FINAL PROJECT:

WORD GALAXY

Lucas Silva & Connor Ameres
Motivation and Audience
The motivation for our project stemmed from the potential to deliver an application that would visualize relationships between words in an intuitive manner. In our research, we found that the visualizations we were envisioning for our project were not present in many of the popular dictionaries and thesauri online. Thus began our research both in the ways of the relationships between words and the ways in which we could represent them.

After reviewing previous course material, we found the development of force directed graphs of interest. Graphs offer us the most intuitive visualization for conveying the relationships between words. Also, we were intrigued by the ways in which we might be able to apply various clustering algorithms to group words. We were also interested in the whether these groupings would reveal any information about the words graphed.

Lastly, we found that we wanted to create a project that would utilize a large amount of data. In our previous class work, we had worked with rather small datasets. For our final project, we wanted to work with a data set that would push the limits of a potential online application accessible from nearly anyone, regardless of their platform of choice. Graphing all words is simply no trivial task, as there are more than 1,000,000 words in the English dictionary and we found that we would later find that words were highly interconnected. This drew us to the creation of this final project, because we had little prior experience in working with graphs of this size and presenting them in a web application.

Our target audience of our application is anyone that is a visual learner. Naturally this audience includes anyone who is interested in learning words and the relationships that they have. This would specifically benefit those in the field of academia. This would likely be students in high school or above, as the application set out to contain all words, including those that are explicit. We also anticipate that users will be able to use the application with no experience with graph theory.

Visualization Goals
Our application must be interactive, web-based, and intuitive. It must fulfill these criteria because it must be easily accessible to everyone on any platform regardless of their computer hardware. The application must be intuitive, because non experienced users must be able to use it. It must be interactive to enable an exploratory mode of learning and grab the attention of the users.

We must also provide a way in which to search for a particular word. The graph structure is inheritably dense and large. It would be very difficult to find a particular node or word. In order to be competitive with alternative online dictionaries, we must provide a search mechanism in order to move to a specific node in the space or to alert the user that the word is not in the graph.
Lastly the application must be fast. The application should load quickly, even on slower networks, as a delay would likely result on frustration or a user bouncing from our application. The application also must have high frame rates, around 60 frames per second, as lower frame rates will result in users having a difficult time navigating the graph and tracing the paths through the exploratory space.

Research Question

We set out to answer whether or not we could create a visualization that is more intuitive than many of the available online dictionaries for understanding the relationships between words.

Expected Results & Hypothesis

Using a graph generated of hyponym trees from Princeton’s WordNet API [4], we hypothesize that our site will have greater average session times when compared to more traditional dictionary sites, as it is designed to be more immersive. We expect that given a survey, the majority of users will find our project more exploratory and visually appealing than traditional sites.

Prior Work/Research Papers

The first draft of this project attempted to use the ForceAtlas2 algorithm designed for Gephi by Jacomy et al. [1], as it is known for displaying communities very well. We used this algorithm for our prototyping using Gephi, and the paper helped us determine the appropriate constants for an optimal layout. However, we could not find a 3D implementation of the algorithm that fit our needs so we decided to abandon it and use a more generic force directed layout implemented by Kashcha [2]. In the future, we may move to ForceAtlas2 by implementing a JavaScript version.

From the beginning of our design, we focused on the concept of overview and detail described by Jakobsen et al. [3]. We wanted users to be able to explore the relations of all words in an overview context, while also being able to specify a particular word and gain more detail about it. The ability to specify a word is a crucial feature in order to make our project compete with traditional dictionaries and thesauruses.

In order to gather the data we needed, we used Princeton’s WordNet [4] which organizes words based on synsets. It then provides semantic relationships between synsets, such as hyponyms, troponyms, meronyms, etc. After reading this paper, we decided hypernyms and hyponyms best represent the relations between words as synonyms can then be defined by words belonging to synsets that share the same hypernym. Using only these two relations, we believe the user would be able to gather enough semantic relationships between synsets to make our project useful.

