Open Secrets: Investigating systemic biases in Wikipedia, the free online encyclopedia

Eamon Ma, Hisbaan Noorani, Rachel Xie, Philip Harker

2021-05-04

1 Introduction

As a free online encyclopedia run by volunteer collaborators, edits to Wikipedia articles form the backbone of the website. Without its open nature, Wikipedia would not be what it is today. However, this does not guarantee greatness — for the same reasons it is a symbol of communal knowledge and free information, Wikipedia is inherently biased.

Most notably, it is biased towards the countries that articles are edited most in, and the dominant cultures in those countries. "[Our] research does show that most editors to Wikipedia come from the United States and Western Europe. And, as of 2020, our survey data indicate that fewer than 1% of Wikipedia's editor base in the U.S. identify as Black or African American. Considering these data, we can say with certainty that we are missing important perspectives from the world that Wikipedia strives to serve." (Uzzell, 2021). Alongside an abundance of information of interest to Western society, there is a distinct lack of information from and for marginalized groups.

The coverage of knowledge is not spread evenly across all of Wikipedia's articles. It varies based on what is desired and what is available, and often, the most available information on English Wikipedia[1] concerns men from "developed" countries and their interests.

Only 17.82% of Wikipedia's biographies are about women. This isn't merely an issue of Wikipedia's editors specifically being unreliable, it is an issue of there not existing reliable sources in the wider Internet available for editors to feel confident creating articles about topics like women in science (Erhart, 2018). In this way, Wikipedia can be thought of as a representation of society's general knowledge base.

Our research question is, **"How can we use connections / links between Wikipedia articles to determine areas where our collective knowledge is lacking?**

2 Description of dataset(s)

All of the data used in this project is open and available at https://en.wikipedia.org/wiki/Wikipedia:Database_download#Where_do_I_get_it?. (Wikimedia, 2021b)

The sole source of data is the Wikipedia pages, available here. (Wikimedia, 2021a)

We used the most recent dataset available to us at the time we first began this project, from (2021-01-01).

This dataset contains every single English Wikipedia page, accurate to 2021-01-01, compiled into one XML file. The singular XML datasets were downloaded via torrent due to its large size, to help reduce server load.

3 Computational overview

3.1 The data

The essence of the Web is captured in this dataset—pages identified with uniform resource locators, interconnected with hyperlinks within the pages. This structure naturally lends itself to the format of a graph: the very concept of a network *is* a graph. Then, it is entirely instinctive to represent a website as a graph: its interconnected articles are the nodes of

[1] For the purposes of this project, only English Wikipedia was considered – demographics and articles of interest will vary between languages, so we focused solely on English articles to avoid confusion.

this graph, with edges—hyperlinks—connecting them. On a more abstract note, knowledge is inherently connected in graph-like structures with varying intensity. Perhaps elementary algebra is a necessary prerequisite for the study of calculus, and perhaps the invention of ice cream is connected to the early life of Leonardo DiCaprio. No matter where we look, we can almost always find connections among pieces of knowledge. Therefore, it is sensible to represent the biggest open compilation of human knowledge as a graph. To limit complexity, we restrict our exploration to English Wikipedia.

3.2 The processing

The overarching goal here is to build a graph of articles on Wikipedia, then perform analyses. The immensity of the dataset—over 81 gigabytes uncompressed—poses a challenge that would otherwise not be present with a smaller dataset. To perform useful analyses with reasonable computing times, we break the data transformation of the large XML file into small steps:

1. Create an index of line numbers of the boundary of the articles. We save this index after creation, as the generation process takes approximately thirty minutes on a modern computer.

2. Partition the large XML into smaller subsets. We partitioned the file into 80 partitions, though the number of choice can be further fine-tuned. This has a number of advantages, notably the ability to parallelize the processing of the files.

    - To partition the large file, we generate a list of partition points. This is achieved by splitting the file at roughly equal line numbers, at the boundary between pages.

3. Process the partitions. In conventional sequential processing, the advantage of partitions is shown in error resistance: if the processing fails at a given partition, the previous work is not lost. We can simply restart before the file which failed. In concurrent processing, we see the benefits manifest themselves in a stunning manner: the production of graph files from the entire 80 gigabytes takes only approximately 12 minutes.

3.2.1 Indexing

Indexing serves to mark the dividing lines for subsequent partitioning. Essentially, the alternative is to iterate through the XML and match `<page>` tags at runtime, so it can be advantageous for this index to be computed in advance. Then, subsequent parties working on the data can decide themselves where to partition based on their system and needs. The function is simple, but can be a time and headache saver. Essentially, it looks for an opening page tag in each line; if such a tag is present, then it is saved as a boundary in `line_numbers`. We make the assumption safely because the XML dump is computer-generated, thus consistent, well formed, and syntactically correct in this regard.

