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

Eamon Ma, Hisbaan Noorani, Rachel Xie, Philip Harker

2021-04-16

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 being 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 first of the two main sources of data will be the Wikipedia pages, available here. (Wikimedia, 2021a)

One of the most recent datasets when we began this project was used (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 was downloaded via torrent due to its large size, to help reduce server load.

3 Computational overview

As people who care about efficiency, we don't like to do more work than necessary. That is why we were ecstatic to find the Python library `wikitextparser` to do work for us—the most important being extracting links from a given article. Unfortunately, this functionality of the library was not written with performance as its highest priority. We can see experimentally that the library takes on average about 30

[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.

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.

These authors believed they could do better—and do better they did. 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. Immediately, we see a significant improvement: 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 from 10 hours to about half an hour—quite the improvement!

4 Instructions for Obtaining Dataset and Running Program

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. If you wish to compute on the given original dataset, 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 and place them accordingly:

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

6. f

5 Justification and Discussion of Changes

The TA feedback was, in polite terms, useless. A few of the provided suggestions had already been addressed in the ini-

tial project proposal. It was 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, but we had to leave the dataset as it was 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.

The major change from our proposal was our choice to change the metrics we were considering to find "under-connected" 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 is not necessarily specific to Wikipedia. We also decided to consider character counts instead of word counts because this was simpler to get from the raw dataset and more efficient. 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 chose [PHILIP WHICH LIBRARY ARE WE USING] instead 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.

6 Discussion

There is error in our results because Wikipedia by its very nature is open source and even with its guidelines and standards, there will be syntax errors and inconsistencies in articles. There are things 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 million lines, there are 4279 articles. Within those, we have 16 incorrect links, as a result of Wikipedia editor errors. That is a 0.3% error rate, which was not our fault to begin with.

OUR CODE IS IMPECCABLE; PEOPLE ARE JUST DUMB.

The results of our program show...???

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 group those articles based on what they're about.