At first we attempted to build our project from scratch, and our prototype was built from the ground up using Meteor.js. However, after demo day and many issues with
dependencies, we decided to built upon an existing project [5] found on Github that had many of the features we were looking for. Since the project is under an open source MIT license, we decided to fork that project and built on top of it. This project was originally used to visualize package managers, like npm. This gave us a great starting point to render the exploratory 3D graph, after which we added our own 2D view and other modifications.

Visualization Design Evolution

Storyboarding

Initially we sketched out a 2D, force directed graph in a storyboarding tool called Sketch. This tool allowed us to create a drawing of what our 2D view of the final design would end up looking like. Figure 1 shows what we believed the initial graph would look like upon loading the application. The sketch showed that we would support exploration of the graph through zooming.

Figure 1 also shows what the visualization would look like after searching for a particular word. When searching for a particular word, the visualization would remove nodes that aren’t directly connected to the word of interest. The visualization would also move the searched node to the center of the visible screen.
We construct our graph with a python script that gathers word relations from WordNet, but before attempting to render it in the browser we wanted a preview of what the result would look like. For this purpose, we used Gephi, a popular open source graph visualization software. Gephi contains many different force layout algorithms built-in, but for our prototyping we used the ForceAtlas2 algorithm.

This was helpful because when using words instead of synsets for nodes, we immediately noticed the very high average valence and interconnectivity of the nodes. This was a problem because nodes were being obscured by edges, and the graph was very cluttered, see Figure 2 for obscured graph with 1,000 words. Knowing this before actually doing the rendering work was very helpful, as we were able to switch to visualizing hyponym trees of synsets instead of words. This cut down our average valence, unobscured nodes, and made the graph more aesthetically pleasing while still encoding enough semantic relationships to make it meaningful.

Figure 2: Left, zoomed out view of Gephi graph of Google’s top 1,000 words. Right, zoomed in view of same graph.
At demo day, we had a working prototype that we built from scratch using open source ngraph modules. The prototype was able to render a 3D graph using a force directed graph layout, but quickly became very cluttered when edges were visualized. Hiding node edges made the graph less cluttered at the expense of removing the semantic relations between words (see Figure 3). We were able to successfully store node positions and edge information in binary files, which were much smaller than their ascii counterparts. This aided in improving load time, but the rendering time for our graph was very large. The prototype would take around 10 seconds to render, during which just a black screen was displayed to the user.

We also had a working search functionality at demo day, where the user would enter a search term and the application would render a 2D graph. The 3D graph would be minimized to the bottom right, and focus would be switched to the 2D version. The root of the 2D graph
would be the search term, and all words connected to that term would branch off from the root. Node labels were not yet implemented, and the force layouting algorithm would continuously run in an infinite loop which made the graph move and rearrange itself. See Figure 4 for this search functionality.

Final Design

During the final iteration of our design, we pivoted to utilize a different visualization project. This project was pm [5]. Firstly, we adapted the word graph that we had previously generated to work with the existing visualization. We added word definitions, changed the relations to be hyponyms and hypernyms, and re-implemented search to the original project. Our new search functionality actually searches through a synset’s lemmas, which are its synonyms, instead of the synset itself. This is better because users can search for more natural terms, and we get higher quality matches. For example, ‘hi’ would now match to the synset ‘hello’, as before it would not. We also added color to nodes based on part of speech, selected from a qualitative color scheme taken from ColorBrewer 2.0. With the addition of color, we also added a legend to the bottom right of the scene, as can be seen in Figure 7.

In addressing feedback regarding the difficulty to navigate the application, we added a navigation control overlay which can be toggled using the scroll wheel or laptop trackpad. The overlay displays all the pertinent navigational controls, many of which were taken from the original pm project, seen in Figure 6.

Lastly, in order to provide the ability to focus on a word and its direct relationships we created a 2D detailed view which is enabled by double clicking on a word. The 3D graph will smoothly transition to the top right of the scene, and a 2D force-directed graph will be
rendered in the center of the scene. The root node of the new graph becomes the selected word, and its direct relationships are connected. The user can then toggle whether to display hypernyms or hyponyms by either clicking the root or the info box in the bottom left corner. Clicking on a leaf node makes that node the new root, and changes the graph accordingly. This view can be seen in Figure 8.
Future Work

In a future implementation we would like to be able to filter synsets based on the age group of the user. We would have a user enter their date of birth in a dialog box during the introduction. This detail would be stored with LocalStorage in the user’s browser.

