Towards classifying actors on Wikipedia talk pages

Carlin MacKenzie, John Hott
University of Virginia
ckm8gz@virginia.edu, jrhott@virginia.edu

May 8, 2020

Abstract

Wikipedia is at the centre of high-quality debate on the internet due to its popularity, diverse user community and collaborative editorial base. Additionally, all data, from traffic to user edits, is available under a free and open license. This paper demonstrates how information can be extracted from Wikipedia for analysis. We uncover that, for the second largest namespace, a database of all Wikipedia edits can be created on consumer hardware over several weeks. With access to more resources it is possible to create this database in 5 days. From this natural language processing, classification and network analysis can be performed.

1 Introduction

Nineteen years after its release, Wikipedia needs no introduction. Started as an experiment in anonymous, public collaboration, it is now the largest and most popular reference work on the internet[1]. Unfortunately, research into this extraordinary success is limited due to several factors. Firstly, like Wikipedia itself, documentation is community generated. There is no top down guidance of best practices or which tools to use. There are at least five lists of tools, and they do not hint when they were last actively developed.

Although the data is made as public as possible, the dumps of the database tables are not published. This is because there are various transformations applied before it can be released, like removing information about passwords and IP addresses. Instead, edit history is released as a set of ~600 archives, each of which extract to ~50 GB XML files. These files are very difficult to work with as most editors try to open the full file before displaying it. For this reason, the dumps must be accessed programmatically through streaming the file line by line.

The structure of Wikipedia itself can also prevent researchers engaging with the website. Wikipedia is split into several namespaces. The most familiar being the article namespace. Every article also has a talk page, which is commonly used as a forum to discuss the sources and structure of the article. Other namespaces exist, such as user, Wikipedia, etc. each having an associated talk namespace as well.

We posit that content posted on talk pages is much more diverse than on the article itself. Article edits are generally either contribution or vandalism, which can be classified by tools like ORES[2]. In contrast, talk page content is argumentative and emotive and hopefully a better indicator of user behaviour.

2 Related Work

While Wikipedia has many avenues for research, there are few papers in this space compared to Twitter or Facebook. Most recently Rawat et al.[11] attempted to classify abusive actors. They acquired their data by scraping user contributions from Wikipedia and applying machine learning to this data set. Their model provided an 84% accuracy, however the data set they used was very small. Our research instead expands to a dataset of to all Wikipedia edits.

In the field of vandalism detection, Javanmardi[4], provides a high performing and fast model. They used a data set of Wikipedia edits which were manually classified to be spam or not spam. They created a classifier with 66 features which had an accuracy of 95.5% Area Under Curve (AUC) on the test set. To create the high performing model, they used the Lasso technique which resulted in 27 features and 95% AUC.

Schneider et al.[12] discuss the articles' talk pages. They aimed to classify the diversity in these pages and created thirteen such categories. They explored how users could signal which category their edit belonged to for aggregation purposes. Interestingly, they found that the most controversial articles have relatively short talk pages due to repetitive arguments in which neither side convinces the other.

From a user standpoint we can look at the social networks of Wikipedia. Massa[8] found that extracting a network based dataset could be approached in three ways, each of them flawed. Manual extraction is the most reliable but very time consuming. Additionally, this would not find edits which were reverted. Scraping talk pages faced many challenges such as custom signatures. Finally, they used the Wikipedia XML dumps. This was the most accurate for finding user's edits but could not verify to whom users were replying.

Finally, Martinez-Ortuno et al.[7] looked at users' talk pages and how this is related to user activity. In contrast to article talk pages, user talk pages can be thought of as the user's profile where people can thank or ask questions of the user. This is therefore a good predictor of a user's standing in the community. The researchers did not find a direct correlation between negative messages and decreased edits, but did find a model that could be used to predict user edit activity.

3 Data Acquisition

For most use cases, it is possible to perform queries on the live Wikipedia dataset through the Wikimedia Toolforge. This is useful, however queries time out after thirty minutes by default and the query language used to interface with the database is SPARQL, for which there is a learning curve. If a Toolforge account is acquired, through emailing the Wikimedia Foundation, a longer query limit of twenty four hours is granted.

