Wiki Workshop (11th edition) – June 20, 2024

Wikidata Vandalism Detection with Graph-Linguistic Fusion

Mykola Trokhymovych
Pompeu Fabra University

Diego Saez-Trumper
Wikimedia Foundation

Abstract

This paper introduces ongoing work on creating a new system to assist the Wikidata community in identifying vandalism on the platform. Utilizing advanced feature engineering methodologies, we build a dataset containing various forms of knowledge alterations, ranging from textual changes in descriptions or labels to structural modifications of knowledge triples. It allows us to develop a unified model capable of effectively addressing diverse change types, enhancing the model's usability, and facilitating ongoing system maintenance. Our study demonstrates the superior performance of our proposed model over the previous production ORES model.

Keywords: knowledge integrity, vandalism detection, fairness, machine learning, Wikidata

Introduction

Wikidata is the largest open-source, multilingual knowledge graph, playing a key role in modern information systems. It empowers web search engines with factual data, mitigating the risk of large language model hallucinations, and links factual information across different languages. As a collaborative and open platform, Wikidata faces challenges related to content quality and vandalism. To address these problems, Wikidata relies on dedicated volunteer editors who verify changes to the content, known as revisions. To support this effort, Wikimedia developed machine learning (ML) systems named ORES (Objective Revision Evaluation Service) to assist editors in identifying potentially damaging changes in a structured knowledge base (Sarabadani et al., 2017).

While ML systems can significantly help editors, there are open challenges to address, including model accuracy and bias against anonymous users. This study presents our initial findings on developing a new generation of models to detect vandalism in Wikidata. Leveraging our prior experience in creating and evaluating systems for Wikipedia, we adapt the procedure to Wikidata by implementing a novel feature engineering process (Trokhymovych et al., 2023). Similar to our previous approach, we construct our model using a community-generated target—the historical label indicating whether a revision was reverted or not.

The main contributions of this work are: *(i)* Defining the architecture of a new open-source, multilingual system for vandalism detection in Wikidata; *(ii)* Presenting a methodology for mining fine-grained content change features. *(iii)* Demonstrating preliminary results of significant model improvement compared to the current production model (ORES) in terms of accuracy and fairness.

Methods

System architecture. Our proposed system receives Wikidata revisions as input and returns a revert-risk score, indicating the probability of a given revision being reverted. The solution is based on converting all Wikidata record changes into text and then processing them using a multilingual Masked Language Model (MLM). To achieve this, alterations to knowledge triplets are converted into text by mapping Wikidata IDs with their corresponding English labels. Simultaneously, modifications to labels, descriptions, or any other textual components are directly processed by MLM. Subsequently, the processed MLM scores are combined with revision and editor metadata, including user and page age and user tags represented through one-hot encoding, and used as input for the final classifier. The inference process of the system is illustrated in Figure 1.

Data preparation. Our dataset construction process involves extracting data from multiple sources within the Wikimedia Data Lake. Initially, we collect metadata for all human-created Wikidata revisions from 01-01-2021 to 01-01-2024, utilizing the mediawiki history table. Given the rarity of reverts, the resulting dataset is too imbalanced. To address this, we balance the initial dataset by keeping all reverted revisions and adding a random sample of five times as many unreverted revisions.

Later, we utilize mediawiki wikitext to capture detailed information regarding content changes. The wikitext representation of a Wikidata item is a complex nested structure of dictionaries or lists. We compare the wikitext of each revision with its parent version to extract fine-grained signals from content modifications. Employing Deepdiff [1], we parse the content differences, getting fea-

[1] `https://github.com/seperman/deepdiff`

tures in the form of a list of inserts, removes, and changes. This includes but is not limited to alterations in descriptions, labels, and knowledge triplets. Later, those components are used to build MLM-based features integrated into the final classifier, which we observe later.

We use a time-based train-test splitting procedure to avoid time-related anomalies that can bias evaluation results. Also, the training dataset consists of two parts, where one is used to fine-tune the MLM model and another to train the final classifier (Figure 2)

Content processing. Content changes in our dataset may represent various actions (inserts, removes, changes) and types (descriptions, labels, knowledge triplets, etc.). We employ specific data preparation techniques to enable processing using a single language model. Initially, we convert changes to graph changes (knowledge triplets) into textual equivalents by mapping Wikidata IDs to their corresponding English labels. Approximately 9% of IDs lacking corresponding labels are mapped to a default value, ”unknown.” Also, we prepend action-specific prefixes to the input data as illustrated in Figure 3.

Models training. We utilize the bert-base-multilingual-cased model, initially trained on multilingual datasets covering 100+ languages. By feeding processed content changes as text input, we fine-tune the model for binary classification. Subsequently, we aggregate the MLM scores across all action types to generate features for the final classifier. Our final classifier model employs the Catboost classification algorithm.

