Analysis of Wikipedia talk pages created before their corresponding articles

Bill Dengler

30th November 2017

Subject: computer science

Research question: Why do some Wikipedia articles have their first visible edits to their talk pages occurring before those of the articles themselves?

word count: 3525

I declare that this work is my own work and is the final version.

I have acknowledged each of the words or ideas of another person, whether written, oral or visual.

Contents

1. Introduction

Wikipedia is a free online encyclopedia that anyone can edit. Founded in 2001 by Larry Sanger and Jimmy (Jimbo) Wales, the site now consists of over forty million articles in more than 250 languages, making it the largest and most popular online general reference work. As of 2015, the site was Ranked by Alexa as the 5th most visited website overall. [1]

Wikipedia allows anyone to edit, without requiring user registration. [2] The site permanently stores histories of edits made to its pages. Each page's history consists of a chronological list of changes (with timestamps in Coordinated Universal Time [UTC]) of each, differences between revisions, the username or IP address of the user making each edit, and an "edit summary" written by each editor explaining their changes to the page. Anyone can view a page's history on its corresponding history page, by clicking the "history" tab at the top of the page. [3]

Sometimes, Wikipedia page histories are incomplete. Instead of using the move function to rename a page (which transfers history to the new title), inexperienced editors occasionally move the text of the page by cut-and-paste. [3], [4] Additionally, users who are not logged in, or users who do not have the autoconfirmed right (which requires an account that is at least four days old and has made ten edits or more)[1] are unable to use the page move function, and sometimes attempt to move pages by cut-and-paste. [4] When pages are moved in this way, history is split, with some at the old title (before the cut-and-paste) and some at the new title (after the cut-and-paste). To fix this split history, a Wikipedia administrator must merge the histories of the two pages by moving revisions from the old title to the new one. [6]

For legal reasons, text on Wikipedia pages that violates copyright and is not subject to fair use must be deleted. In the past, entire pages with edits violating copyright would be deleted to suppress copyrighted text from the page history. However, deleting the entire page had the consequence of deleting the page's entire history, not just the copyrighted text. In many of these cases, this led to page history fragmentation. To mitigate this, Wikipedia administrators now tend to delete only revisions violating copyright using the revision deletion feature, unless there are no revisions in the page's history that do not violate copyright.

Originally, Wikipedia did not store full page histories. The site used a wiki engine called UseMod Wiki. [7] UseMod Wiki has a feature called KeptPages, which periodically deletes old page history to save disk space and "forgive and forget" mistakes made by new or inexperienced users. Due to this feature, some old page history was deleted by the UseMod Wiki software, so it has been lost.

[1]In some special cases, the privileges of the autoconfirmed right are granted manually by a Wikipedia administrator[5]

In February 2002, an incident known on Wikipedia as the “Great Oops”[8] caused the timestamps of many old edits to be reset to 25 February 2002, 15:43 or 15:51 UTC. Wikipedia had recently transitioned to the phase 2 software; the precursor to MediaWiki (their current engine) and the replacement for UseMod Wiki. [9] The Phase II Software’s new database schema had an extra column not present in the UseMod Wiki database. This extra column was filled in with a default value, which inadvertently caused the timestamp reset. [10]

Each Wikipedia page also has a corresponding talk page. Talk pages allow Wikipedia editors to discuss page improvements, such as controversial edits, splits of large pages into several smaller pages, merges of related smaller pages into a larger page, page moves (renames), and page deletions. [11] Since talk pages are just Wikipedia pages with a special purpose, they have page history like any other Wikipedia page, and all the aforementioned page history inconsistencies.

An indicator of page history inconsistency is the creation time of a Wikipedia page relative to its talk page. Logically, a Wikipedia page should be created before its talk page, not after; Wikipedians can’t discuss pages before their creation! The aim of this extended essay is to find out why some Wikipedia articles have edits to their talk pages appearing before the articles themselves.

2. Data collection

To determine which articles have edits to their talk pages occurring before the articles themselves, I wrote and ran a database query on Wikimedia Tool Labs[1], an openstack-powered cloud providing hosting for Wikimedia-related projects as well as access to replica databases; copies of Wikimedia wiki databases, sans personally-identifying information, for analytics and research purposes. [13] The Wikipedia database contains a page table, with a page_title column representing the title of the page. Since there are often multiple (related) Wikipedia pages with the same name, Wikipedia uses namespaces to prevent naming conflicts and to separate content intended for readers from content intended for editors. [14] In the page title and URL, namespaces are denoted by a prefix to the page's title; articles have no prefix, and article talk pages have a prefix of talk:. However, in the database, the prefix system is not used; the page_title column contains the page's title without the prefix, and the page_namespace column contains a numerical representation of a page's namespace. Wikipedia articles have a page_namespace of 0, and article talk pages have a page_namespace of 1. The page_id field is a primary key uniquely identifying a Wikipedia page in the database.