We also would compress the data files that are served from the server. If the files are compressed, we would achieve approximately 75% compression on the labels file that contains all node information. This compression would result in lower latency for navigation and coloring of nodes.

Improving the search performance is desired, because the application can perform slowly when shorter words are searched. This is because when a user enters a search term, we find all nodes whose lemma (synonyms) lengths are longer than the entered term. Then, we look through all lemmas and try to find ones that match the search term. A match is considered valid if the lemma starts with or ends with the entered term, which is why we first pruned shorter lemmas as they cannot match. This means that for shorter search terms, the search space is larger because less lemmas are pruned. They also match more nodes, because more nodes are likely to start with the term.

Feedback on Project

Not Implemented

Our project received great feedback from the class during demo day. However, we weren’t able to implement everything that our classmates had suggested due to time constraints. One of the first great ideas that was suggested to us was to “create a 2D rendering of a mini-map for location in 3D graph”. We really liked this idea, because we would have been able to reuse UI elements from our application to render the mini-map, while continuing the trend of providing a detail and overview of the exploration of the graph of words.

Another great idea was to “color words based on their country or language of origin”. This idea would have been incorporated if we had the time to research a data set to pull the country or language of origin from. Coloring the nodes would have been trivial.

Lastly we had difficulty with our use of clustering algorithms, so we were unable to implement the suggestion to “color nodes by the clustering of related words”. Without the use or development of a clustering algorithm, the application wouldn’t be able to color nodes by their clustering in a way in which the user would be able to intuitively explore.

Implemented

We were able to implement nearly all suggestions that didn’t require another data source or the addition of a clustering algorithm. Many students claimed that we should “display a tutorial for users to become familiar with the application”. We attributed these claims to students having a vague idea of what our application was used for once we told them.
However, they didn’t feel comfortable sitting down without a background lesson and or additional help.

Students also claimed that “the velocity of the camera was difficult to control”. While our nodes were still unlabeled during the demo day, users had a difficult time returning to specific nodes in the graph and performing tasks. This was primarily due to navigational issues with the camera’s velocity.

Lastly, students claimed that we should “use autocomplete to suggest words to the user during search”. During this stage of development, the user received no feedback when typing a word into search. The user could only interact with the application after they had typed in a word and pressed the enter key. The user also wasn’t presented with any feedback when a word was not present in the graph, so users thought autocomplete would work to combat these other issues.

Contributions
The various work was primarily divided into client facing work and work dedicated to initially generating the graph. Connor worked on the storyboarding with Sketch to initially design a UI for the application. After the storyboard was completed, he worked to generate the graph and word information using NetworkX and WordNet in python so that it could be later laid out in JavaScript. Lastly, Connor worked on implementing the 2D force directed graph into the visualization. This was created using D3.

Lucas worked on the prototyping using Gephi to test graph layouts, and implemented the Meteor prototype that was shown during demo day. He worked with the layouting algorithm in JavaScript. This involved tuning the gravity and spring constants to get an aesthetically pleasing graph. Lucas also worked on modifying the pm project to generate the 3D graph using our generated layout, and re-implemented search in order to search by lemmas instead of synsets.

Technical Implementation Details/Challenges
The difficulty that we haven’t yet covered is the lack of 3D graph tools. We specifically had difficulties finding a forceAtlas2 algorithm that would work for our platform. We were also unable to find an algorithm that would provide comparable clustering.

Future Work
Both members will likely continue to work on the project, due to great feedback we received during our final presentation. We were notified on social media that users were interested in using our visualization long term. We will be working on the tasks that we were not able to implement from the demo day feedback.
Project Links:
Video Demo: https://www.youtube.com/watch?v=56is1Xv97y4

References


