ECS272 Final Project: Pokedex-KG

QIFAN REN, University of California, Davis, USA
WENCHANG LIU, University of California, Davis, USA

In our final project, we are aiming to build a system to explore interesting stories of Pokemon games with the help of the Veekun open-sourced Pokemon dataset[2]. We introduce a node-link diagram based knowledge graph visualization system called Pokedex-KG to find stories by extracting 5 key aspects of the Pokemon game as nodes: Pokemon, moves, types, abilities, and egg groups, and 5 different relationships between them as edges: Pokemon can master moves, Pokemon has types, Moves has types, Pokemon can have abilities, Pokemon belong to egg groups. The system follows a context-focus design where users can select a node to check its details, and we also provides some advanced features such as search for a node or customize the layout and node size encoding. Also, our work includes a storytelling mode where we will provide some example stories and a exploration mode where users can use it to explore Pokemon stories themselves. Our source code are available at https://github.com/williamlwclwc/ECS272-PokedexKG, and our demo is deployed at https://ecs272-final-pokedex-kg.vercel.app

Authors' addresses: Qifan Ren, qfren@ucdavis.edu, University of California, Davis, Davis, California, USA; Wenchang Liu, wchliu@ucdavis.edu, University of California, Davis, Davis, California, USA.

1 MOTIVATIONS

In our final project, we are aiming to explore interesting stories of Pokemon games. The Pokemon game looks simple, all ages can play, but it is actually not, there are hundreds of different Pokemon and moves, 18 different types, not to mention the stats and abilities a Pokemon process. Although there are existed analysis about Pokemon which can answer some easy questions such as the portion of different types of Pokemon, average stats of a certain type, etc. However, these are not helpful enough to answer many questions concerned specific Pokemon or its attributes, for example, a player may want to have a Pokemon into his team that is fire type, has the solar power ability, and also has great attack stat. None of the existed data analysis can help, the player needs to iterate through the Pokedex database to find out, which is tedious can inconvenient.

We are expecting to find stories with a Pokemon knowledge graph visualization system. We mapped 5 key kinds of instances of the Pokemon game as nodes: Pokemon, moves, types, abilities, and egg groups, and 5 different relationships between them as edges: Pokemon can master moves, Pokemon has types, Moves has types, Pokemon can have abilities, Pokemon belong to egg groups. We want to make our system informative, providing both general overview of all data and detailed information of a specific instance, as well as explorable, providing an interactive way to explore the Pokedex data, which allows users to explore more stories on their own besides our storytelling examples.

2 DRIVING APPLICATION AND DATASET

One of the key driving application is Pokedex. In each Pokemon game, there will be a task for the player to collect Pokedex by catching Pokemon. A Pokedex is a database that records all Pokemon related information, there are also a lot of third-party Pokedex application such as BulbaGarden [1] that players can either iteration through the list of all Pokemon or search for a specific instance. Our system has the similar functionality of a pokedex but in a more interactive and visualized way.

The dataset we used for this project is a subset of the Veekun[2] open-sourced Pokemon dataset from GitHub. We considered Pokemon(and its stats), moves, type, ability, egg group instances data, and Pokemon-Move, Pokemon-Type, Move-Type, Pokemon-Ability, Pokemon-EggGroup relations data from generation 1 to 8 Pokemon games. For simplicity, we filtered out different Pokemon forms and some special moves that not belongs to a specific Pokemon like "max moves" from generation 8 and "Z moves" from generation 7.

We use Python to preprocess the csv datasets, and React, Ant Design, D3 to implement the front-end application.

3 CHALLENGES

The challenges we met including the following and we will explain our solutions in later sections:

(1) Interaction design of the node-link diagram
(2) How to make the node-link diagram advanced
(3) How to avoid the node-link diagram to be chaotic
(4) How to render a large node-link diagram smoothly
(5) What kinds of stories to expect

[1] https://bulbapedia.bulbagarden.net/wiki/List_of_PokÃĭmon_by_National_PokÃĭdex_number
[2] https://github.com/veekun/pokedex

4 RELATED WORKS

In knowledge representation and reasoning, knowledge graph[2] is a knowledge base that uses a graph-structured data model or topology to integrate data. Knowledge graphs are often used to store interlinked descriptions of entities – objects, events, situations or abstract concepts – while also encoding the semantics underlying the used terminology.

Our work can be considered as a visualization of simplified knowledge graphs, where we only considered that there is a relation between two instances but not the semantics or advanced knowledge of the edges. Our idea is inspired by several successful applications of knowledge graph visualization such as connected papers[3], a node-link diagram based visual tool to help researchers find and explore papers relevant to their field of work based on the citation relations between papers. We think that Pokemon games has more kinds of relations that may leads to more interesting stories when we visualize them.

We choose to use HTML5 Canvas instead of SVG and WebGL because we saw this large graph demo[4] that can render 75k elements using HTML5 Canvas smoothly, and we referenced an Obverable example[5] to implement our own Canvas based d3 force directed graph.

Our interaction design of expanding nodes is inspired from the friendship and aggression network visualization[4], and the context-focus interface design is similar to our previous homework.