The revision table of the Wikipedia database contains a record of all revisions to all pages. The rev_timestamp column contains the timestamp, in SQL timestamp form [15], of a revision in the database. The rev_page column contains the page_id of a revision. The rev_id column contains a unique identifier for each revision of a page. The rev_parent_id column contains the rev_id of the previous revision, or 0 for new pages.

The database query retrieved a list of all Wikipedia pages in namespace 0 (articles) and namespace 1 (talk pages of articles). For each page, the title, timestamp of the first revision (the first revision to have a rev_parent_id of 0), and namespace were collected. My SQL query is below:

```sql
select page_title, rev_timestamp, page_namespace
from page, revision
where rev_parent_id=0
and rev_page = page_id
and (page_namespace=0 or page_namespace=1);
```

Due to the size of the Wikipedia database, I could not run the entire query at once; the connection to the database server timed out or the server threw a "query execution was interrupted" error. To avoid the error, I segmented the query, partitioning on the page_id field. During the query, I adjusted the size of each collected "chunk" to maximize the number of records collected at once; the sizes ranged from

[1]During the course of my writing of this extended essay, Wikimedia Tool Labs was renamed to Wikimedia Cloud Services. [12] The essay will use the old name, because that was current at the time of the conclusion of my research.

one million to ten million. To partition the query, I added a **where** clause as follows:

```
select page_title, rev_timestamp, page_namespace
from page, revision
where page_id >1000000
and page_id <=2000000
and rev_parent_id=0
and rev_page = page_id
and (page_namespace=0 or page_namespace=1);
```

I wrapped each database query in a shell script which I submitted to the Wikimedia Labs Grid; a cluster of servers that perform tasks on Wikimedia projects. [16] An example wrapper script follows:

```
#!/bin/bash
sql enwiki -e "query"
```

sql enwiki is an alias on Wikimedia Labs for accessing the database of the English Wikipedia, and query is the SQL query. The Wikimedia Labs Grid writes standard output to scriptname.out and standard error to scriptname.err, where scriptname is the name of the script. My set of wrapper scripts were named eecollect1.sh through eecollect12.sh, one script containing each line of the SQL query (see appendix A for the wrapper scripts submitted to the Wikimedia Tool Labs). Running **cat** eecollect*.out > eecollect.out concatenated the various "chunks" of output into one file for post-processing.

3. Post-processing

The database query retrieved a list of all articles and talk pages in the Wikipedia database, along with the timestamps of their first revisions. This list contained tens of millions of items; it was necessary to filter it to generate a list of articles where the talk page appeared to be created before the article. To do this, I wrote a Python program, *eeprocess.py* (see appendix B for source code) that read the list, compared the timestamps of the articles to those of their talk pages and generated a csv file of articles whose talk pages have visible edits before those of the articles themselves. The csv file contained the names of all articles found, along with the timestamp of the first revision to the article itself and the article's talk page. After downloading the concatenated output file from Wikimedia Labs, I ran my post-processor against it.

The first run of the post-processor found a list of 49,256 articles where the talk page was created before the article itself. Further investigation showed that many of these articles had talk pages created with in seconds of the article, which are not useful for my purposes; they are not indicative of missing history.

In hopes of reducing the list, I added a command-line option to the post-processor, −−window, that requires an article's talk page to be a specified number of seconds older than the article for inclusion in the list. In other words, an article's talk page must be at least −−window seconds older than the article itself to be included in the list. I then ran the post-processor with several values of −−window, saved a copy of the output of each run, and counted the number of articles found in each .csv file. To count the number of rows in an output file, I fed the file to standard input of the wc utility by piping the output of the **cat** command to wc. I used the wc −l switch to count the number of lines in the file. I then subtracted 1 from each result to avoid counting the header row. Table 3.1 contains the number of articles in the output of the post-processor given various values of −−window.[1]

Table 3.1: Number of articles in the output of *eeprocess.py* given various values of −−window. One day is equal to 86,400 seconds.

