Visualizing Wikipedia as a Graph

By Samudra Neelam Bhuyan

Table of Contents

Goals

The goal of this project is to create a new way for users to browse Wikipedia, using a knowledge graph that makes transparent the naturally occurring relationships between different topics.

"Naturally occurring relationships" are the ones that have emerged in the content of Wikipedia without any human intentions, as contrasted with the categories that are maintained by Wikipedia editors themselves. This underlying knowledge graph is built on incoming links to the articles themselves – a sort of basic PageRank algorithm built to analyze the Wiki content.

Interface Goals

The visualization is aimed at two primary tasks:

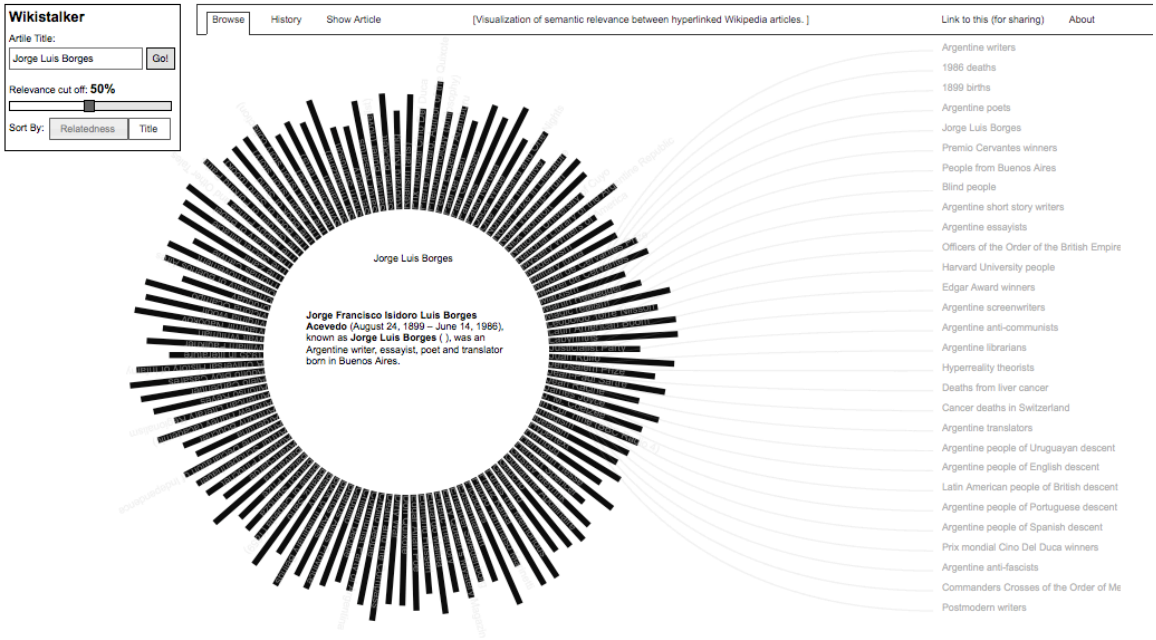
1. Discovering topics that are related to the current topic, and browsing to them. More specifically, to answer the question "for which other important topics is this topic important".

2. Drilling down from one topic, and discovering sub topics. More specifically, to answer the question “which are the most important topics that this topic links to”

Related Work

WikiStalker (http://sepans.com/wikistalker)

Wikistalker illustrates the relations between different things by visualizing the semantic relevance between the inter-connected structure of their Wikipedia entry articles. It uses Wikipedia Miner which focuses on the outgoing links and applies machine learning to get at the semantic relevance between articles.

