-of-words model, but in conjunction with our other features, its contribution to overall detection performance was negligible.

Sentence level features. The comment tail can be considered a "sentence" and at this level, vandalism typically includes changes of suspicious lengths (encoded in `commentTailLength`), as well as the addition of labels, descriptions, and aliases unrelated to the current item. Given Wikidata's multilingual nature, we observe that labels and sitelinks often share similarities. To some extent, this even applies across languages, since named entities are often spelled similarly in families of languages that share an alphabet. We hence compute the Jaro-Winkler distance of the comment tail to the English label, to the English sitelink, and to the comment tail of the previous revision (i.e., `commentLabelSimilarity`, `commentSitelinkSimilarity`, and `commentCommentSimilarity`). The former two features quantify the similarity of new labels and sitelinks to those already present, and the latter feature quantifies the current revision's similarity to its predecessor. We refrained from using the entire comment tail as a categorical feature for its negligible contribution to overall detection performance.

Table 1: Overview of feature groups, features, and their effectiveness in terms of area under curve of the receiver operating characteristic (ROC_{AUC}) and area under the precision-recall curve (PR_{AUC}) using the default scikit-learn random forest. To the best of our knowledge, only 7 of the 47 features have been previously evaluated for Wikidata.

Feature group			**Test dataset**	
Feature	Reference	Wikidata	ROC_{AUC}	PR_{AUC}
Content features		–	0.735	0.141
Character features		–	0.840	0.088
`lowerCaseRatio`		–	0.873	0.042
`upperCaseRatio`	[2]	–	0.852	0.036
`nonLatinRatio`		–	0.870	0.020
`latinRatio`		–	0.865	0.020
`alphanumericRatio`	[2]	–	0.847	0.020
`digitRatio`	[2]	–	0.833	0.018
`punctuationRatio`		–	0.859	0.012
`whitespaceRatio`		–	0.854	0.011
`longestCharacterSequence`	[2]	–	0.767	0.010
`asciiRatio`		–	0.547	0.004
`bracketRatio`		–	0.516	0.003
Word features		–	0.811	0.140
`languageWordRatio`		–	0.575	0.104
`containsLanguageWord`		–	0.558	0.101
`lowerCaseWordRatio`	[30]	–	0.763	0.019
`longestWord`	[2]	–	0.814	0.018
`containsURL`		–	0.508	0.012
`badWordRatio`	[2]	–	0.514	0.005
`proportionOfQidAdded`	[17]	×	0.643	0.003
`upperCaseWordRatio`	[30]	–	0.549	0.003
`proportionOfLinksAdded`	[17]	×	0.560	0.002
Sentence features		–	0.686	0.025
`commentTailLength`		–	0.844	0.030
`commentSitelinkSimilarity`		–	0.736	0.006
`commentLabelSimilarity`		–	0.770	0.006
`commentCommentSimilarity`		–	0.730	0.005
Statement features		–	0.807	0.005
`propertyFrequency`	[17, 27]	×	0.793	0.005
`itemValueFrequency`		–	0.670	0.003
`literalValueFrequency`		–	0.533	0.002
Contextual features		–	0.870	0.289
User features		–	0.909	0.156
`userCountry`	[2]	–	0.911	0.212
`userTimeZone`		–	0.911	0.192
`userCity`		–	0.901	0.186
`userCounty`		–	0.909	0.145
`userRegion`		–	0.909	0.115
`cumUserUniqueItems`		–	0.933	0.113
`userContinent`		–	0.843	0.107
`isRegisteredUser`	[2, 17]	×	0.915	0.073
`userFrequency`	[27]	–	0.952	0.043
`isPrivilegedUser`	[36]	–	0.611	0.003
Item features		–	0.674	0.009
`logCumItemUniqueUsers`		–	0.678	0.009
`logItemFrequency`		–	0.665	0.007
Revision features		–	0.798	0.187
`revisionTag`		–	0.865	0.265
`revisionLanguage`	[17]	×	0.888	0.052
`revisionAction`	[17]	×	0.896	0.018
`commentLength`	[2]	–	0.827	0.016
`isLatinLanguage`		–	0.820	0.014
`revisionPrevAction`		–	0.764	0.007
`revisionSubaction`	[17]	×	0.799	0.005
`positionWithinSession`		–	0.557	0.004

Statement features. To quantify vandalism in an item's body, namely statements and sitelinks, we include three features in our model. For a given statement that is affected by an edit, these features compute the frequency with which its property has been used within Wikidata, and that of its value, distinguishing item values and literal values (see Figure 1): `propertyFrequency`, `itemValueFrequency`, and `literalValueFrequency`. These features basically quantify the "accumulated popularity" of properties and values within Wikidata, and while they do not pinpoint vandalism by themselves, they help doing so in combination with other features. Sitelinks are already covered by the features described above. The site ID is represented by the language parameter (i.e., by the feature `revisionLanguage`) and the sitelink's title appears in the comment tail and thus is subject to our character, word, and sentence features.

[3] These self-explanatory names are used consistently throughout the paper as well as the accompanying source code.

We further experimented with more elaborate statement features: for example, abstracting from item values by computing their super-item according to the item hierarchy induced by the instance-of property and using that as a categorical feature, and, computing conditional probabilities of pairs of item, super-item, property, and value as features, such as $P(\text{property}|\text{item})$, $P(\text{property}|\text{super-item})$, and $P(\text{value}|\text{property})$. The intuition of these features was to catch cases where properties or values are supplied in statements that are unexpected in comparable contexts (e.g., with a few exceptions, the gender property should only have the values "male" or "female"). However, neither of these features improved the effectiveness of our model, which can be attributed to the fact that Wikidata's knowledge graph is still only loosely connected. Also, features based on qualifiers, references, ranks, and badges had only a negligible effect. They, too, are not widely used, yet. Nevertheless, the above features may gain importance in the future.