Time Period	Number of Articles
one day	26,040
one month (30 days)	20,877
six months (180 days)	15,755
one year (365 days)	12,616
two years (730 days)	8,983
five years (1,825 days)	3429

eeprocess.py reads the SQL query output in a linear fashion. Since the program must read one row at a

[1] The full contents of the .csv file could not be included due to length (the table spans hundreds of pages).

time from the file, it runs in $\mathcal{O}(n)$ time. In other words, the speed of the program is directly proportional to the number of rows in the input (eecollect.out) file. While this linear algorithm is extremely inefficient for large SQL queries, it is necessary for accurate results; the program must read each page name and timestamp into a corresponding dictionary for the page's namespace.

To check if an article's talk page is older than the article itself, *eeprocess.py* used the dateutil.parser module in the Python standard library to convert the SQL timestamp of the first revision of each article into a Python datetime.Datetime object; a datatype in the standard library for representing dates and times. These datetime.Datetime objects are then converted to unix time using the datetime.Datetime .timestamp method. The difference of these timestamps is taken and checked against --window; if the difference is greater than or equal to --window, it is included in the list. Instead of comparing unix timestamps, I could have treated the timestamps as integers, taken their difference and checked if it was greater than or equal to a predetermined value; this would have been more efficient, but an accurate --window option would have been near impossible to implement.

4. Automatic analysis

After filtering the list to find articles whose talk pages were created at least one day before the articles themselves (with the `--window` option to *eeprocess.py*), I wrote another Python program (see appendix C for source code) to compare the list against a database dump of the Wikipedia deletion and move logs, taken on 20 April 2017. The program writes an analysis of this comparison to a .csv file.[1]

My program, *eeanalyze.py*, scanned for two possible reasons why the article's talk page would appear to have edits before the article itself. If an article was deleted due to copyright violation, the article will be deleted with "copyright" or "copyvio" (an on-wiki abbreviation of "copyright violation") in the text of the deletion log comment field.

Normally, article deletions must be discussed by the community before they take place. However, in some cases, articles may be speedily deleted (deleted without discussion) by a Wikipedia administrator. Criterion G12 (unambiguous copyright infringement) and historical criterion A8 (blatant copyright infringement) apply to copyright violations. [17] If an article is speedily deleted under one of these criteria, a speedy deletion code for copyright violation ("A8" or "G12") will appear in the comment field of the deletion log. If a matching string is found in an article's deletion log comments, *eeanalyze.py* flags the article as being deleted for copyright violation.

Another possible cause is an incorrect article move; in some cases, an article is moved by cut-and-paste, but its talk page is moved correctly. When this happens, the article's history is split, but the talk page's history is complete. To fix this, the article's history needs to be merged by a Wikipedia administrator. *eeanalyze.py* searches the page move logs for instances where a talk page is moved (the destination of a page move is the current article title), but no move log entry is present for the article itself.

eeanalyze.py also generates *eemoves.csv*, a file containing a list of "move candidates"; page moves where the destination appears in the list of articles generated by *eeprocess.py.* While I ultimately did not use this list during my analysis, it may yield additional insight into the page history inconsistencies.

eeanalyze.py uses the mwxml Python library to efficiently process XML database dumps from MediaWiki wikis, like Wikipedia. For a MediaWiki XML database dump, the library provides `log_items`; a generator of `logitem` objects containing log metadata from the dump. Initially, the library only supported dumps containing article revisions, not logs. I contacted the developer requesting the latter functionality. Basic support for log dumps was added in version 0.3.0 of the library [18]; I tested this new support through my program and reported library bugs to the developer.

[1]The data in this .csv file could not be included due to length.

eeanalyze.py reads the database dump in a linear fashion. Since linear search runs in $\mathcal{O}(n)$ time, its speed is directly proportional to the number of items to be searched. While linear search is extremely inefficient for a dataset of this size, it is necessary for accurate results; there is no other accurate way to check the destination (not source) of a page move.

In theory, I could have iterated over just the articles found by *eeprocess.py*, binary searching the dump for each one and checking it against the conditions. While the number of articles to search (n) would have been reduced, the streaming XML interface provided by mwxml does not support Python's binary search algorithms. Additionally, if it was possible to implement this change, it would have slowed the algorithm to $\mathcal{O}(n \log n)$ because I would need to sort the log items by name first.