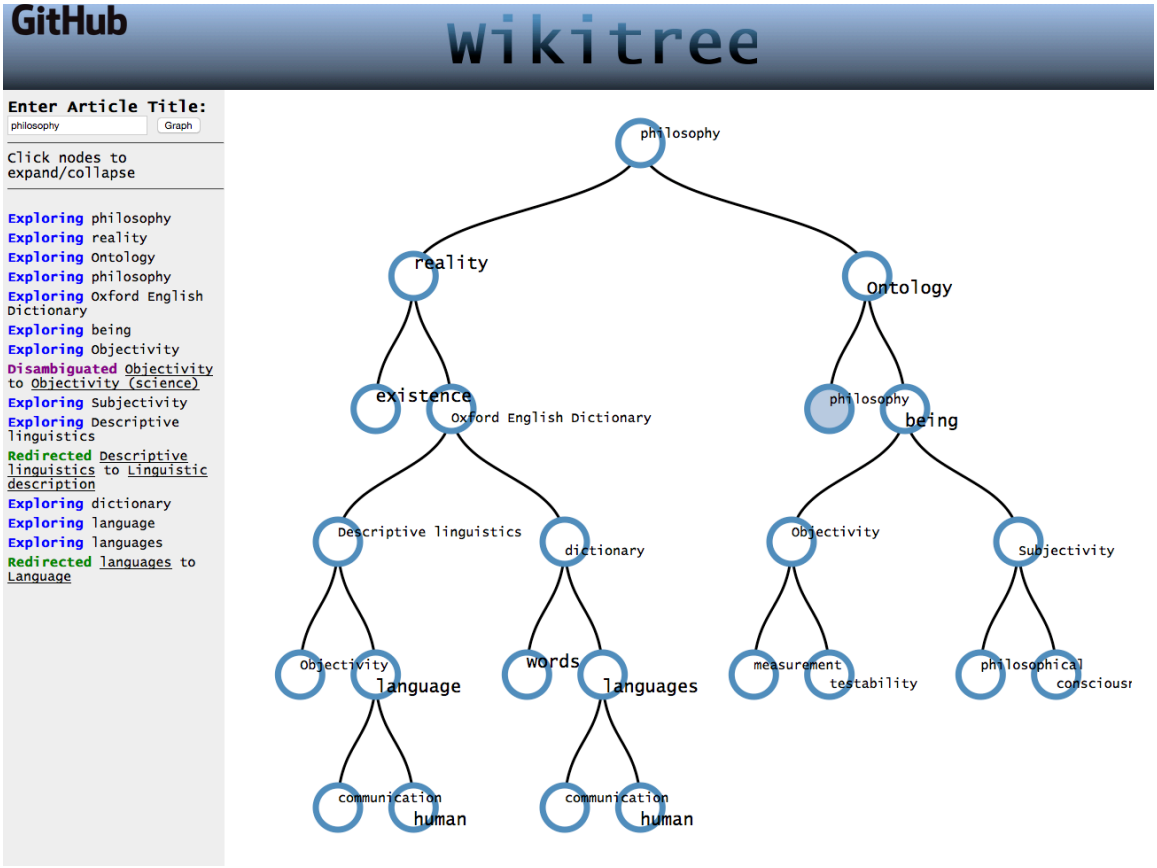


WikiStalker looks at the outgoing links from a page and calculates their relatedness according to the semantic relevance of the two articles. It then visualizes this relationship, by varying the length of the bar to represent the semantic relevance of the two articles.

While this visualization hints at the underlying structure of the knowledge graph, it is quite difficult to actually use because of the presentation of the information itself. Those black bars around the central topic are supposed to be other topics, but it is extremely unreadable. Most people in the usability tests don’t even realize that there are “readable” labels to the bars.

Wikitree (http://bergeron.im/wikitree/)

Wikitree graphs connections between Wikipedia articles using the MediaWiki API. It extracts the first two links in each Wikipedia article and graphs the resulting tree of information using the D3 library.