3.2.2 Partitioning

Partitioning serves to allow concurrency in the program, greatly speeding up practical running time of analysis functions on modern operating systems and CPUs which support concurrent processing. Before partitioning, we produce a list of numbers at which the file is split. The partitions are produced by iterating through the master XML file, incrementing a line counter, accumulating lines, and writing accumulated lines when a partition point is reached.

3.2.3 Processing

Processing the dataset—and by extension the partitions is run through a concurrent context manager, specifically `ProcessPoolExecutor`. This is a high-level library that enables the execution of asynchronous code with blocking operations (**concurrent**). It differs from `ThreadPoolExecutor`, a similar library, in that `ProcessPoolExecutor` runs tasks with their own child process, while `ThreadPoolExecutor` runs tasks within threads in the main process (**difference**). Specifically within the context managers, a process is submitted to a queue to be created for each individual partition. `process_partition` is called on each partition, and the results are saved to disk. For each partition, two TAB SEPARATED VALUES files, or TSV, are created. The first contains rows of articles in each partition and associated information such as whether or not it is a redirect, its character count, and the time delta between the last edit and 2021-01-01; the second contains rows of links each article contains. Finally,

we collapse the redirect articles. When visiting Wikipedia, occasionally there is small text below the article title indicating that one was *Redirected from* ___. These are accomplished by articles whose sole purpose is to redirect, and so we must take these into account.

3.2.4 Extracting information from wikitext

We saw experimentally that an available library takes on average about 30 seconds on a modern computer to parse through an XML file approximately one million lines long. (Note that though the parser is written for parsing wikitext, in contrast to XML, the mere *presence* of markup does not affect the functionality). Extrapolating, the 1.3 billion line Wikipedia dump would take about $1300 \cdot 30$ seconds, or under 10 hours.

Diving into the code for `wikitextparser`, we see the project relies on regular expressions as a primary method of string manipulation. Unfortunately, this tool falls short for large files, and performs functionality we do not need. Instead, we decided on a hybrid approach: match all the wikilinks using a regular expression, then conduct further processing on each wikilink with string operations. In combination with performing fewer computations, we see a significant improvement over the library: on that same million line file, the string operation implementation takes under 1.3 seconds over a 10 run average. This is more than an order of magnitude faster than the library's implementation—about 20-23 times faster, to be precise. This allows us to reduce the projected computing time for producing the graph from 10 hours to about half an hour.

Subsequent information extracted from a given article's wikitext also use similar tricks of micro optimization. For instance, we notice that the length between the ending text element and the ending page element is constant. Then, we only need to find the beginning text tag.

3.3 Score

We calculate a score for each article with the following equation, proceeding from defined attributes: `self.score = (self.char_count * len(self.neighbours)) / (math.log(self.last_edit) + 10)` we have determined from our investigation, articles that are less connected often have lower character counts, fewer links (neighbours), and a longer time since their last edit. We prioritized connectedness more and decided a higher score indicates greater connectedness, and a greater character count, a greater number of neighbours, and a shorter time since last edit should in turn produce a higher score. Therefore, we made the number of links and the character count into multiplicative factors in our expression. Additionally, a smaller time since last edit should result in a greater score, hence the use of `self.last_edit` in the denominator.

4 Obtaining the dataset and running the program

By virtue of the size of the dataset we are working with, obtaining the dataset and running the program must be done somewhat differently.

1. Clone the project from Markus.
2. Open the project root in the command line—paths mentioned hereafter are relative to project root. We highly recommend using a Python virtual environment:
   - Create a venv: `python -m venv venv`
   - On Windows cmd, run the bat script to activate the venv: `venv\Scripts\activate.bat`
   - In shell, source the venv: `source venv/bin/activate`

   Install requirements using pip3: `pip3 install -r requirements.txt`
3. Install the project package with "`pip3 install -e .`"
4. Due to the impracticality of distributing a 17GB file on servers at the University of Toronto, if you wish to compute on the given original dataset, you may use your preferred BitTorrent client to obtain the file from Wikimedia's meta page, ensuring that the file downloaded is `enwiki-20210101-pages-articles-multistream.xml.bz2` on `nicdex.com` (17.79 GB).
   - Uncompress the original dataset to `data/raw/enwiki-20210101-pages-articles-multistream.xml`

5. If you wish to test the program on smaller datasets, download the provided XML files: https://uoft.in/k0i8p and place them accordingly:

   ```
   data/raw/reduced/hundredk.xml
   data/raw/reduced/million.xml
   ```

   Note that the first four files are produced with the shell command `head -n (m x) > word(m).xml`, where x is the original file, where m in [1000, 10000, 100000, 1000000], and word(m) is a function that maps m to the respective item in [k, ninepointthreek, hundredk, million]. Subsequently, if not already present at the base of the page (viewed with `tail -n 5 m.xml`), run `echo "</page>" >> word(m).xml` and `echo "</mediawiki>" >> m.xml` to complete the dataset. The latter two files are produced by manually extracting an XML page, and extracting wikitext, respectively.