3.3 Context Features

As much as the content of an edit may reveal its nature with respect to being vandalism, the context of an edit helps a lot as well: context features include features that quantify users, the edited item, and their respective histories.

User features. To quantify Wikidata's users, our features capture user status, user experience, and user location. User status is encoded as Boolean feature `isRegisteredUser` which indicates whether a user is registered or anonymous. User experience is captured by the number of revisions a user has contributed to the training dataset (`userFrequency`), the cumulated number of unique items a user has edited up until the revision in question (`cumUserUniqueItems`), and the Boolean feature `isPrivilegedUser` that indicates whether or not a user has administrative privileges. These features are perhaps most predictive for well-intentioned users rather than vandals, since the latter more often remain anonymous and contribute little to nothing before engaging in vandalism. The IP address of anonymous users at the time of editing is recorded, which allows for their geolocation; using a geolocation database [15], we derive the features `userContinent`, `userCountry`, `userRegion`, `userCounty`, `userCity`, and `userTimeZone`. For registered users, whose IP addresses are withheld by Wikimedia for privacy reasons, these features are set to a non-existent location. Features that had a negligible effect and did not make it into our final model include the time difference since the last revision of a user, the average number of bytes added by a user per revision, and the number of edits a user has made on item heads vs. item bodies.

Item features. To inform our vandalism detection model about the item being edited, we devise features to characterize and quantify item popularity. We compute the number of revisions an item has (`logItemFrequency`), and the number of unique users that have created them (`logCumItemUniqueUsers`). To avoid overfitting, we apply a log transformation on both features and round the result. Since 70% of vandalism cases are the only ones in their item's entire revision history [14], pinpointing frequently vandalized items is infeasible. Features that did not make it into our final model therefore include basic item statistics such as numbers of labels, descriptions, aliases, statements, and sitelinks an item has, as well as the Boolean feature indicating whether an item describes a human (which forms part of ORES baseline). These features overfit, whereas a log transformation did not help, either.

Revision features. Further features encode meta data about a revision, namely revision type, revision language, revision context, and revision tags. Based on the automatically generated comment of a revision, its revision type can be derived from the comment's `<action>` and `<subaction>` as features `revisionAction` and `revisionSubaction`, which encode content types affected (e.g., label, description, alias, statement, sitelink) and change type (insert, add, remove). The affected language `revisionLanguage` can be derived from the parameters `<param>+`. The feature `isLatinLanguage` indicates whether or not the language is usually written in Latin alphabet [38]. Some obvious damaging edits insert, for example, Chinese characters as an English label or the other way around. Revision context is encoded using the `revisionPrevAction` and `positionWithinSession`. The former denotes the content type changed in the previous revision on the same item, and the latter denotes this revision's position within a session of consecutive revisions by the same user on the same item. Moreover, rule-based scripts on Wikidata assign revision tags to revisions. We found 26 different tags which can be divided into two groups, namely tags originating from the Wikidata Abuse Filter [38], and tags originating from semi-automatic editing tools such as the "Wikidata Game" and the "Visual Editor." The tags are encoded in the feature `revisionTag`. Features that did not make it into our final model include revision size and revision time in the form of hour of day, day of week, and day of month. Despite trying different scaling and discretization techniques, these features were prone to overfitting.

4. EVALUATION DATA

To evaluate our Wikidata vandalism model, we employ a significant portion of Wikidata's revision history: our evaluation is based on the recently published Wikidata Vandalism Corpus WDVC-2015 [14]. This section briefly describes the corpus and gives relevant corpus statistics, discusses its division into datasets for training, validation, and test, and reports statistics on selected features.

4.1 Wikidata Vandalism Corpus WDVC-2015

The Wikidata Vandalism Corpus WDVC-2015 [14] is currently the only large-scale vandalism corpus for crowdsourced structured knowledge bases available. It contains all of about 24 million revisions that were manually created between October 2012 (when Wikidata went operational) and October 2014, disregarding revisions created automatically by bots. A total of 103,205 revisions were labeled as vandalism if they were reverted using Wikidata's rollback function—an administrative tool dedicated to revert vandalism [40]. A manual investigation shows that 86% of the revisions labeled as vandalism are indeed vandalism and only about 1% of the revisions labeled non-vandalism are in fact vandalism that has either been reverted manually, or not at all.

Table 2 gives an overview of key figures of the corpus. Regarding the entire corpus (first column), the 24 million revisions were created by 299,000 unique users editing about 7 million different items in about 14 million work sessions. About 18% of users have vandalized at least once and 1% of items were targeted at least once. Comparing item parts, there are about 4 times more revisions on item bodies than on item heads. However, the majority of vandalism occurs on the latter. Similarly, there are about 2.7 times more users vandalizing item heads than bodies, and the vandalism of about 2 times as many vandalized items can be found within its head. We hypothesize that vandals are drawn more to an item's head since it is rendered at the top of the page. Also, item heads are currently used for search suggestions on Wikipedia and mobile devices, thus being visible to a large audience. However, the focus of vandals will likely shift towards item bodies as Wikidata's integration into third party information systems progresses. Altogether, we are dealing with an unbalanced learning task, where 0.4% of all revisions are the to-be-detected vandalism. Considering item heads, the imbalance is 1.3%, whereas it is 0.2% for item bodies.

Table 2: The Wikidata Vandalism Corpus WDVC-2015 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	24,004	103	23,901	4,297	59	4,238	17,202	41	17,160
Sessions	14,041	63	13,980	2,693	44	2,650	11,313	20	11,294
Items	6,665	51	6,665	1,805	37	1,789	5,981	18	5,978
Users	299	55	249	157	41	118	164	15	151

Table 3: Evaluation datasets for training, validation, and test in terms of time period covered, revisions, sessions, items, and users. Numbers are given in thousands.