5. Classification of results

Once the automatic analysis was generated, I wrote a Python program, *eeclassify.py* (see appendix D for source code). This program compared the output of *eeprocess.py* and *eeanalyze.py* and performed final analysis. The program also created a .csv file, *eefinal.csv*, which contained a list of such articles, the timestamp of their first main and talk edits, the result (if any) of automatic analysis, and (when applicable) log comments. [1]

A bug in an early version of *eeprocess.py* led to incorrect handling of articles with multiple revisions where rev_parent_id = 0. The bug caused several timestamps of the first visible edits to some pages to be miscalculated, leading to false positives. The bug also caused the output to incorrectly include pages that had some edits deleted by an administrator using the revision deletion feature. When I discovered the bug, I patched *eeprocess.py* and reran *eeprocess.py* and *eeanalyze.py* to correct the data. While I am fairly confident that *eeprocess.py* no longer incorrectly flags pages with revision deletions, *eeclassify.py* attempts to filter out any pages that have been mistakenly included as an additional precaution.

In some cases, Wikipedia articles violating copyright are overwritten with new material as opposed to being simply deleted. In these cases, the revisions violating copyright are deleted from the page history, and a new page is moved over the violating material. *eeclassify.py* searches for cases in which a page move was detected by *eeanalyze.py*, but the comment field of the log indicates that the page was a copyright violation ("copyright", "copyvio", "g12", or "a8" appears in the log comments). In these cases, *eeclassify.py* updates the automatic analysis of the page to show both the page move and the copyright violation.

eeclassify.py found a list of articles whose talk pages appeared to be created before the articles themselves due to the Great Oops and UseMod KeptPages. It did this by checking if the timestamps of the first visible main and talk edits to a page were before 15:52 UTC on 25 February 2002.

Before the English Wikipedia upgraded to MediaWiki 1.5 in June 2005, all article titles and contents were encoded in ISO 8859-1 (nominally Windows-1252). This meant that many special characters, such as some accented letters, could not be used. [19] After the upgrade, many pages were moved to new titles with the correct diacritics. However, not all pages were correctly moved, leading to history fragmentation in several cases. *eeclassify.py* scans for this case and flags affected articles.

The program generated statistics showing the reasons why the talk pages of certain articles appear to be created before the articles themselves, which it wrote to standard output. Table 5.1 shows the statistics generated by *eeclassify.py*: the number of automatically analyzed articles with their corresponding

[1] The data in this .csv file could not be included due to length.

reasons.

Table 5.1: Numbers of articles classified by *eeclassify.py*, organized by reason

Reason	Number of Articles
Copyright violation	1,325
Copyright violation, but a new page was moved over the violating material	72
Likely moved by cut-and-paste, while talk page moved properly	20
Split history, with differences in capitalization or diacritics in the title	101
Affected by the Great Oops or UseMod KeptPages	360
Unknown reason (automatic analysis condition not met)	24,061

6. Analysis of results

Out of the 25,941 articles with the first visible edits to their talk pages appearing at least one day before those of the articles themselves, only 1,880 articles could be automatically analyzed. The reason that so few articles could be automatically analyzed is that there is a large number of unusual cases of page history inconsistency.

For example, in the case of “Paul tseng”, the creator of the article began writing it on their user page, a Wikipedia page that each user can create to describe themselves or their Wikipedia-related activities. Users can also create sandboxes in the user namespace, areas where they can experiment or write drafts of their articles. Users also have talk pages, which can be used for communication between users on the wiki. [20] Typically, these sandboxes are subpages of the user page. However, in this case, the creator of the “Paul tseng” article did not create a separate sandbox for the article, instead writing it directly on their main user page. When they completed the article, they moved both their user page which contained the article text, as well as their personal talk page, to “Paul tseng”. Clearly, the user had received messages from other users on the wiki before this move, so the talk page of “Paul tseng” contained personal messages addressed to the creator of the “Paul tseng” article. Upon discovering this, I reported the situation to a Wikipedia administrator, who split the talk page history, placing the user talk messages back in their appropriate namespace. [21] On the English Wikipedia, it is good practice to place a signature at the end of messages and comments, by typing four tildas (˜˜˜˜). [22] Signatures can contain the username of the commenter, links to their user or talk pages, and the timestamp of the comment in coordinated universal time (UTC). The talk page was created by SineBot, a bot that adds such signatures in case a user fails to do so. If a user fails to sign three messages in a 24-hour period, SignBot leaves a message on their talk page informing them about signatures, creating the user talk page if it does not already exist. [23] To make sure that no other similar cases have occurred, I checked if SineBot has created any other pages in the talk namespace. It has not, so this seems to be a unique occurrence.

