COMPSCI 2XB3:Computer Science Practice and Experience: Binding Theory to Practice
Project Proposal

Project Title:	***Link-ipedia***
Lab Section Number:	**L04**
Student Names:	***Jessica Lim, Jay Mody, Pranay Kotian, Maanav Dalal***
Group Number:	**6**
Student McMaster Emails:	***Limj31@mcmaster.ca, modyj@mcmaster.ca, kotianp@mcmaster.ca, dalalm1@mcmaster.ca***

By virtue of submitting this document I electronically sign and date that the work being submitted is my own individual work.

Abstract

Wikipedia is the largest online encyclopedia of knowledge with over 40 million articles covering an extensive range of topics [1]. Articles are often connected to each other via hyperlinks to related topics and articles. Quantifying the various paths between articles can provide key insights into the relationships between knowledge bases.

Link-ipedia is a product designed to discover how one topic relates to another. Using the 'wiki-topcats' dataset , it provides the shortest path between over a million topics. The dataset contains files that list all Wikipedia web pages with their respective id, all web pages in a given category, and all hyperlink connections between web pages. The data will be manipulated using a variety of shortest-path (Dijkstra's algorithm), searching and sorting algorithms, to provide users many options for hyperlink paths.

1. Objective

Using the 'wiki-topcats' database containing Wikipedia articles and their network of hyperlinks, our objective is to, given one wikipedia page, find the shortest path to a second wikipedia page. We will accomplish this objective by graphing wikipedia pages as nodes and connecting them via hyperlinks as edges.

2. Motivation

While encyclopedias and books provide a plethora of information about a particular topic, there are few resources that provide information on multiple topics. We recognize that Wikipedia is one of the biggest online databases for free and accessible information. Thus, it is one of the first resources any student will use when trying to get a basic understanding of a topic. When researching, it is critical to recognize the connections and links between various ideas. We therefore want to simplify that process, so that users can better understand the many ways in which ideas, people, places etc. intertwine with one another.

Wikipedia is the largest and most general reference resource on the internet, so the sheer amount of data makes it oftentimes extremely difficult to navigate between pages [1]. When trying to understand a

connection, having a basic knowledge of the intermediates is extremely helpful. Thus, we would like to provide the names and links of pages that connect various topics, as this would make it easier for individuals to understand the topics being researched.

Wikipedia has nearly 500 million unique viewers per month, many of whom are simply using it as a resource for general information [1]. This product would be useful for all viewers looking to find connections - whether for the purpose researching a particular topic or discovering something new. We hope that the product will be able to give users a variety of options to connect multiple articles or topics. Given particular topics, users should be able to find multiple ways in which the topics connect, thus giving them a comprehensive understanding of the topics being searched, and all intermediates.

3. Prior Work

Graph Visualization of Wikipedia [2]. This website graphs all the wikipedia web pages, sorts them by category, and shows all the connections between the wikipedia pages. Our solution likely uses a different dataset (based on the year it was created) and in addition to analyzing the connections between web pages, we will be finding the shortest path between two web pages. We will be creating a visual representation of the connections like this website.

The Wiki Game [3]. The wiki game served as inspiration for our project. The user must get from a start page to an end page by clicking on hyperlinks within the current Wikipedia page they are on. Our final product will find this quickest path, essentially having the computer play the game for us.

Studying the Wikipedia Hyperlink Graph for Relatedness and Disambiguation [4]. This paper examines the relationship between Wikipedia hyperlinks and two tasks, word relatedness and named-entity disambiguation. The factors they consider include direct links, reciprocal links (pages that link back to each other), number of mutual connections, etc. In comparison to our project, this paper covers more advanced topics like machine learning that don't fit within the scope of this course. We may, however, still implement it if given the time!

COMPSCI 2XB3:Computer Science Practice and Experience: Binding Theory to Practice
Project Proposal

4. Input/output and proposed solutions

Input

Our input data is sourced from http://snap.stanford.edu/data/wiki-topcats.html, which contains three files that make up the graph dataset. Each node represents a wikipedia article entry, and each edge represents a hyperlink connection from one article to the other (directed). Here's a summary of what each of the files contains:

1. *wiki-topcats.txt (http://snap.stanford.edu/data/wiki-topcats.txt.gz): Space separated values file for all the directed edges (source_node_id and destination_node_id separated by a space on each line).*
2. *wiki-topcats-categories.txt (http://snap.stanford.edu/data/wiki-topcats-categories.txt.gz): Space separated values file that map a category name with all the nodes that are of that category.*
3. *wiki-topcats-page-names.txt (http://snap.stanford.edu/data/wiki-topcats-page-names.txt.gz): Space separated values file that maps node ids to their related wikipedia article name.*

The dataset contains a total of 1,791,489 nodes and 28,511,807 edges.