Dataset	From	To	Revisions	Sessions	Items	Users
Training	May 1, 2013	Jun 30, 2014	12,758	7,399	4,401	208
Validation	Jul 1, 2014	Aug 31, 2014	4,094	2,799	2,327	41
Test	Sep 1, 2014	Oct 31, 2014	3,975	2,666	2,200	40

4.2 Datasets for Training, Validation, and Test

The Wikidata Vandalism Corpus has not been split into datasets for training, validation, and test. Simply doing so at random would be false, since unrealistic situations might occur where an item's later revisions are used to train a classifier to classify its earlier revisions, and, where some revisions of a user's work session end up in different datasets. Given that a significant amount of vandalism occurs within work sessions that result in more than one revision, this renders the latter likely. This and the fact that a vandalism detector has to deal with a stream of revisions in the order in which they are created, we found it appropriate to split the corpus by time. Table 3 gives an overview of how the corpus was split into datasets for training, validation, and test. Although the corpus comprises revisions that date back to October 2012, we omit all of the revisions up to May 2013, since beforehand Wikidata's data model and serialization format was relatively unstable. In our experiments, we performed all feature selection and hyperparameter tuning *solely* based on the validation dataset. Only after our four models were optimized on the validation dataset, we ran them on the test dataset to evaluate their effectiveness in comparison to that of our two baselines.

4.3 Statistics on Selected Features

During feature engineering and selection, we analyzed each individual feature against the training dataset in order to gain insights into their nature; Table 4 reports analyses for selected features. The best-performing feature is `revisionTag`, which has not been evaluated before. Revision tags divide into two categories: tags by the Wikidata Abuse Filter and tags by semi-automatic editing tools, such as the Wikidata Game. The former provide a signal for vandalism while the latter provide a signal for regular edits. The feature `revisionLanguage` reveals that while only one third of Wikidata's content is language-dependent it still attracts about 90% of all vandalism. In particular, we found 11.51% of revisions affecting Hindi content to contain vandalism, whereas only 2.43% of revisions affecting English do so. Likewise, regarding `userCountry`, 35.29% of revisions by unregistered users from India contain vandalism, significantly more often than from other countries. Finally as outlined above, revisions that contain language names (`languageWordRatio`) have an empirical probability of 6.61% to be considered damaging and being rolled back. These findings suggest new opportunities to study cultural differences in crowdsourcing knowledge, but they also hint at ways to improve Wikidata's interfaces.

Table 4: Statistics on selected features (`revisionTag`, `languageWordRatio`, `revisionLanguage`, and `userCountry`). The tables show the number of vandalism revisions, total revisions, and the empirical vandalism probability. Rows are ordered by vandalism revisions. Numbers are given in thousands.

`revisionTag`	Vand.	Total	Prob.
Rev. with tags	52	8,619	0.60%
By abuse filter	49	122	39.90%
By editing tools	3	8,496	0.03%
Rev. w/o tags	52	15,386	0.34%

`lang.Word.Ratio`	Vand.	Total	Prob.
Rev. with comment	102	23,304	0.44%
Ratio equals 0	79	22,955	0.34%
Ratio greater than 0	23	349	6.61%
Rev. w/o comment	1	700	0.21%

`revisionLang.`	Vand.	Total	Prob.
Rev. with lang.	92	8,747	1.05%
English	40	1,664	2.43%
Spanish	4	370	1.11%
Hindi	3	28	11.51%
German	3	865	0.31%
French	2	623	0.38%
Other languages	39	5,196	0.75%
Rev. w/o lang.	12	15,258	0.08%

`userCountry`	Vand.	Total	Prob.
Rev. by unreg. users	88	705	12.42%
USA	13	65	20.85%
India	11	31	35.29%
Japan	5	46	11.39%
United Kingdom	3	20	14.60%
Germany	3	45	6.09%
Other countries	52	498	10.49%
Rev. by reg. users	16	23,299	0.07%

5. MODEL OPTIMIZATION AND EVALUATION

This section reports on a series of experiments to optimize the detection performance of our Wikidata vandalism model, comparing it with two state-of-the-art baselines. Besides parameter optimization, we utilize bagging and multiple-instance learning to almost double our model's performance. To cut a long story short, Table 5 shows the evaluation results for all approaches and their optimizations, and Figure 2 shows the corresponding precision-recall curves. We further investigate our model's performance against the baselines in an online learning scenario, analyzing performance fluctuation over time. In all experiments, our model outperforms the baselines by factors ranging from 2 to 3.

5.1 Experimental Setup and Reproducibility

Baselines. Our baselines are the Wikidata Abuse Filter (FILTER) [38] and the Wikidata vandalism detector recently deployed within the Objective Revision Evaluation Service (ORES) [39]. The Abuse Filter implements 89 rules that, when triggered, create tags on revisions for later review. Our model incorporates the rules within our `revisionTag` feature. Though the Abuse Filter has long been operational, its performance has never been publicly evaluated. ORES is poised to detect quality problems across Wikimedia's projects. The component to detect Wikidata vandalism was recently released as a "birthday gift."[4] Since ORES is tightly integrated with Wikidata's backend, we reimplemented the vandalism detection component, reproduced its detection performance as reported by its authors,[5] and used the reimplementation within our experiments.