Firefox has a built-in Wikipedia search feature. In old versions, entering “wp” (the Wikipedia search keyword) without a search term would redirect users to `https://en.wikipedia.org/wiki/%25s`. As a temporary workaround, a redirect was created to send these users to the Wikipedia main page. The associated talk page was used to discuss both the redirect and %s as a format string used in various programming languages. [24] The redirect has since been replaced with a disambiguation page; a navigation aid to help users locate pages with similar names. [25] The talk page has been preserved for historical reasons. Clearly, it contains edits older than those to the disambiguation page.

In the case of the “Arithmetic” article, the talk page was intentionally created before the article itself, so it does not indicate missing history. A user moved some discussion about the article from the “Multiplication” talk page to a new page, which would later serve as the talk page for the “Arithmetic” article. [26], [27] While it is definitely an unusual case, it all seems to add up in the end!

References

[1] Wikipedia. (11th May 2017). Wikipedia, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia&oldid=779855151 (visited on 11/05/2017).

[2] ——, (11th May 2017). Wikipedia:why create an account?, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Why_create_an_account%3F&oldid=779949646 (visited on 12/05/2017).

[3] ——, (22nd February 2017). Help:page history, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Help:Page_history&oldid=766814914 (visited on 11/05/2017).

[4] ——, (9th May 2017). Wikipedia:moving a page, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Moving_a_page&oldid=779647774 (visited on 12/05/2017).

[5] ——, (20th August 2017). Wikipedia:user access levels, [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:User_access_levels&oldid=796409520#Confirmed (visited on 20/08/2017).

[6] ——, (24th December 2016). Wikipedia:administrators' guide/fixing cut-and-paste moves, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Administrators%27_guide/Fixing_cut-and-paste_moves&oldid=756438110 (visited on 12/05/2017).

[7] ——, (27th August 2017). Usemodwiki, [Online]. Available: https://en.wikipedia.org/w/index.php?title=UseModWiki&oldid=797572191 (visited on 28/08/2017).

[8] B. Vibber. (25th Jun. 2003). Wikipedia:village pump/june 2003 archive 6, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Village_pump/June_2003_archive_6&oldid=577146248 (visited on 28/08/2017).

[9] Wikipedia. (31st March 2017). Wikipedia:phase ii software, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Phase_II_software&oldid=773132107 (visited on 29/08/2017).

[10] J. Hidders. (26th February 2002). Wikipedia:special pages bug reports, [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Special_pages_bug_reports&oldid=421916577#My_watchlist (visited on 28/08/2017).

[11] Wikipedia. (21st April 2017). Help:using talk pages, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Help:Using_talk_pages&oldid=776611301 (visited on 11/05/2017).

[12] B. Davis. (12th Jul. 2017). Labs and tool labs being renamed, [Online]. Available: https://phabricator.wikimedia.org/phame/post/view/59/labs_and_tool_labs_being_renamed/ (visited on 28/08/2017).

[13] Wikitech. (8th May 2017). Help:tool labs, [Online]. Available: https://wikitech.wikimedia.org/w/index.php?title=Help:Tool_Labs&oldid=1758703 (visited on 11/05/2017).

[14] Wikipedia. (5th April 2017). Wikipedia:namespace, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Namespace&oldid=773957607 (visited on 12/05/2017).

[15] M. D. T. et al. (2017). Date and time literals - mysql 5.7 reference manual, [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/date-and-time-literals.html (visited on 12/05/2017).

[16] Wikitech. (5th April 2017). Help:tool labs/grid, [Online]. Available: https://wikitech.wikimedia.org/w/index.php?title=Help:Tool_Labs/Grid&oldid=1755063 (visited on 01/05/2017).

[17] Wikipedia. (6th May 2017). Wikipedia:criteria for speedy deletion, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Criteria_for_speedy_deletion&oldid=779052095 (visited on 15/05/2017).

[18] A. Halfaker. (3rd May 2017). Mwxml 0.3.0 documentation, [Online]. Available: http://pythonhosted.org/mwxml/ (visited on 03/05/2017).

[19] Wikipedia. (1st Sep. 2017). Help:multilingual support, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Help:Multilingual_support&oldid=798440338#cite_note-1 (visited on 04/09/2017).