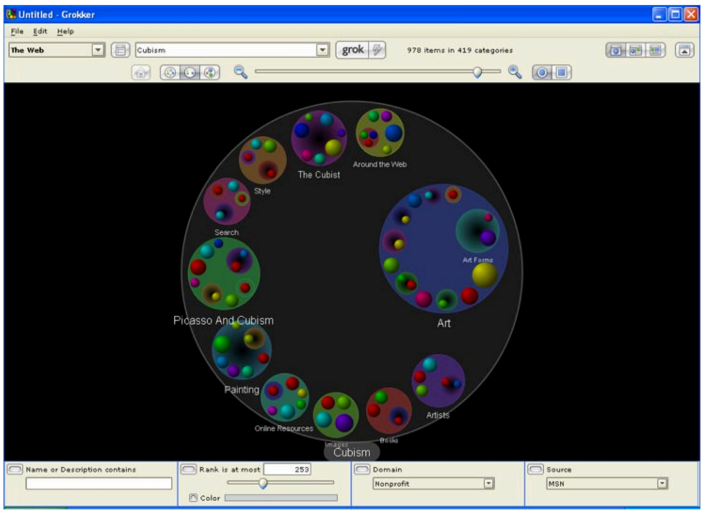


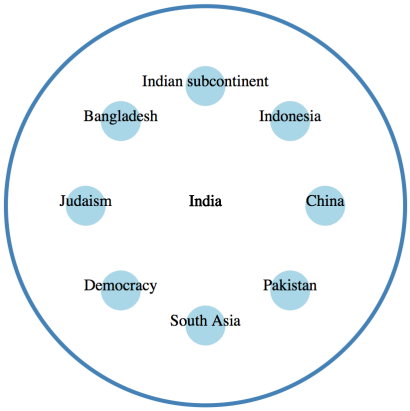
This project uses the idea of a graph, but it limits itself to only the first two outgoing links in an article. Because of this, sometimes the graph does not make sense. E.g. “Reality” is linked to “Oxford English Dictionary”, which does not make much sense until you realize that that link is a part of a citation note for the definition of “Reality”.

By relying solely on outgoing links, this also fails to look at the structure of a graph with nodes that have incoming relationships. That information is a valuable signal that illustrates the topic’s importance in the graph.

Grokker

Grokker was a visual search engine, which displayed search engines as a series of categories set in a circular map.





This visual nested representation of categories was inspiration for the Wikigraph visualization for outgoing links.

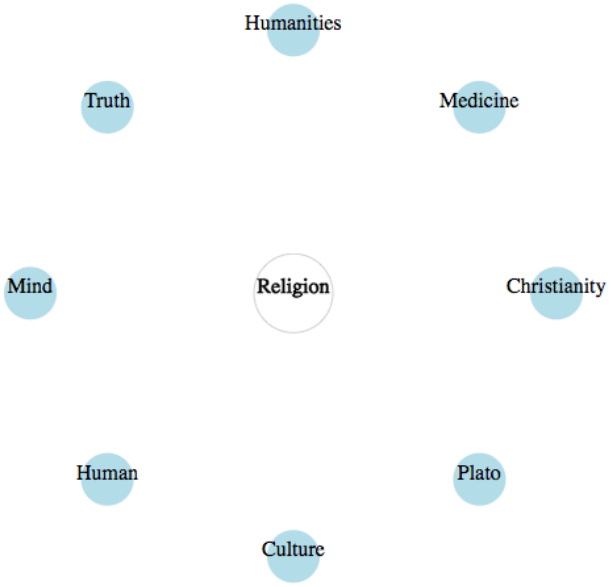
Project Assumptions

One big assumption in my approach is that the links in the first few paragraphs of a Wikipedia article are a good indicator of the linked topic's relevance. This was done to eliminate the headache of weighing different links – e.g. a link in the first few paragraphs would probably be much more important than the last link at the bottom of the page.

A better way to do this would be to collect all the links from a page, but weigh the relationships according to the position of the link in the page. Currently, all the links that are being collected from the first few introductory paragraphs in the page are given equal weightage.

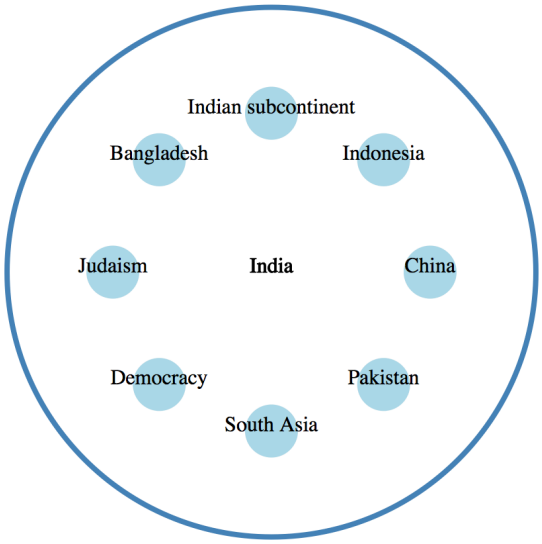
Description of Visualization

The visualization consists of two views – one that shows the most important topics that link to our target topic, and a second that shows the most important topics that our target topic links to.



This view is for the incoming links. It shows the most important topics that link to the topic of interest, in this case, "Religion." The related topics are arranged in order of their importance.

To highlight the fact that these topics are outside our topic of interest, I added an animation which shows the outer nodes to be flying in towards the inner topic.



In this view, we visualize the most important topics that our topic of interest links to.