Performance measures. To assess detection performance, we employ two performance measures, namely the area under curve of the receiver operating characteristic (ROC_{AUC}), and the area under the precision-recall curve (PR_{AUC}). Regarding their advantages and disadvantages for imbalanced datasets, we refer to Davis and Goadrich [10] and He and Garcia [13]: while ROC_{AUC} is the de-facto standard for machine learning evaluation, we report PR_{AUC} as well for a more differentiated view with respect to the imbalance of our learning task. PR_{AUC} is essentially equivalent to average precision (AP), a common measure for ranking tasks [19].

Preprocessing. For some features, we performed additional preprocessing, e.g., to fill in missing values by the median, and to determine the value range of categorical features. For validation, preprocessing is based on data from October 2012 to June 2014 (incl.), for testing from October 2012 to August 2014 (incl.).

[4] https://www.wikidata.org/wiki/Wikidata:Third_Birthday/Presents/ORES

[5] https://github.com/wiki-ai/wb-vandalism/tree/master/tuning_reports

Table 5: Evaluation results of the Wikidata vandalism detector, WDVD, proposed in this paper, and that of two baselines FILTER and ORES. Performance measures are the area under curve of the receiver operating characteristic ($\mathrm{ROC}_{\mathrm{AUC}}$), and the area under the precision-recall curve ($\mathrm{PR}_{\mathrm{AUC}}$). Performance values are reported for the entire test dataset, divided by item part. The darker a cell, the better the performance.

Classifier	**Item head**		**Item body**		**Entire item**	
Optimization	$\mathrm{ROC}_{\mathrm{AUC}}$	$\mathrm{PR}_{\mathrm{AUC}}$	$\mathrm{ROC}_{\mathrm{AUC}}$	$\mathrm{PR}_{\mathrm{AUC}}$	$\mathrm{ROC}_{\mathrm{AUC}}$	$\mathrm{PR}_{\mathrm{AUC}}$
WDVD (our approach)						
Multiple-instance	**0.985**	**0.575**	**0.981**	**0.216**	**0.991**	**0.491**
Bagging	0.980	0.521	0.879	0.175	0.960	0.430
Optimized random forest	0.980	0.487	0.942	0.171	0.978	0.406
Default random forest	0.922	0.451	0.800	0.087	0.894	0.342
FILTER (baseline)						
Multiple-instance	0.819	0.345	0.893	0.020	0.900	0.218
Bagging	0.768	0.297	0.816	0.014	0.865	0.201
Optimized random forest	0.770	0.351	0.816	0.015	0.865	0.257
Default random forest*	**0.770**	**0.358**	**0.816**	**0.015**	**0.865**	**0.265**
ORES (baseline)						
Multiple-instance	0.962	0.269	0.946	0.132	0.975	0.228
Bagging	0.956	0.197	0.900	0.124	0.960	0.169
Optimized random forest	0.953	0.214	0.896	0.111	0.960	0.182
Default random forest*	**0.882**	**0.176**	**0.749**	**0.058**	**0.859**	**0.135**

*These approaches represent the state of the art; to the best of our knowledge, the outlined optimizations have not been tried with ORES and FILTER until now.

Learning algorithm. In a series of pilot experiments, we determined which learning algorithm is best suited for our task. The random forest [8] algorithm outperformed all others tested, including logistic regression and naive Bayes with different hyperparameters. This finding is corroborated by the fact that random forest has also been found to perform best for vandalism detection on Wikipedia [2, 31], and that it is the algorithm of choice for the ORES baseline. For brevity, and due to space constraints, we omit a detailed report of the performance of other classification algorithms.

Reproducibility. To ensure the reproducibility of our experiments, the code base underlying our research is published alongside this paper.[6] It enables those who wish to follow up on our work to replicate the plots and performance values reported. Our feature extraction implementation is in Java and depends on the Wikidata Toolkit,[7] our experiments are implemented in Python and depend on the scikit-learn library, version 0.17.1 [23], and, $\mathrm{PR}_{\mathrm{AUC}}$ is computed with the AUCCalculator, version 0.2 by Davis and Goadrich [10].

5.2 Random Forest Parameter Optimization

Table 5 (rows "Default random forest") shows the detection performance of our Wikidata vandalism detector, WDVD, and that of the baselines FILTER and ORES when employing the scikit-learn random forest implementation with default parameters. To improve our model, we first optimized the parameters *maximal tree depth*, *number of trees*, and *number of features per split* in a grid search against the validation dataset. Figure 3 (top) shows the random forest's performance depending on the maximal depth of the trees (fixing the other parameters at their defaults). The best $\mathrm{PR}_{\mathrm{AUC}}$ of 0.493 on the *validation dataset* can be achieved with a maximal depth of 8, whereas for the baselines FILTER and ORES the optimal maximal depth is at 16. Likewise, we optimized the number of trees and the number of features per split: slight improvements were achieved by, simultaneously, increasing the number of trees, increasing the maximal depth, and decreasing the number of features per split. However, increasing the number trees linearly increases runtime at marginal performance improvements. For WDVD, we hence stick to the default number of trees (10), the default number of features per split ('sqrt') and use a maximal depth of 8. For ORES and FILTER, we do so, too, except for a maximal depth of 16. Table 5 (rows "Optimized random forest") shows the resulting performance on the test dataset; Figure 2 the corresponding precision-recall curves. Both $\mathrm{ROC}_{\mathrm{AUC}}$ and $\mathrm{PR}_{\mathrm{AUC}}$ of WDVD and ORES significantly improve: WDVD by 19% (from 0.342 $\mathrm{PR}_{\mathrm{AUC}}$ to 0.406 $\mathrm{PR}_{\mathrm{AUC}}$), and ORES by 35%. FILTER did not improve.

