This document contains five sections. The first section is the Introduction, and it explains the idea and motivation for a graph based wiki. The second section is a list of features for the wiki – including the use of edge weights, graph algorithms, and filtering. The third section describes two example use cases. The fourth section describes the implementation details, and the fifth section contains a list of deliverables.

Introduction

A lot of information and knowledge can be usefully represented as graphs. Connections between topics can be modeled as edges, with the edge weights indicating some quantitative measure like distance or strength. Graph algorithms can then be run on the structured graph representation to obtain interesting insights and discover new connections.

Thus, it seems like there would be great potential for a graph based wiki. Most wikis seem to currently organize information in a linear way, but this linear presentation does not have some of the significant advantages that could be obtained using a graph based representation. A graph based wiki would show information as a graph with nodes and edges. A node would represent some item of knowledge (a technique, object, topic, etc), and clicking the node would take you to the page for that topic. The weighted edges would represent connections between topics, and clicking an edge would take you to the page that explains the connection between the topics. The graph representation would be interactive, and graph algorithms could be run to find insights
about the connections between the topics. We could also add filtering based on the edge weights to let users see what a chosen topic is closely connected to.

For example, a student or researcher might be interested in asking about the “nearest neighbors” of a technique, object, topic, or area. This could naturally be modeled using edge weights. While it could be argued that the “distance” between topics in current wikis can be measured using link distance (the number of links needed to get from one page to the other), the link distance was not specifically designed with this use in mind and can often give misleading results. In our graph based wiki approach, we would ask verified contributors to specifically add weighted edges between topics, or we would come up with another focused measure of how to compute the distance between two topics (when applicable – sometimes it would not make sense to have an edge between two topics).

Information would then be presented as a graph, which would allow a clean, visual presentation of many areas. For example, we could have a page with many species of animals and the genetic distances between them. We could then add more features, like filtering on the edge weights and directions. We could even add multiple layers of edges – for example, we could have one layer of edges that shows the genetic distances between animals and another layer of edges that shows the predator-prey relationships between animals. It would then be possible to choose which layer of edges to display – so we could cleanly shift from looking at the genetic distances edges to looking at the predator-prey relationship edges. We could even add algorithms that compute the overlap between edge layers – a possible question might be whether animals that have a small genetic distance (and are thus closely related) are less likely to be involved in a predator-prey relationship.
The graph based wiki would explicitly ask contributors to add information in a graph form. Verified contributors would have the option of adding weighted edges between different nodes, or requesting for administrators to add filtering options or new layers of edges. The wiki would be designed in a modular fashion, so that the software could be extended with new graph features and graph algorithms. The graph based display of information would also be interactive, so that a user could scroll around in the graph of nodes, zoom in on a cluster or area, and choose to hide parts of a graph. I think that a graph based wiki would allow visualization of information in a way that a linear based wiki does not. Students could easily look at the connections between areas in an interactive, visual graph format. The wiki would also allow users to discover new connections or insights between areas, by using graph algorithms on the weighted edges.

Features
The wiki would be designed as a web application. I have added some possible features below.

-Display of information as a visual graph with nodes and edges. Clicking a node would take you to the page for that topic. Clicking an edge would take you to the page that explains the connection between the two topics that are the two endpoint nodes of the edge.

-Filtering based on edge weights. This helps to see what a topic is close to or far from (where the edge weight would indicate some measure of distance). The edge weight’s meaning would depend on the edge layer – we might add an edge layer for genetic distances, and another edge layer for predator-prey relationships.
-Edge layers would let us use multiple sets of weighted edges for the topics in an area. We might want one layer of edges that shows the genetic distances between a set of animals, and another layer of edges that shows the predator-prey relationships between a set of animals. We could then run algorithms that compare the edge layers, to see if there’s any overlap between genetic distances and predator-prey relationships.

-Users can add pages of linear text (like on Wikipedia) or they can add graphs. To add a graph, they would click the option “Add graph”, specify the nodes and weighted edges, specify some details about the graph (the units of the weights, what the edges represent, etc) and optionally add a link for each node or edge. The wiki then displays the resulting graph with interactive features as described. Users can add different edge layers as described above.

-Interactivity for the graph. The visual graph presentation of information would be interactive, which means that users could move around in a large graph of information using Javascript. Users could also zoom in on one area of the graph, or hide parts of the graph. For example, a user might click a node and choose to hide all nodes within a chosen distance of that node.

-Ability to export the data sets specifying the graphs in various formats, so that people can run their own graph algorithms on the data sets or use the data sets in other ways.

**Example Use Cases**

This section contains two example use cases for a graph based wiki.
Use Case 1 – Looking for a correlation between genetic distances and the existence and strength of predator-prey relationships

I have touched upon this possible use case throughout my proposal, but I will include a more explicit description here. One possible use case is that a team of biologists (or an individual biologist) is interested in whether there is any correlation between the genetic distance between two animals and the existence and strength of a predator-prey relationship between those same two animals. For example, we might predict that animals that have a smaller genetic distance (and are thus more closely related) are less likely to be involved in a predator-prey relationship. Since the wiki encourages entering information in a graph format with weighted edges, it is very possible that a graph of some animals with weighted edges representing genetic distances exists. There might also be a graph of the same animals with directed, weighted edges representing the predator-prey relationships between the animals, with the weight of the edge in this case being a number \( x \) between 0 and 1 that indicates what proportion of the predator’s diet the prey constitutes (where the predator and prey are the two nodes of the directed, weighted edge).