As we did not want these limitations placed on us, we investigated the available tools. Of the 29 available tools, 16 had not been developed in the last five years. Of the rest, none of the tools were helpful to creating a manageable database of edits. Consequently, we created a tool to fill this gap. We chose Python as it is the most popular and stable language for data science.

3.1 XML Data Dumps

The Wikipedia XML dumps have a strict structure which enables parsing to be performed. Each dump is composed of a siteinfo section followed by a variable number of pages. The siteinfo section includes in-

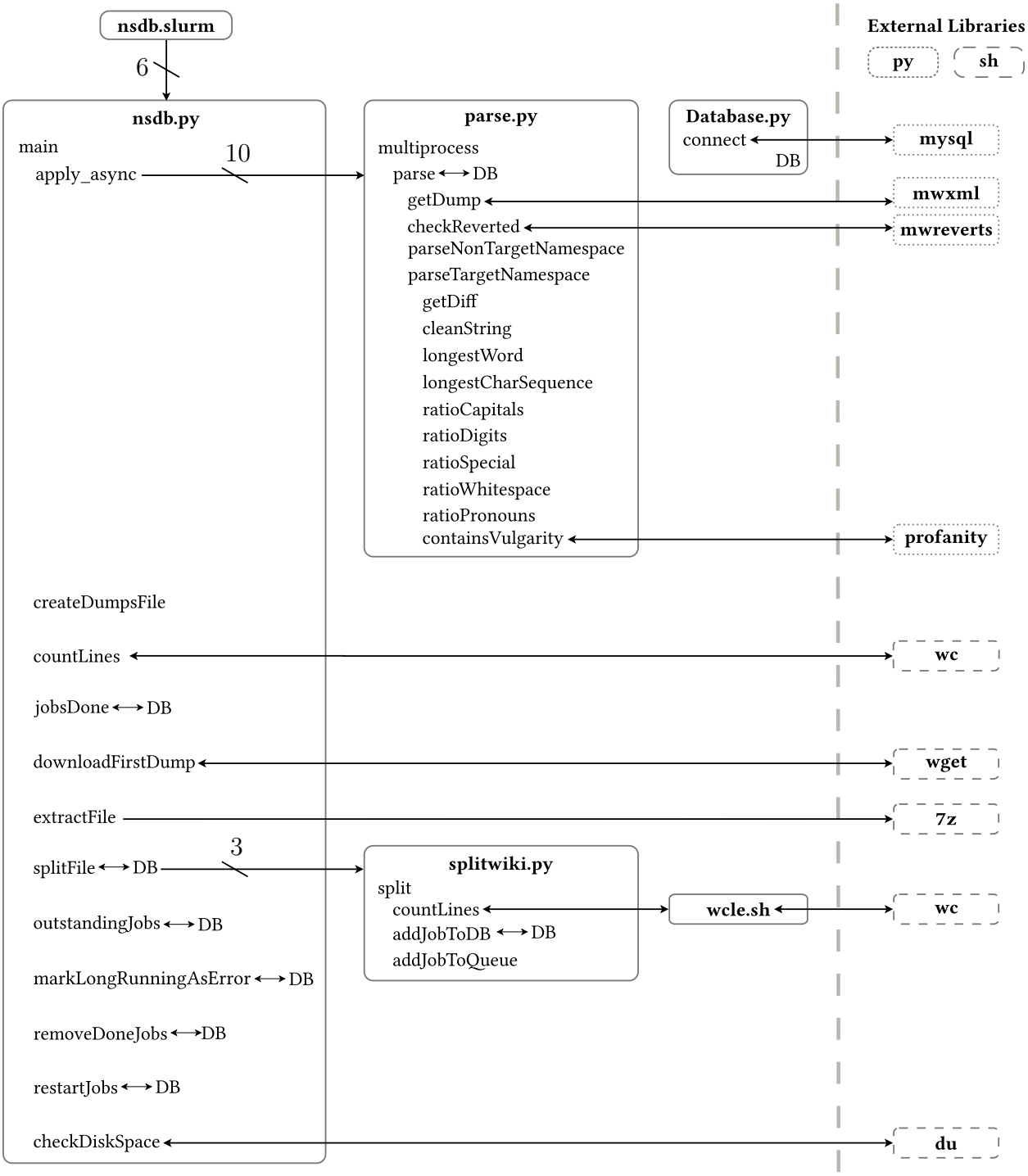

Figure 1: Program flow

formation such as the name of the site, the database that was dumped, and the namespaces of the wiki. Each page includes information about the title, id, namespace, followed by the full revision history of the page. Each revision has an id, a timestamp, contributor, comment, model, format, text, and a SHA-1 hash. The text field is the new full page after the edit is performed, no diff of the change is provided.

Even though the structure makes the dumps easy to parse, it was initially challenging to do this due to the file size. Fortunately, the Python library `mwxml`[9] allows for streaming of the file for processing.

To find the contribution of a user, manual diff-ing must be performed. The process of finding a diff is known as finding the longest common subsequence. The solution to this cannot be perfect as it can be ambiguous which transformation was performed between two states. In the end, we used `wdiff`[6], a linux tool which provides word level differences, as this produced the best results on average. wdiff transforms the two files so that there is one word per line and creates a diff between these files using the POSIX `diff`[3] utility.

3.2 Extraction Tool

We created a tool[5] which downloads, splits, and extracts features to create a database of edits for a Wikipedia namespace, which is specified on the command line.

Downloading the XML dumps can take significant time due to high load, throttling, traffic etc. Consequently, a speed test is performed to each mirror before the dump is downloaded.

As each dump is independent of the others, we can easily parallelise parsing. To maximise parallelisation and localise errors we split each dump into several partitions. This functionality is performed by a helper tool, `splitwiki.py`, which splits each dump at page boundaries.