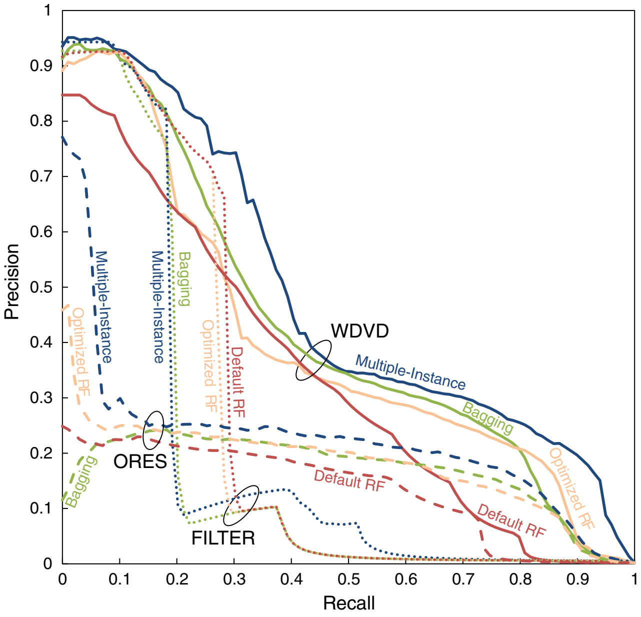


Figure 2: Precision-recall curves corresponding to the approaches listed in Table 5 on the test dataset.

5.3 Training Dataset Size and Bagging

Given the size of our training dataset (see Table 3), we investigated how much training data is actually required to achieve the performance of our model, which ultimately lead us to consider bagging as another means of optimization. Figure 3 (bottom) shows the performance of our model when taking random samples of the training data. The performance varies relatively little with sample size, possibly due to redundant or related revisions in the training dataset, such as users adding "instance of human" in bulk to many different items in a coordinated effort. Manual spot checks also showed items having a life cycle where, e.g., labels and sitelinks are inserted before specialized properties. But we did not further investigate this observation and leave it for future work.

These findings made us wonder whether it would help to create an ensemble of models, each trained on a subset of the training dataset. We experimented with bagging [7] and performed another grid search to determine good random forest parameters: 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 2 features per split. It should be noted, though, that detection performance varied only slightly for different hyperparameters and reasonable performance values could be achieved across a wide range of parameter settings. Indeed, bagging increased the detection performance on our *validation dataset* to $\mathrm{PR}_{\mathrm{AUC}}$ 0.517. Table 5 (rows "Bagging") shows detection performance when applying bagging with optimized parameters on the test dataset; Figure 2 shows the corresponding precision-recall curves: bagging yields a 6% improvement for WDVD over the optimized random forest (from 0.406 $\mathrm{PR}_{\mathrm{AUC}}$ to 0.430 $\mathrm{PR}_{\mathrm{AUC}}$), but $\mathrm{ROC}_{\mathrm{AUC}}$ suffers. The baselines hardly improve.

[6] http://www.heindorf.me/wdvd

[7] https://www.mediawiki.org/wiki/Wikidata_Toolkit

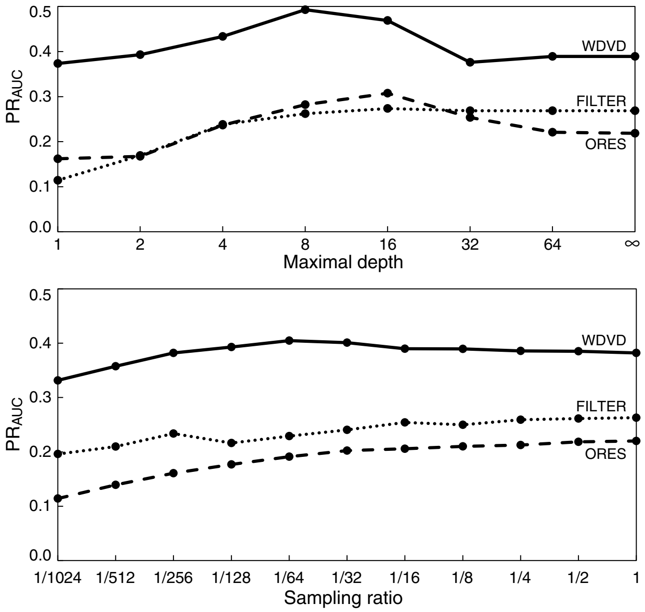


Figure 3: *Top*: validation performance dependent on maximal depth in a random forest. *Bottom*: validation performance dependent on random samples of training data using a default random forest, averaged over ten repetitions with different random seeds for sampling and for the random forest.

5.4 Multiple-Instance Learning

Since the user interface of Wikidata makes users submit each change individually, this may result in many consecutive revisions by the same user on the same item, which we call a work session. Until now, we have considered every revision of an item in isolation, disregarding sessions. This is unjustified, since one case of vandalism calls into question all other revisions created in the same session, and 60% of revisions are part of a session consisting of at least two revisions. To improve our model further, we exploit work sessions via multiple-instance learning and experiment with two such techniques, namely single-instance learning (SIL) and simple multiple-instance learning (Simple MI) [4, 12].

SIL is a so-called instance space method which works as follows: first, a classifier is applied on single revisions without considering sessions. Second, each revision in a session is assigned the same classification score, namely the average of the classification scores of a session's revisions.

Simple MI is a so-called embedded space method, where the vectors representing revisions in a session are embedded into a new space: a session comprises, say, d revisions represented as n-dimensional feature vectors, where $\mathbf{x}^i = (x_1^i, ..., x_n^i)$ denotes the i-th vector for $i \in [1, d]$. For such a session, a new $2n$-dimensional feature vector $\bar{\mathbf{x}} = (a_1, ..., a_n, b_1, ..., b_n)$ is computed, where $a_j = \max_{i \in [1,d]} x_j^i$ and $b_j = \min_{i \in [1,d]} x_j^i$ for $j \in [1, n]$. In plain words, a session is represented by a vector comprising the minimal and the maximal vector components of its constituent revision vectors. We compute the classification scores for each session and assign these scores to each individual revision of its session in much the same way as the aforementioned SIL method does.