Output

The output for our application will be the shortest path between one article to another. For example to get from Canada to Lebron James it might return:

Canada -> Toronto -> Toronto Raptors -> NBA -> Los Angeles Lakers -> Lebron James

Solutions

Here, we would use a lexicographical search so that the user can search for Canada and Lebron James. Then we use some kind of graph search algorithm (say a simple DFS, BFS, or Dijkstra's shortest path algorithm) to find the shortest path from a to b.

5. Algorithmic challenges:

The main algorithmic functionality of our product is going to be finding the shortest path from one wikipedia article to another via hyperlinks (which are the edges of our graph). Of course, we want the user to be able to choose the source and destination articles, so we will also need to implement a lexicographical search algorithm for the article names. Additionally, we can also sort articles by number of connections to it, from it, categories, name, etc… For more complex algorithms, we can also implement an algorithm to find the shortest path from 'article a', to 'article c', while also hitting a user-specified 'article b' on the way. There's also potential to scrape extra information from the wikipedia articles themselves (subcategories, metadata, urls, edits, comments, authors, protection level) and use the additional data to give even more insight into the wikipedia network of articles (for example, implementing a supervised clustering algorithm for categories). One example of this would be using the article's title to scrape the corresponding URL from the internet, and appending that URL to our dataset. This would allow for users to better interact with the program when it's completed.

6. Project plan

Date range	*Milestone*	*Deliverable*	*Description*
Feb 9 - Feb 15	***Project Proposal Presentation***	*Project presentation powerpoint*	*A powerpoint of our project idea, and how we plan on implementing it. To be presented to the class.*
Feb 23 - Feb 29	***Project Scope Document***	*Project scope document*	*A document that will help us create our requirements specification. This document will narrow down our scope in terms of what we believe we can actually accomplish (after having had some time to work with the dataset).*
Mar 1 - Mar 7	***Requirements Specifications***	*Requirement specification*	*A completed requirement specification, based on what we learned about creating one in 2XB3 and our other software courses. Will include mockups of our final product.*
Mar 8 - Mar 14	***Project Progress Checkpoint***	*Working code (without any gui)*	*Over this time, we should be able to have working code that is able to correctly interpret an input and output page, and produce the given length between pages*
Mar 15 - Mar 28	***GUI Development***	*Working GUI*	*Over this week, using Swing in Java, we will enhance our code by containing it in a GUI (which will eventually compile into an application(*
Mar 29 - Apr 4	***User Testing & Refinement***	*Robust code*	*In effective software development, it's important to get user feedback and some QA completed. As a group and enlisting the help of our peers, we will test and refine our code to ensure it meets our requirement specification and works as expected.*
Apr 5 - Apr 12	***Final Project Code***	*Eclipse archive file*	*Over this last week, we will put the finishing touches on our code, archive the file, and submit it. Ideally, we can also find a way to host this code publicly so that people can use it.*

References

[1] “Wikipedia,” Wikipedia, 05-Feb-2020. [Online]. Available: https://en.wikipedia.org/wiki/Wikipedia. [Accessed: 08-Feb-2020].

[2] Nilsson, A. (n.d.). Graph Visualization. Retrieved February 7, 2020, from https://wiki-insights.epfl.ch/dynamic-graphs/

[3] The Wiki Game - Wikipedia Game - Explore Wikipedia! (n.d.). Retrieved February 7, 2020, from https://www.thewikigame.com/

[4] Agirre, E., Barrena, A., & Soroa, A. (2015). Studying the wikipedia hyperlink graph for relatedness and disambiguation. arXiv preprint arXiv:1503.01655.

[5] Wikipedia network of top categories. (n.d.). Retrieved February 7, 2020, from http://snap.stanford.edu/data/wiki-topcats.html