[20] ——, (24th August 2017). Wikipedia:user pages, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:User_pages&oldid=796983967 (visited on 05/09/2017).

[21] graham87. (29th April 2017). Logs related to paul tseng, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Special:Log&offset=20170429051557&limit=8&type=&user=Graham87&page=&tagfilter=&hide_thanks_log=1&hide_patrol_log=1&hide_tag_log=1&hide_review_log=1&month=5&year=2017 (visited on 29/04/2017).

[22] Wikipedia. (1st Sep. 2017). Wikipedia:signatures, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Signatures&oldid=798283702 (visited on 05/09/2017).

[23] Slakr. (14th November 2016). User:sinebot, [Online]. Available: https://en.wikipedia.org/w/index.php?title=User:SineBot&oldid=749437682 (visited on 05/09/2017).

[24] Wikipedia. (16th Jun. 2017). Talk:%s, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Talk:%5C%25s&oldid=785959123 (visited on 05/09/2017).

[25] ——, (30th August 2017). Wikipedia:disambiguation, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Disambiguation&oldid=798019104 (visited on 05/09/2017).

[26] ——, (). Arithmetic: Revision history, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Arithmetic&dir=prev&action=history (visited on 05/09/2017).

[27] ——, (). Talk:arithmetic: Revision history, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Talk:Arithmetic&dir=prev&action=history (visited on 05/09/2017).

Appendices

A. SQL Wrapper Scripts Submitted to Wikimedia Tool Labs

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id<=1000000 and rev_parent_id=0 and rev_page = page_id and (
    page_namespace=0 or page_namespace=1);"
```

eecollect1.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>1000000 and page_id<=3000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect2.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>3000000 and page_id<=4000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect3.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>4000000 and page_id<=5000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect4.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>5000000 and page_id<=6000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect5.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>6000000 and page_id<8000000 and rev_parent_id=0 and rev_page = page_id
     and (page_namespace=0 or page_namespace=1);"
```

eecollect6.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>8000000 and page_id<10000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect7.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>10000000 and page_id<=15000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect8.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>15000000 and page_id<=25000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect9.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>25000000 and page_id<=35000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect10.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>35000000 and page_id<=45000000 and rev_parent_id=0 and rev_page =
    page_id and (page_namespace=0 or page_namespace=1);"
```

eecollect11.sh

```
#!/bin/bash
sql enwiki -e "select page_title, rev_timestamp, page_namespace from page, revision
    where page_id>45000000 and rev_parent_id=0 and rev_page = page_id and (
    page_namespace=0 or page_namespace=1);"
```

eecollect12.sh

B. Post-Processor Source Code

```
# Imports
import argparse
from dateutil import parser as dateparser
# Set up command-line arguments
parser = argparse.ArgumentParser()
parser.add_argument("file",help="the tab-separated output file to read")
parser.add_argument("-w","--window",type=int,help="the time window to scan (the minimum
    amount of time between the creation of the talk page and the article required for
    inclusion in the output list), default is 86,400 seconds (one day)",default=86400)
args=parser.parse_args()
# Declare dictionaries
main={} #map of pages in namespace 0 (articles) to the timestamps of their first
    revision
talk={} #map of pages in namespace 1 (article talk pages) to the timestamps of their
    first revision
# Declare the chunk counter (count of number of times the header row appears)
chunk=0
# Read in file
with open(args.file) as fin:
    for line in fin:
        #Split fields
        t=line.strip().split("\t")
        # Check line length
        if len(t) != 3:
            print("Warning: The following line is malformed!:")
            print(line)
            continue
        if t[0] == "page_title" and t[1] == "rev_timestamp" and t[2] == "page_namespace"
    :
            #New chunk
            chunk+=1
            print("Reading chunk " + str(chunk) + "...")
            continue
        #Is the page already in the dictionary?
        if t[0] in main and t[2]=="0":
            if int(t[1])<int(main[t[0]]):
                main[t[0]]=t[1]
            else:
                continue
        if t[0] in talk and t[2]=="1":
            if int(t[1])<int(talk[t[0]]):
                talk[t[0]]=t[1]
            else:
                continue
        # If not, add it.
        if t[2] == '0':
            main[t[0]]=t[1]
        elif t[2] == '1':
            talk[t[0]]=t[1]
print("Data collected, analyzing...")
matches=[]
for title,timestamp in main.items():
    if title not in talk:
        #No talk page, probably a redirect.
        continue
    elif dateparser.parse(main[title]).timestamp()-dateparser.parse(talk[title]).timestamp()>=args.window:
        matches.append(title)
print("Analysis complete!")
print("The following " + str(len(matches)) + " articles have visible edits to their talk pages earlier than the articles themselves:")
for match in matches:
    print(match.replace("_"," "))
print("Generating CSV report...")
import csv
with open("eeprocessed.csv","w") as cam:
    writer=csv.writer(cam)
    writer.writerow(("article","first main","first talk"))
    for match in matches:
        writer.writerow((match.replace("_"," "),main[match],talk[match]))
print("Done!")
```

eeprocess.py

C. Analyzer Source Code