For both, SIL and Simple MI, we employ the bagging random forest introduced above, whereas the aforementioned default and optimized random forest performed worse. Using SIL, we achieve 0.553 PR_{AUC} on the validation dataset, and using Simple MI, 0.546 PR_{AUC}. Lastly, we combine SIL and Simple MI by taking the arithmetic mean of their respective classification scores for a given revision, yielding 0.568 PR_{AUC} on the validation dataset.

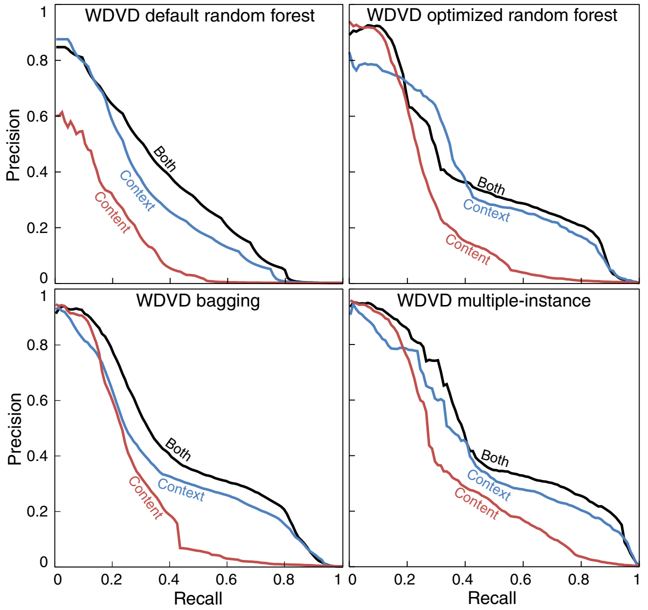


Figure 4: Precision-recall curves for content features, context features, and both combined on the test dataset by classifier.

Table 5 (rows "Multiple-instance") shows detection performance when applying our combination of multiple-instance learning techniques on top of our optimized bagging random forest on the test dataset; Figure 2 shows the corresponding precision-recall curves: this classifier achieves the best overall detection performance of an astounding 0.991 ROC_{AUC} at 0.491 PR_{AUC}, improving significantly over the optimized bagging random forest. The ORES baseline also benefits from multiple-instance learning, achieving 0.975 ROC_{AUC}, but only at 0.228 PR_{AUC}. The FILTER baseline does not seem to benefit from multiple-instance learning. On a practical note, employing multiple-instance learning brings about a minor limitation, namely that revisions cannot be classified immediately upon arrival, but only after a work session is concluded (i.e., after a timeout).

5.5 Content Features vs. Context Features

In all of the above optimizations, our model incorporates all the features listed in Table 1. Nevertheless, it is interesting to study the performance differences with regard to the feature subsets of content features vs. context features when applying the aforementioned optimizations. Figure 4 shows the results: we observe that content features particularly contribute to a high precision, whereas context features contribute to a high recall. Regardless of the classifier, combining both feature groups yields better performance than any single feature group alone (except for a small range of recall values for "Optimized random forest"). Both feature groups benefit from more advanced machine-learning algorithms than the default random forest, whereas the benefits for content features are larger. The best performance of both content and context features can be achieved with multiple-instance learning.

5.6 Online Learning

Vandalism detection can be viewed as an online learning problem since new revisions arrive at Wikidata in a steady stream, allowing for instant classifier updates once new vandalism cases are identified. In an initial experiment, we employed four online learning algorithms from scikit-learn: stochastic gradient decent with log loss, stochastic gradient decent with modified Huber loss, multinomial naive Bayes, and Bernoulli naive Bayes. All of them perform significantly worse than the random forest in a batch setting.

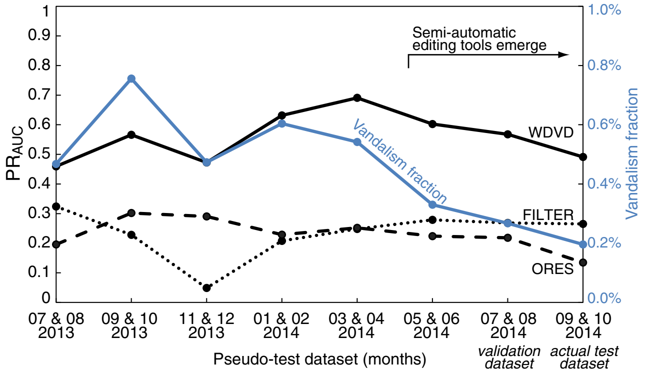


Figure 5: Effectiveness over time (black line plots) and vandalism fraction over time (blue line plot). The former are based on our best vandalism detection model (multiple-instance) and the two state-of-the-art baselines. Test datasets were varied while using all preceding revisions as training dataset.

Resorting to our best-performing classifier instead, we conducted another experiment to measure performance over time by using all revisions of two consecutive months as pseudo-test dataset, and all preceding revisions for training. This way, the training dataset grows as the pseudo-test dataset approaches the present, whereas the penultimate test dataset corresponds to our validation dataset, and the final test dataset to the actual one used in the above experiments (it still has never been used for training or optimization). The black line-plots in Figure 5 contrast the detection performance of our model to that of the two baselines. Our model's performance varies between a minimum of 0.46 PR_{AUC} in July & August 2013, and a maximum of 0.69 PR_{AUC} in March & April 2014. The FILTER baseline remains relatively constant over time, except for November & December 2013, where its performance plunges. The ORES baseline steadily declines over time, except for two time periods (July & August 2013 and January & February 2014). The rise and fall of detection performance over time raises the question why this is happening. A partial explanation may be found in the variance of class imbalance over time: the blue line-plot in Figure 5 shows the class imbalance in terms of fraction of vandalism in the pseudo-test datasets, and this curve correlates to some extent with that of our model. The main reason for the decreasing fraction of vandalism within more recent months is not a decrease of vandalism but an increase of non-vandalism which in turn coincides with the emergence of semi-automatic editing tools, such as the Wikidata Game. We further suspect that new forms of vandalism may have appeared which are not sufficiently represented in the training dataset, emphasizing that a vandalism detector needs to be updated regularly.