The parsing of each partition is handled by our parser, `parse.py`. The parser iterates through each page in the dump and extracts features depending on whether the page is in the target namespace or not. If it is, we extract all the features described in the edit table. Otherwise, we count the number of edits and the number of reverted edits per user. For these pages in the target namespace we insert into the database for every revision, whereas in the non-target namespace we only insert for each user. Finally, we insert information about the page like the title, namespace, and number of revisions.

As there is a large variation in the edits made, error handling was very important. Errors are logged to a file unless it stops parsing, in which case it is logged to the database.

Co-ordinating these scripts is `nsdb.py` which is an acronym of Namespace Database, the name of the GitHub repository. It downloads a list of dumps and starts the process of downloading, splitting and processing in parallel. It accepts arguments of where it should store the files, the maximum space it should use and the number of cores it should keep available.

The strategy to parallelise this program was a combination of Slurm Workload Manager and Python multiprocessing. Slurm was used to distribute nsdb.py on several nodes, while multiprocessing creates several processes of split and parse on the same node.

Figure 1 shows how the different functions interact with each other and the external libraries that are relied upon.

page_id	int
namespace	smallint
title	varchar
file_name	varchar

Table 1: Page table

id	int
file_name	varchar
status	enum
error	text
start_time	timestamp
start_end	timestamp

Table 2: Partition table

id	int
user_id	int
username	varchar
ip_address	varbinary
number_of_edits	int
reverted_edits	int
talkpage_number_of_edits	int
talkpage_reverted_edits	int
namespaces	set

Table 3: User table

id	int
edit_id	int
edit_date	datetime
page_id	int
user_table_id	int
added	text
deleted	text
added_length	mediumint
deleted_length	mediumint
blanking	tinyint
comment_copyedit	tinyint
comment_length	tinyint
comment_personal_life	tinyint
comment_special_chars	decimal
del_words	mediumint
ins_capitalization	decimal
ins_digits	decimal
ins_external_link	smallint
ins_internal_link	smallint
ins_longest_character_sequence	smallint
ins_longest_inserted_word	smallint
ins_pronouns	decimal
ins_special_chars	decimal
ins_vulgarity	tinyint
ins_whitespace	decimal
reverted	tinyint