```python
import mwxml
import argparse
import csv
from dateutil import parser as dateparser
from collections import defaultdict
# Set up command-line arguments
parser = argparse.ArgumentParser()
parser.add_argument("file",help="the .csv output file (from eeprocess.py) to read")
parser.add_argument("dump",help="the uncompressed English Wikipedia pages-logging.xml
    dump to check against")
args=parser.parse_args()
print("Reading " + args.file + "...")
with open(args.file) as fin:
    reader=csv.reader(fin)
    #Do we have a valid CSV?
    head=next(reader)
    if head[0] != "article" or head[1] != "first main" or head[2] != "first talk":
        raise ValueError("invalid .csv file!")
    #valid CSV
    #Create main and talk dicts to store unix times of first main and talk revisions
    main={}
    talk={}
    for row in reader:
        if row[0] in main or row[0] in talk:
            raise ValueError("Duplicate detected in cleaned input!")
        main[row[0]]= dateparser.parse(row[1]).timestamp()
        talk[row[0]]= dateparser.parse(row[2]).timestamp()
print("Read " + str(len(main)) + " main, " + str(len(talk)) + " talk. Checking against "
     + args.dump + "...")
with open(args.dump) as fin:
    d=mwxml.Dump.from_file(fin)
    #Create reasons, dict mapping article names to reasons why their talk pages appear
    to have edits before the articles themselves.
    reasons={}
    #Create comments, dict mapping article names to log comments.
    comments={}
    #Create moves, defaultdict storing page moves for later analysis
    moves=defaultdict(dict)
    for i in d.log_items:
        if len(main) == 0:
            break
        try:
            if (i.page.namespace == 0 or i.page.namespace == 1) and i.params in main and i.action.startswith("move"):
                moves[i.params][i.page.namespace]=(i.page.title,i.comment)
            if (i.page.namespace == 0 or i.page.namespace == 1) and i.action == "delete" and i.page.title in main:
                c=str(i.comment).lower()
                if ('copyright' in c or 'copyvio' in c or 'g12' in c or 'a8' in c):
                    reasons[i.page.title]="copyright"
                    comments[i.page.title]=i.comment
                    print("Copyright violation: " + i.page.title + " (" + str(len(reasons)) + " articles auto-analyzed, " + str(len(main)) + " articles to analyze, " + str(len(moves)) + " move candidates)")
            if i.params in moves and i.params in main:
                del main[i.params]
            if i.page.title in reasons and i.page.title in main:
                del main[i.page.title]
        except (AttributeError,TypeError):
            print("Warning: malformed log entry, ignoring.")
            continue
    print(str(len(moves)) + " move candidates, analyzing...")
    for article,movedict in moves.items():
        if 1 in movedict and 0 not in movedict:
            reason="move from " + movedict[1][0]
            comment=movedict[1][1]
            reasons[article]=reason if article not in reasons else reasons[article]+", then " + reason
            comments[article]=comment if article not in comments else comments[article]+", then " + comment
print("Writing move candidate csv...")
with open("eemoves.csv","w") as cam:
    writer=csv.writer(cam)
    writer.writerow(("from","to","namespace","comment"))
    for article,movedict in moves.items():
        for namespace,move in movedict.items():
            writer.writerow((move[0],article,namespace,move[1]))
print(str(len(reasons)) + " pages auto-analyzed, generating CSV...")
with open("eeanalysis.csv","w") as cam:
    writer=csv.writer(cam)
    writer.writerow(("article","reason","comment"))
    for page, reason in reasons.items():
        writer.writerow((page, reason,comments[page]))
print("Done!")
```

eeanalyze.py

D. Classifier Source Code