6. DISCUSSION

Item head vs. item body. In our experiments, we differentiate detection performance on item heads vs. its bodies (see Table 5). While significantly outperforming the baselines on both item parts, on head content, the detection performance of our model is higher. For PR_{AUC}, part of this difference can be explained by head content having a smaller class imbalance than body content (see Section 4.1) and PR_{AUC} varying with class imbalance [10, 6]. ROC_{AUC}, in turn, is independent of class imbalance [11]. Having reviewed a large sample of vandalism cases, we believe another important reason for this to be that vandalism in body content is more sophisticated: here, vandalism often attempts to introduce false facts that still appear alright so that it must be tackled at a semantic level. This hypothesis is also supported by the fact that most vandalism in item heads is done by unregistered or inexperienced users, whereas most vandalism in item bodies is done by registered users [14]. We leave it to future work to develop features that work better on item bodies, which may possibly involve external data sources such as other databases and web search engines to double-check information. Nevertheless, the labels, descriptions, and aliases found in item heads are widely visible across various Wikimedia projects, e.g., in Wikipedia search suggestions and Wikipedia infoboxes, requiring constant maintenance to ensure their integrity.

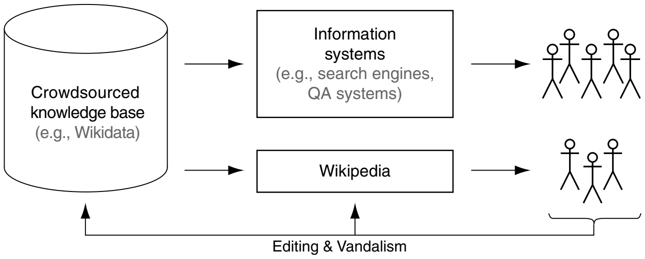


Figure 6: Data consumers of knowledge bases in general, and Wikidata in particular. Information systems provide the knowledge directly to its users, who in turn have the ability to edit the underlying knowledge base.

Practical applicability. We develop a model to automatically detect vandalism in Wikidata which achieves 0.991 ROC_{AUC} at 0.49 PR_{AUC}. Our model can be applied in two ways by setting up two classifiers with different performance characteristics, namely one with a high precision and one with a high recall. Up to 30% of vandalism can be detected and reverted fully automatically (see Figure 2). Considering cases where the classifier is less confident in its decision, they can still be ranked according to classifier confidence so as to allow for a focused review from likely to less likely vandalism. Altogether, it is possible to reduce the number of revisions that human reviewers have to review by a factor of ten while still identifying over 98.8% of all vandalism.[8]

Besides aiding Wikidata's community in its maintenance, the confidence scores returned by our classifier can be directly integrated into information systems so that they can avoid showing vandalized information (see Figure 6). Currently, Wikidata's most prominent data consumers are the different language editions of Wikipedia. For example, Wikipedia infoboxes (e.g., about genes [9, 20]) are populated from Wikidata, and Wikipedias which do not have an article about a certain topic (in particular smaller language editions) automatically generate article stubs from Wikidata containing the most important information.[9] It is foreseeable that many third party information systems will rely on Wikidata. Maybe the most notable use of Wikidata is its inclusion in major search engines, like Freebase in Google. Many factual queries can directly be answered without having to go through the search results. Moreover, question answering systems (e.g., Wolfram Alpha or IBM Watson) and personal assistants (e.g., Siri, Google Now, Cortana) may rely on semantic data from Wikidata. While most of them will rely on a multitude of data sources, having a vandalism score for Wikidata will help to better estimate the data's trustworthiness and to choose the most appropriate source.

Lastly, our current features do not impose high demands on computational power. For 24 million revisions, they can be computed on a standard workstation (16 cores and 64 GB RAM) in less than 2 hours. This results in a throughput of more than 3,000 revisions per second. Training a model takes about 10 minutes, and classifying a revision a fraction of a second.

[8] This number is derived from the ROC curve which is omitted due to space constraints: the true positive rate is 98.8% at a false positive rate of 9%; the latter implies that less than 10% of revisions are predicted to be vandalism (due to class imbalance).

[9] https://www.mediawiki.org/wiki/Extension:ArticlePlaceholder

7. CONCLUSION AND OUTLOOK

In this paper, we develop a new machine learning-based approach for the automatic detection of vandalism in the structured knowledge base Wikidata. Our vandalism detection model is based on a total of 47 features and a rigorous optimization using a variety of advanced machine learning techniques. As far as features are concerned, both content and context features are important. The best classification results have been obtained with a parameter-optimized random forest in combination with bagging and multiple-instance learning. Altogether, our classifier achieves 0.99 ROC_{AUC} at 0.49 PR_{AUC} and it thereby significantly outperforms the state of the art by a factor of 3 in case of the Objective Revision Evaluation Service (ORES), and by a factor of 2 in case of the Wikidata Abuse Filter.