I used the circular border to denote that these related topics are within the topic of India. Also, the animation to create this is a transition where the circle expands from the center, along with the topics. This helps the user sense that he is "zooming in" to a topic.

Description of Data

I used the graph database Neo4J to store the graph.

Nodes

Each article in Wikipedia has been represented as a node in the graph. It has 3 attributes:

- Title: String representation of the topic title
- URL: String which contains the URL at which this topic's Wikipedia article resides
- Incoming: an integer that represents the number of incoming links that each node has

Relationships

Each link in an article is denoted as a relationship "LINKS_TO" in our graph. And every time a link is processed, it increases the "incoming" count on the destination node.

Consider an example of two topics: "Berkeley" and "California". The page for Berkeley has a link to California in the first paragraph itself. This shows that California is an important topic when it comes to the topic of Berkeley.

However, Berkeley itself is not that important a topic when you want to learn about the whole of California. And this is reflected in the fact that the page of California does not have an outgoing link in the first few paragraphs. Hence, our knowledge graph reflects these differences in the relationships.

Title: Berkeley, California
URL:
https://en.wikipedia.org/wiki/Berkeley,_California
Incoming: 28

Title: California
URL:
https://en.wikipedia.org/wiki/California
Incoming: 721

Tools Used

Data extraction:	BeautifulSoup4 (A Python Web crawler)
Knowledge graph:	Neo4J (a graph database)
Visualization:	D3.js
Front end application:	Flask

Steps To Accomplish Goals

1. The first step was to create the graph in the graph database itself. To do this, I created a python program which scraped the Wikipedia page and added the new articles to the graph. The pseudocode for that is:

   For each unprocessed node in the graph, do:
   a. Fetch the Wikipedia article from the URL.
   b. Extract all the links in the text
   c. If the node corresponding to the link does not already exist,
      i. Create a new unprocessed node in the graph
   d. Create a relationship from current node being processed to the new nodes.
   e. Increment the "incoming" number in the destination node by 1.

2. The next step was create a Flask app that could translate the graph database queries into a JSON output that could be used by the D3 application. To do this, I used the library "Py2neo", which is connects to the Neo4J database using python.

   Using this, I created a Flask application that provides a RESTful API access to the graph. E.g. to fetch the most important incoming links:

   API call: /get_incoming/<topic>

   Graph DB query: match (a:Topic {title:{topic}})<-[:LINKS_TO]-(b:Topic), b.title as title, b.url as url order by b.incoming DESC limit 8"

   Similarly, to fetch the most important outgoing links:

   API call: /get_outgoing/<topic>

   match (a:Topic {title:{topic}})-[:LINKS_TO]->(b:Topic), b.title as title, b.url as url order by b.incoming DESC limit 8"

3. The final step was to create a D3 application that consumed the REST API for the graph. This application would take the JSON data from the API, and then represent the results in the visualization.

Results

1. The graph was definitely appreciated by many. This representation unearthed new information that, even though it already existed on the page, was not visible to the Wikipedia users earlier. This was a novel interface for them to discover something new about their topics.
2. The usability of the interface left a lot to be desired though.
    a. The animations were a little wonky, and did not communicate as much as was intended. The zoom-in and zoom-out transitions were not very clear to some users.
    b. The relationships between the nodes itself was not very clear. Maybe adding some directed arrows might help communicate this idea better.
    c. Search function was also an important missing feature. People would want to be able to explore a particular topic, not just one of those on the screen.
    d. People also wanted to see some snippets of the topic in question, so that they could read about it and not have to go to a different tab or browser window to find out more about the topic.
    e. People also might like to see the context of the links. This need might be solved if on hovering over the nodes, a small snippet of text is displayed in an overlay, which contains the sentence in which the link was created.
    f. The colors could also be improved upon to communicate interconnectedness between the various ideas.
    g. Colors could also be used to denote the importance levels of the related topics. Currently, even though the articles are arranged in order of importance, visually they are all equal.

Links

YouTube video: http://youtu.be/l3ZUlvlNSdQ
Code: https://bitbucket.org/samudranb/wiki-graph-viz/

Contributions

All the parts of this project (using the Graph database NEO4J, Flask application and the D3 visualization) were done by Samudra, except for a few lines of code in the file "wikiparser,py" which are related to scraping Wikipedia by using the library BeautifulSoup4. Dinu Nair, a friend who has had experience scraping websites before, submitted those lines of code.