```
import csv
import argparse
from dateutil import parser as dateparser
from collections import Counter,defaultdict
# Requires unidecode from PyPI
import unidecode
parser = argparse.ArgumentParser()
parser.add_argument("eeprocessed",help="the .csv output file (from eeprocess.py) to read")
parser.add_argument("eeanalysis",help="the .csv output file (from eeanalyze.py) to read")
args=parser.parse_args()
# Declare main and talk dicts, mapping article names to timestamps of their first main and talk edits respectively
main={}
talk={}
# Declare reasons and comments dicts, mapping article names to reasons and comments (from eeanalyze)
reasons={}
comments={}
# Read in CSVs
with open(args.eeprocessed) as fin:
    reader=csv.reader(fin)
    #Skip the header
    next(reader)
    #read in main and talk dicts
    for row in reader:
        main[row[0]]=row[1]
        talk[row[0]]=row[2]
with open(args.eeanalysis) as fin:
    reader=csv.reader(fin)
    #Skip the header
    next(reader)
    #Read in reasons, filtering out revision deletion based on the comment field (I'm sure there's a better way, but the log_deleted field in the db which determines if a deletion is page or revision doesn't properly exist for all deletes or mwxml doesn't see it in all cases)
    for row in reader:
        if "rd1" not in row[2].lower():
```

```python
            reasons[row[0]]=row[1]
            comments[row[0]]=row[2]
print("Read " + str(len(main)) + " main, " + str(len(talk)) + " talk, and " + str(len(reasons)) + " articles that were automatically analyzed (not counting revision deletions; they are false positives for my purposes).")
# Fix misclassified copyvios
for article,reason in reasons.items():
    c=comments[article].lower()
    if "copyright" not in reason and ("copyright" in c or "copyvio" in c or "g12" in c or "a8" in c):
        reasons[article]="copyright (" + reasons[article] + ")"
# Classify articles affected by the Great Oops (15:52, 25 February 2002 UTC) and UseMod keep pages
reasons.update({a:"great oops" for a,ts in main.items() if dateparser.parse(ts).timestamp() <= 1014652320 and dateparser.parse(talk[a]).timestamp() <= 1014652320})
comments.update({a:"" for a,r in reasons.items() if r == "great oops"})
# find split histories (pages with identical names except caps and diacritics)
acounter=Counter([unidecode.unidecode(a).lower() for a in main])
splitkeys=[k for k,v in acounter.items() if v>1]
splithist=defaultdict(dict)
for a,ts in main.items():
    k=unidecode.unidecode(a).lower()
    if k in splitkeys:
        splithist[k][dateparser.parse(ts).timestamp()]=a
for a,m in splithist.items():
    t=sorted(m.keys())
    reasons[m[t[0]]]="split from " + m[t[1]]
    comments[m[t[0]]]=""
# Add unknowns
reasons.update({a:"unknown" for a in main if a not in reasons})
comments.update({a:"" for a,r in reasons.items() if r == "unknown"})
# Write eefinal.csv
print("Writing eefinal.csv...")
with open("eefinal.csv","w") as cam:
    writer=csv.writer(cam)
    writer.writerow(("article","first main","first talk","reason","comment"))
    for a in sorted(reasons.keys()):
        if reasons[a]=="unknown" and unidecode.unidecode(a).lower() in splitkeys:
            continue
        writer.writerow((a,main[a],talk[a],reasons[a],comments[a]))
print("CSV written. Generating stats...")
copyvios=0
```

```
copymoves=0
talkmoves=0
histsplits=0
oopses=0
unknowns=0
for a,r in reasons.items():
    if r == "copyright":
        copyvios+=1
    elif r.startswith("copyright ("):
        copymoves+=1
    elif r.startswith("move from"):
        talkmoves+=1
    elif r.startswith("split from"):
        histsplits+=1
    elif r == "great oops":
        oopses+=1
    elif r == "unknown":
        unknowns+=1
print(str(copyvios) + " articles were copyright violations.")
print(str(copymoves) + " articles were copyright violations, but a new page was moved
    over the violating material.")
print(str(talkmoves) + " articles were likely moved by cut and paste, while their talk
    pages were moved properly.")
print(str(histsplits) + " articles have split history, with differences in
    capitalization or diacritics in the title.")
print(str(oopses) + " articles were affected by the Great Oops or UseMod keep pages.")
print(str(unknowns-histsplits) + " articles could not be automatically analyzed.")
print("Done!")
```

eeclassify.py