As future work, we plan to further improve detection performance by implementing a retrieval-based vandalism detector that double-checks facts in external databases and web search engines. Furthermore, vandalism detection can be cast as a one-class classification problem, which opens interesting directions for the application of corresponding machine learning algorithms, as does deep learning which has not been applied to vandalism detection before. Another promising direction, which has not been explored for vandalism detection, could be to provide user-friendly explanations why a given revision is classified as vandalism, in order to improve and speed up manual review as well as to improve retention of new users in case their edits are reverted.

Acknowledgement

This work was supported by the German Research Foundation (DFG) within the Collaborative Research Center “On-The-Fly Computing” (CRC 901).

References

[1] B. Adler, L. de Alfaro, and I. Pye. Detecting Wikipedia Vandalism Using WikiTrust. CLEF Notebooks 2010.

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

[3] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding High-Quality Content in Social Media. WSDM 2008.

[4] J. Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, Aug. 2013.

[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. SIGMOD 2008.

[6] K. Boyd, V. Santos Costa, J. Davis, C. D. Page. Unachievable Region in Precision-Recall Space and Its Effect on Empirical Evaluation. ICML 2012.

[7] L. Breiman. Bagging Predictors. *Machine learning*, 24(2):123–140, 1996.

[8] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[9] S. Burgstaller-Muehlbacher, A. Waagmeester, E. Mitraka, J. Turner, T. E. Putman, J. Leong, P. Pavlidis, L. Schriml, B. M. Good, and A. I. Su. Wikidata as a Semantic Framework for the Gene Wiki Initiative. *bioRxiv* 032144, 2015.

[10] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. ICML 2006.

[11] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861-874, 2006.

[12] T. Gärtner, P. Flach, A. Kowalczyk, and A. Smola. Multi-Instance Kernels. ICML 2002.

[13] H. He and E. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept. 2009.

[14] S. Heindorf, M. Potthast, B. Stein, and G. Engels. Towards Vandalism Detection in Knowledge Bases: Corpus Construction and Analysis. SIGIR 2015.

[15] IPligence. Ipligence. http://www.ipligence.com, 2014.

[16] K. Y. Itakura and C. L. A. Clarke. Using Dynamic Markov Compression to Detect Vandalism in the Wikipedia. SIGIR 2009.

[17] A. Ladsgroup and A. Halfaker. Wikidata features. https://github.com/wiki-ai/wb-vandalism/blob/31d74f8a50a8c43dd446d41cafee89ada5a051f8/wb_vandalism/feature_lists/wikidata.py.

[18] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak. Analyzing and predicting question quality in community question answering services. WWW 2012.

[19] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] E. Mitraka, A. Waagmeester, S. Burgstaller-Muehlbacher, L. M. Schriml, A. I. Su, and B. M. Good. Wikidata: A platform for data integration and dissemination for the life sciences and beyond. *bioRxiv* 031971, 2015.

[21] S. M. Mola-Velasco. Wikipedia Vandalism Detection Through Machine Learning: Feature Review and New Proposals: Lab Report for PAN at CLEF 2010. CLEF Notebooks 2010.

[22] P. Neis, M. Goetz, and A. Zipf. Towards Automatic Vandalism Detection in OpenStreetMap. *ISPRS International Journal of Geo-Information*, 2012.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] M. Potthast, B. Stein, and R. Gerling. Automatic Vandalism Detection in Wikipedia. ECIR 2008.

[25] L. Ramaswamy, R. Tummalapenta, K. Li, and C. Pu. A Content-Context-Centric Approach for Detecting Vandalism in Wikipedia. Collaboratecom 2013.

[26] C. Shah and J. Pomerantz. Evaluating and Predicting Answer Quality in Community QA. SIGIR 2010.

[27] C. H. Tan, E. Agichtein, P. Ipeirotis, and E. Gabrilovich. Trust, but Verify: Predicting Contribution Quality for Knowledge Base Construction and Curation. WSDM 2014.

[28] T. P. Tanon, D. Vrandecic, S. Schaffert, T. Steiner, and L. Pintscher. From Freebase to Wikidata: The Great Migration. WWW 2016.

[29] K.-N. Tran and P. Christen. Cross Language Prediction of Vandalism on Wikipedia Using Article Views and Revisions. PAKDD 2013.

[30] K.-N. Tran and P. Christen. Cross-Language Learning from Bots and Users to Detect Vandalism on Wikipedia. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):673–685, Mar. 2015.

[31] K.-N. Tran, P. Christen, S. Sanner, and L. Xie. Context-Aware Detection of Sneaky Vandalism on Wikipedia Across Multiple Languages. PAKDD 2015.

[32] L. von Ahn. Offensive/Profane Word List. http://www.cs.cmu.edu/~biglou/resources/, 2009.

[33] D. Vrandečić and M. Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 2014.

[34] W. Y. Wang and K. R. McKeown. "Got You!": Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-semantic Modeling. COLING 2010.

[35] A. West and I. Lee. Multilingual Vandalism Detection Using Language-Independent & Ex Post Facto Evidence. CLEF Notebooks 2011.

[36] A. G. West, S. Kannan, and I. Lee. Detecting Wikipedia Vandalism via Spatio-temporal Analysis of Revision Metadata. EUROSEC 2010.

[37] Q. Wu, D. Irani, C. Pu, and L. Ramaswamy. Elusive Vandalism Detection in Wikipedia: A Text Stability-based Approach. CIKM 2010.

[38] Wikimedia Foundation. Wikidata Abuse Filter. https://www.wikidata.org/wiki/Special:AbuseFilter, 2015.

[39] Wikimedia Foundation. Objective Revision Evaluation Service. https://meta.wikimedia.org/wiki/Objective_Revision_Evaluation_Service, 2016.

[40] Wikimedia Foundation. Wikidata:Rollbackers. https://www.wikidata.org/wiki/Wikidata:Rollbackers, 2016.