Results

Performance. We use the AUC score and the precision at a recall level of 0.99, 0.90, and 0.5 as the main metrics for model comparison. All the metrics are calculated using the holdout testing dataset. As the baseline, we use the rule-based model (all anonymous revisions predicted as revert with a score of 1.0). We compare it with ORES and our proposed system (Graph2Text). The results are presented in Table 1. It is important to note that the comparison tables include only revisions for which ORES scores were available.

Our final findings demonstrate that the proposed system, Graph2Text, outperforms the current production system ORES across all proposed metrics by a substantial margin. It achieves an increase in the AUC score from 0.687 (ORES) to 0.846. Additionally, we evaluate the inference time on a CPU-only instance and determine that the proposed system can process approximately five revisions per second in a synchronous, sequential setup.

Fairness. Anonymous edits tend to be more vandalized than those by registered users, but Wikimedia encourages anonymous editor participation. The main idea is to prioritize retaining newcomers rather than penalizing them for errors, thereby mitigating the decline in active editors in the long term. Consequently, we incorporate a bias analysis of our model — an essential step before deploying similar models in real-world contexts.

To assess bias against anonymous users, we employ the Disparate Impact Ratio (DIR) and the Difference in AUC score for anonymous and registered users. The results are summarized in Table 2. Our analysis reveals that our proposed model exhibits a lower DIR value, indicating reduced discrimination against anonymous users. Additionally, the difference in AUC scores for anonymous and registered users is notably lower, indicating a smaller difference in model performance across these user groups.

Discussion

Summary. This paper presents the preliminary findings of a study focused on developing a novel system for detecting vandalism on Wikidata. We utilize a large dataset prepared using complex feature engineering techniques. The model shows significantly improved accuracy and fairness compared to the current production ORES system. These initial results underscore the promising potential of our proposed system in detecting vandalism in Wikidata.

Limitations. When interpreting the results, it is essential to recognize several limitations of the current study. Parsing differences between Wikidata record versions can be enhanced by including qualifier changes. Additionally, using Wikidata IDs for English label mapping may affect accuracy due to incomplete label availability. Furthermore, advanced data filtering techniques could improve model performance by addressing anomalies like self-reverts or edit-wars. These limitations highlight areas for our future system refinement and optimization.

References

[Sarabadani et al.2017] Amir Sarabadani, Aaron Halfaker, and Dario Taraborelli. 2017. Building automated vandalism detection tools for wikidata. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW ’17 Companion, page 1647–1654, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[Trokhymovych et al.2023] Mykola Trokhymovych, Muniza Aslam, Ai-Jou Chou, Ricardo Baeza-Yates, and Diego Saez-Trumper. 2023. Fair multilingual vandalism detection system for wikipedia. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD ’23, page 4981–4990, New York, NY, USA. Association for Computing Machinery.

Table 1: System performance on holdout testing set.

Model	AUC	Pr@R0.99	Pr@R0.9	Pr@R0.5
Rule-based	0.629	0.173	0.173	0.173
ORES	0.687	0.175	0.198	0.328
Graph2Text	**0.846**	**0.209**	**0.304**	**0.572**

Table 2: Fairness metrics evaluation.

Model	DIR	AUC diff
Rule-based	-	-
ORES	5.71	-0.092
Graph2Text	2.32	**-0.041**

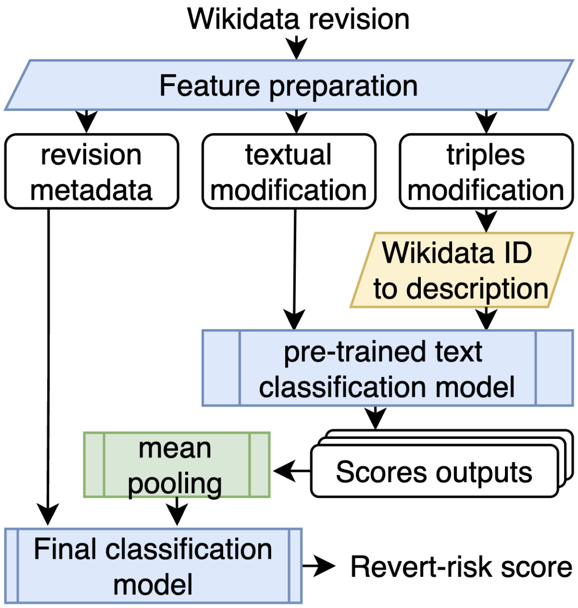

Figure 1: System inference logic schema.

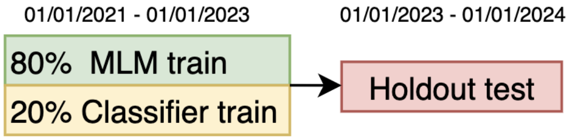

Figure 2: Train-test split logic

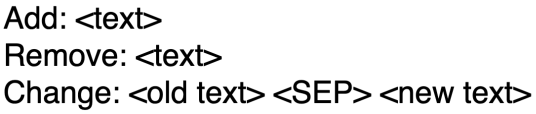
Add: <text>
Remove: <text>
Change: <old text> <SEP> <new text>

Figure 3: Text processing schema.