There would then be a built in algorithm to find correlations between the two sets of edges (that is, in this case it would check whether the genetic distance between two animals is correlated to the strength of the predator-prey relationship between the same two animals). This would be a built in algorithm, since it seems like it could be a common use case. However, the data that specifies the graphs would also be available in JSON and other easily parsed formats, so that researchers could easily download the data specifying the graphs and run graph algorithms themselves.

This use case shows some advantages of a graph based wiki over a linear based wiki. The graph based wiki encourages entering information in graph format, which means that data
specifying graphs will be easily available to parse and run graph algorithms on. Also, it will be possible to visualize graphs to try and get an intuitive look at possible connections. In contrast, a linear based wiki like Wikipedia does not provide data that specifies graphs. If someone wanted to get a graph of genetic distances between a set of animals from Wikipedia, they would have to read through the articles and look for the facts by hand. A graph based wiki would encourage the creation of clean data sets that specify graphs, and users and researchers would be able to run built in algorithms on these data sets (or download the data sets in common formats and run their own graph algorithms).

Use Case 2 – A student looking for relationships between areas of study

This is more of a “visualization” case. Apart from being able to present graphs specified by data sets, the wiki should also be able to automatically generate graphs. For example, a student studying calculus might want to know about where calculus is used in machine learning. The wiki should be able to generate a graph to show some of the important connections between calculus and machine learning.

I will describe one possible way that this might be done. We’ll assume that registered students have indicated their academic level – for example, undergraduate or graduate. Then suppose a student John who is an undergraduate requests a graph that shows some of the important connections between calculus and machine learning (that an average undergraduate might be expected to come across).

The wiki will create this graph using counts of how many times hyperlinks have been clicked by other undergraduate students. We assume that we want to show the kinds of connections between calculus and machine learning that an average undergraduate student might be expected
to come across. Some of the linear text pages in the wiki will be tagged with machine learning (for example, the linear regression or neural networks page), and some of the linear text pages in the wiki will be tagged with calculus (for example, the page on integrals and the page on derivatives). Then the wiki will maintain a list of how many times a hyperlink has been clicked by other undergraduate students in the past. Then it will take the top hyperlinks where the two endpoints were tagged with (machine learning, calculus) or tagged with (calculus, machine learning). It will then create a graph that shows these nodes with the corresponding edges.

For example, it might be the case that many undergraduate student accounts in the past three months clicked the link on the “neural networks” page that took them to the “derivatives” page. Then when the undergraduate student John requests a graph of the connections between calculus and machine learning, the wiki will generate a graph that contains the nodes “neural networks” and “derivatives” with an edge connecting them – along with other pairs of connected nodes. John might then click on the neural networks page in the wiki (that is, the neural networks page that contains linear text) and find out that derivatives are used in neural networks as a part of gradient descent.

If John wants to contribute, he could save the graph as a featured graph in the wiki and then click on the edge between “neural networks” and “derivatives” to create a new page that explains some of the connections between the two. He could then add that derivatives are used in neural networks as a part of gradient descent. A later user could look at the same graph and click the edge between “neural networks” and “derivatives” to go to the page created by John that explains some of the connections between neural networks and derivatives.

John could tell the wiki to limit the graph to the fifty most highly clicked links that went from a page tagged with “machine learning” to a page tagged with “calculus” (or the other way
around). This use case would help people to visualize connections between techniques, objects, or areas. Users would be able to look at a clean graph representation, and then click on the nodes or edges for more details.

In the above example, if John was a graduate student, the wiki might return a very different graph. The counts for the hyperlinks would be taken from graduate student accounts, and we might expect that graduate students don’t usually click on the link on the neural networks page that goes to the derivatives page, since graduate students in machine learning would already know about how derivatives are used as a part of gradient descent. Instead, the most highly clicked links from pages tagged with “machine learning” to pages tagged with “calculus” – where the counts only include clicks by graduate students – might be more advanced connections that are related to research problems in the area. By generating a graph targeted toward the academic level of the user, the wiki can show the kinds of connections that the user is interested in.

**Implementation Details**

I am planning to write this as a web application. I will probably use Python and the Flask framework for the server side. The front end will be Javascript heavy, and I am planning to use a Javascript library to implement the interactive parts. All the page presentation and appearance will be done with HTML and CSS, as it’s intended to be a web application. Depending on what part of the project I choose to focus on, I might extend existing wiki software or use existing libraries for some of the built in graph algorithms.
Deliverables

The deliverables will be:

1. The code for the application (with the understanding that I am planning to use libraries to implement some of the functionality).

2. A 10 - 15 page write up that includes a description of the project, a description of the built in graph algorithms, an overview of the code architecture and implementation, and a short guide for users and administrators on how the wiki should be used and extended.