



Web Mining - Wikilosophy

Raed Abdennadher, Ludovic Gindre & Edward Ransome



Initial observation

“Wikipedia trivia: if you take any article, click on the first link in the article text not in parentheses or italics, and then repeat, you will eventually end up at “Philosophy””

Context

- Previous quote first appeared on the alt-text of the xkcd webcomic #903
- As of February 2016, 97% of all articles lead to Philosophy in this manner

WIKIPEDIA
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

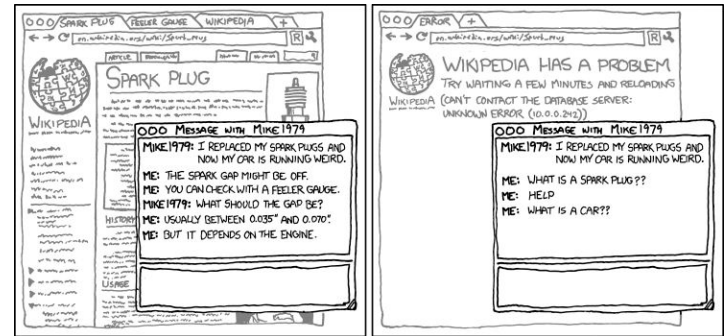
[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

← Try it!



WHEN WIKIPEDIA HAS A SERVER OUTAGE, MY APPARENT IQ DROPS BY ABOUT 30 POINTS.

<https://xkcd.com/903/>



Why does this happen?

- Most pages start by describing the topic of the page
- Topics naturally get broader as they contain multiple other topics
- Eventually we reach the widest reaching pages, such as Mathematics, Science, and Philosophy



Objectives

- Crawl the entirety of the English Wikipedia in this manner (code should be easily adaptable to other languages)
- Calculate an up-to-date estimate of how many pages lead to Philosophy
- Calculate connected components sizes
- Estimate average distance from a random page to Philosophy
- Graph visualisation if feasible



Data

- Wikipedia regularly blocks IP addresses performing excessive crawling (not viable for an entire site crawl)
- A downloadable database of all Wikipedia pages is available as XML
- Contains an index with byte offsets allowing to decompress 100 page sections at a time
- Approximately 16GB in size and 200MB for the index

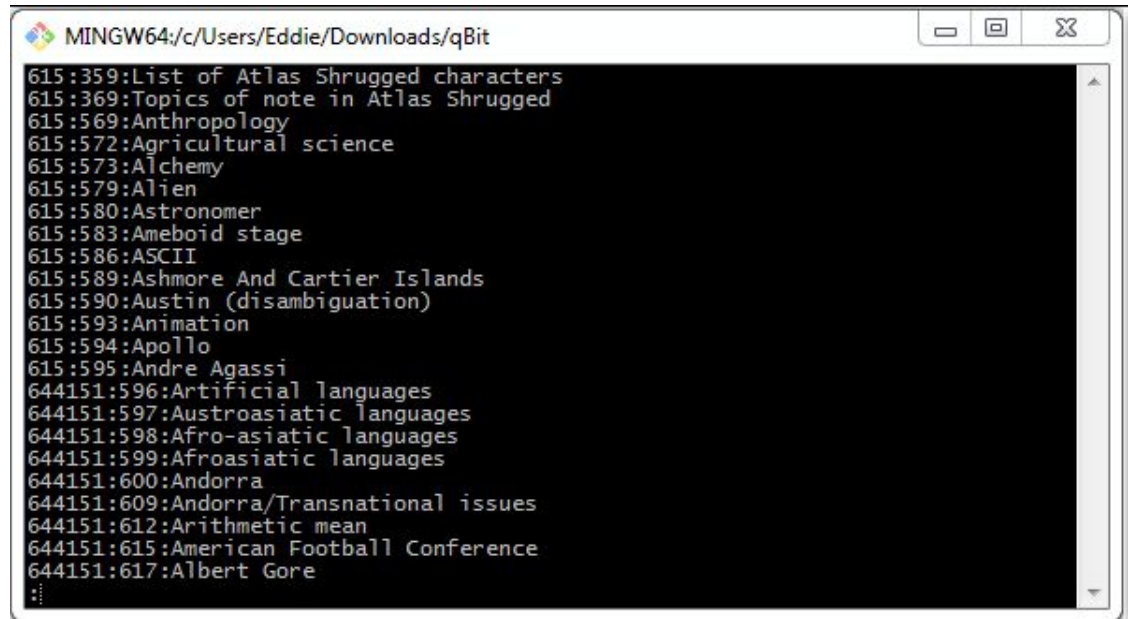
Please do not use a web crawler

Please do not use a [web crawler](#) to download large numbers of articles. Aggressive crawling of the server can cause a dramatic slow-down of Wikipedia.

Kind disclaimer on the Wikipedia Offline Dump page
https://en.wikipedia.org/wiki/Wikipedia:Database_download

Index structure

- Byte offset
- Page ID
- Page Title



```
MINGW64/c/Users/Eddie/Downloads/qBit
615:359:List of Atlas Shrugged characters
615:369:Topics of note in Atlas Shrugged
615:569:Anthropology
615:572:Agricultural science
615:573:Alchemy
615:579:Alien
615:580:Astronomer
615:583:Ameboid stage
615:586:ASCII
615:589:Ashmore And Cartier Islands
615:590:Austin (disambiguation)
615:593:Animation
615:594:Apollo
615:595:Andre Agassi
644151:596:Artificial languages
644151:597:Austroasiatic languages
644151:598:Afro-asiatic languages
644151:599:Afroasiatic languages
644151:600:Andorra
644151:609:Andorra/Transnational issues
644151:612:Arithmetic mean
644151:615:American Football Conference
644151:617:Albert Gore
:
```



Graph generation

- Each page represents a graph vertex
- Easy to create an edge list using the index
 - Unzip 100 pages
 - For each page, find the first hyperlink not in italic or parentheses
 - Place the start and end vertices separated by a tab character in a file
- Size of edge list should around the same size as the index since they both contain one line per page
- Must only be done once (unless using a different dump or language)



Technologies

- Java
- Apache Commons Compress API for decompression
(Allows the use of an offset)
- Neo4j (Allows graph analysis and importation using an edge list)





Expected result

- A large connected component that includes most of the pages that do not lead to Philosophy
 - We expect another large science-related page
- A large number of authorities linking to the Philosophy page (the graph should have a tree structure leading to Philosophy)
- A certain number of dead-ends, pages with no valid hyperlink leading to a new page



Risks and problems

- Large graph size might make visualisations unfeasible
- Since the 2016 estimate, there are a lot more pages that do not lead to Philosophy (Based on random tests we performed)
- Group has no experience with Neo4j or Apache Commons Compress API



Planification

- Decompress XML pages using a Java library (By 10.05)
- Parse the XML to find the first valid hyperlink (By 10.05)
- Create graph and import to Neo4j (By 17.05)
- Analyse graph using Neo4j and aggregate interesting statistics (By 24.05)
- Finish project, complete report, add any additional features like visualisations (By 07.06)
- Finishing touches for project end date (10.06)