6. If you wish to produce graphs for these truncated datasets, manually obtain the precise line count with `wc -l word(m).xml` in shell, or the appropriate tool on Windows. Then, edit the constant `FILE_LINE_COUNT` at the top of `wikigraph/partition_data.py` to the number obtained.

For best results, we strongly recommend at least 16GB of RAM with adequate memory paging schemes (swapfile/swap partition) set up for your operating system. We recommend dedicating as much memory to these tasks as possible due to their high storage and computational load.

5 Justification and discussion of changes

The provided teaching assistant feedback was considered carefully. A few of the provided suggestions had already been addressed in the initial project proposal, such as the suggestion to download and process the data once, storing it for future use to avoid having to load from this enormous dataset every time. It was also suggested that we reduce the size of the dataset we were working with to avoid having to work with the enormous dataset representing every single one of Wikipedia's articles; however, it is necessary to leave the dataset as it is because there was no clear way to separate out just a smaller group of articles—we would have had to find some group of articles that link only to each other and never to any articles outside of that group, and doing that manually would be incredibly time-consuming, if not impossible. Indeed, this defeats the purpose of our research question.

The major change from our proposal was our choice to change the metrics we were considering to find "underconnected" articles in the dataset. To get view counts, we would have had to download another 3.5 TB of data, which is far too much for the relatively limited applicability of this metric. Looking at how much a page is viewed might tell us how often people are searching up certain topics, but that would not necessarily provide interesting results specific to Wikipedia. We also decided to consider character counts instead of word counts because this was simpler to get from the raw data and could be done more efficiently. This would, overall, increase the base numbers we get for this metric, but higher character counts generally also mean higher word counts, so the end result (finding the shortest articles on Wikipedia) is the same.

We also stated that we would be using the Python library `NetworkX`, but did not use it in the end as `NetworkX` was quite slow and our dataset was quite large—prioritizing efficiency was a necessity. We neglected to use `pandas` as our method for processing the data never led to us needing it. The new library we chose to visualize data with was `PyVis`. While it is not exceptionally fast, it does provide a loading screen and somewhat more visually appealing interactive windows for the user to look at. We would have liked to find a faster library for visualization, but finding a free and well designed one for Python was extremely difficult.

6 Discussion

There is necessarily error in our results because Wikipedia by its very nature is open source. Even with its guidelines and standards, there will be syntax errors and inconsistencies between articles. There are mistakes caused by editor error that our program simply cannot account for. However, we

believe the dataset is large enough that the overall results are still worth analyzing. From the first one million lines of the data, there are 4279 articles. Within those, we found 16 incorrect links as a result of Wikipedia editor errors–a 0.3% error rate.

The results of our computational exploration helped answer our question. We found that while it is possible for short articles to indicate a lapse in our collective knowledge, it can also simply be a result of the article being extremely new and not yet fully polished. There is, in fact, a list of pages that are too new to have any links provided by Wikipedia itself: https://en.wikipedia.org/wiki/Category:All_dead-end_pages. By manually examining the results, we found that while the general statistics from our introduction might be true (we are unsure of the precise research that led to those conclusions) there are certainly exceptions, where a woman in science, for example, actually had a higher score than men involved in a similar field around the same time (see, for example, Mary the Jewess as compared with Stephanus of Alexandria). There were also several articles that pertain to ancient and / or non-Western history that were found to have low scores — this falls in line with expectations from our research question, namely that English Wikipedia's knowledge base focuses mostly on the affairs of white men in Western society. A graph is attached here: https://uoft.in/k0i8p.

For future research, we might extend the functionality of the program to sort the articles it finds that score the lowest on a given metric by topic, i.e. , when considering the articles with the lowest character counts, it might be interesting to then programmatically group those articles based on what they're about. This would save time on analysing how the metrics compare, off-loading the work of sifting through the returned results, and might allow us to explore more complex combinations of variables to better understand how different attributes of a Wikipedia article correlate to its representation of how much knowledge we have.

Interestingly, the broken link situation described above can also be a way to contribute to the Wikipedia community. We could modify our code just slightly so that we obtain the database dump whenever it is released, find any broken links, and fix them. If it is intended to be a continuous effort, we can even write further functions to aid in this discovery.

References

Erhart, E. (2018). Why didn't Wikipedia have an article on Donna Strickland, winner of a Nobel Prize? *Medium*. https://medium.com/freely-sharing-the-sum-of-all-knowledge/why-didnt-wikipedia-have-an-article-on-donna-strickland-winner-of-a-nobel-prize-dff8e518daaa

Uzzell, J. (2021). Who tells your story on Wikipedia. *Medium*. https://medium.com/freely-sharing-the-sum-of-all-knowledge/who-tells-your-story-on-wikipedia-6d6c29f45028

Wikimedia. (2021a). Data dump torrents [Data set]. *Wikimedia Meta-Wiki*. https://meta.wikimedia.org/wiki/Data_dump_torrents#English_Wikipedia

Wikimedia. (2021b). Wikimedia Downloads [Data set]. *Wikimedia*. https://dumps.wikimedia.org/