Table 4: Edit table

Figure 2: Schema of the tables

3.3 Data Extraction Performance

The bulk of processing was done under the following circumstances:

- nsdb.py running on 6 nodes
- target namespace was 1 (article talk pages)
- each node has 3 splitwiki.py processes
- splitwiki was set to create 80 partitions
- each node also has 10 parse.py processes

Under these circumstances, the performance was as follows:

- 100x reduction in database size to dump size
- 7 partitions parsed per minute
- 84 partitions per dump
- 5 dumps per hour
- 132 hours total parsing time

Running on consumer hardware, we found a reduction to 2-3 partitions parsed per minute. This would amount to several weeks of processing, which could be reduced if a sample of the dumps were used instead.

We did not have time to investigate the size and time taken to parse the article namespace. An estimate from the diagnostics we performed would suggest it would be at least 10 times larger and slower.

4 Database

When deciding which features we wanted to extract, we took guidance from Javanmardi[4]. In total we were able to extract 17 out of 27 of their features, which can be seen in the schema shown in Figure 2. Features that were not implemented were features in which implementation wasn't clear, like DDSR which was a metric of the user's reputation.

5 Future Work

Future researchers can now investigate in any number of directions. WikiProjects could be analysed to see how they evolve over time. Blocked users of various types could be investigated for editing patterns. Sentiment analysis could flag when a user "turns bad". Any of these projects would help guide future systems to create a better encyclopedia for all of us.

6 Conclusion

In this paper we have described a method for the creation of a database of Wikipedia edits in any namespace.

Acknowledgement

The author would like to thank Aaron Halfaker for his MediaWiki Utilities[10] which were invaluable for parsing the dumps and Lane Rasberry for his support of this project.

The author would also like to link to the repository and WikiMedia Research page but is unsure how to in a journal format.

References

[1] Alex Woodson. Wikipedia remains go-to site for online news. https://www.reuters.com/article/us-media-wikipedia/wikipedia-remains-go-to-site-for-online-news-idUSN08194291200070708, 2007. [Accessed 5-May-2020].

[2] Aaron Halfaker and R. Stuart Geiger. Ores: Lowering barriers with participatory machine learning in wikipedia, 2019.

[3] IEEE and The Open Group. Posix diff. https://pubs.opengroup.org/onlinepubs/9699919799/, 2008. [Accessed 8-May-2020].

[4] Sara Javanmardi, David W McDonald, and Cristina V Lopes. Vandalism detection in wikipedia: a high-performing, feature-rich model and its reduction through lasso. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 82–90. ACM, 2011.

[5] Carlin MacKenzie. Namespace database - a tool to create a database of wikipedia edits. https://www.github.com/carlinmack/NamespaceDatabase/, May 2020.

[6] Martin von Gagern. Gnu wdiff. https://www.gnu.org/software/wdiff/, 2014. [Accessed 8-May-2020].

[7] Sergio Martinez-Ortuno, Deepak Menghani, and Lars Roemheld. Sentiment as a predictor of wikipedia editor activity. 2014.

[8] Paolo Massa. Social networks of wikipedia. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, pages 221–230, 2011.

[9] MediaWiki. Mediawiki-utilities/mwxml — mediawiki, the free wiki engine, 2017. [Online; accessed 8-May-2020].

[10] MediaWiki. Mediawiki-utilities — mediawiki, the free wiki engine, 2020. [Online; accessed 8-May-2020].

[11] Charu Rawat, Arnab Sarkar, Sameer Singh, Rafael Alvarado, and Lane Rasberry. Automatic detection of online abuse and analysis of problematic users in wikipedia. In *2019 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, apr 2019.

[12] Jodi Schneider, John G Breslin, and Alexandre Passant. A content analysis: How wikipedia talk pages are used. 2010.