5 DESIGN AND METHODOLOGY

5.1 Visualization Design

To explore the relationships with different aspects mentioned above, we think that a node-link diagram is a great option to achieve this. A node can represent a Pokemon, a type, a move, an ability, or an egg group. The links are undirected and represent the relation between two nodes, i.e. a Pokemon can have "can master" relation with a move, a Pokemon or a move can "has" a type, a move "has" a type, a Pokemon can "have" an ability, a Pokemon "belongs to" an egg group. In this way, we can connect all 5 selected elements in Pokemon games together and have an undirected graph.

Regarding the layout, we used the force directed layout[3] and we are expecting to see that some of the nodes can be clustered together with these relation forces. With force directed layout, a graph is treated as a system of interacting physical objects. Edges are mechanical springs and nodes are electrically charged particles, and nodes are pulled together or pushed apart by the forces. The relaxed (energy-minimal) states of the system corresponds to desired layouts, reaching an equilibrium in the end.

While the node-link diagram is our core visualization, we also want to display some supplemental information because users may want to know what exactly a node in the diagram is. To achieve that, we design the interface following the context-focus design. We are putting some basic information regarding that selected node, including a radar chart for Pokemon, and a link to a pokedex if the users need more information. We have a small menu side bar at left, a main large canvas for our node-link diagram in the middle, and a side bar at the right hand side to show the details for selected node. To make full use of the screen, we put legends, customization, search bar all into the menu on the left side bar. The final effect of our design is as Figure 1.

[3]https://www.connectedpapers.com/

[4]https://vasturiano.github.io/force-graph/example/large-graph/

[5]https://observablehq.com/@fbunt/d3-force-directed-graph-on-canvas-with-drag-pan-and-zoom

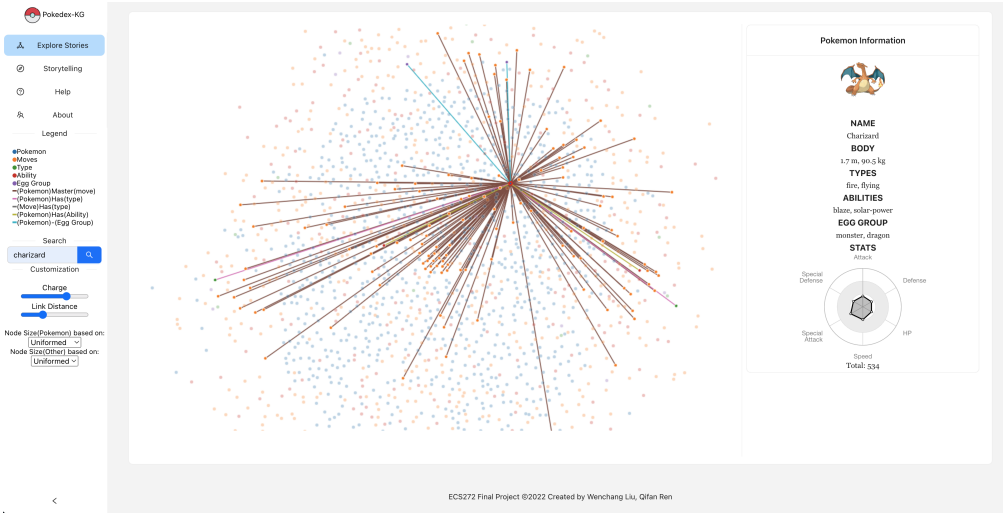


Fig. 1. A screenshot of Visualization Design

5.2 Interaction Design

Since we have 1898 nodes and 70072 links in total, interaction design is important for users to explore such large amount of data. First, we introduces pan and zoom for the node-link diagram so that users can either zoom out to see the general overview of all data or zoom in and pan to focus on a small area. Also, as introduced in the previous section, the users may want to know about a specific node, so when a node is clicked, we will show the detailed information of that node in our right side bar.

Another key design is node expansion, which is inspired from the friendship and aggression network visualization[4]. Traditionally, node-link diagram will show all nodes and links, but in our case, it looks like almost every node will have some connections with others, so it could be really chaotic to display them all, not to mention the pressure it brings to the rendering process. Therefore, we hide all links at first, and when a user double clicks on an unexpanded node to explore the relations, we will expand that node, display the links connected with that node, make nodes other than itself and its neighbors transparent. And another double click on an already expanded node will recover it. In this way, we not only preserve the relative positions of each node from the force directed layout, but also making it less chaotic.

A search feature is also necessary since it would be difficult for a user to find a specific node without it. In Addition, to make our visualization more advanced, we provide some customization options for users to change the node size encoding and the force directed layout settings. We have sliders to change the charge force and link distance of d3 force simulation. We also have two drop down selection for users to change the node size encoding of Pokemon nodes or other nodes.

5.3 Storytelling Design

The goals of our project are to guide viewers diving into the world of Pokemon games and spark the interest of viewers who know little about Pokemon. To achieve these goals, the structure of our project are supposed to consider both narrative and discovery, aka balance between author-driven and reader-driven stories[5].

Our storytelling will follow a two-stage structure, the first stage is a brief introduction into Pokemon series as well as several example stories we find with our system, heavily author-driven, then the second part leaves viewers to freely explore the detailed stories of all Pokemon, basically reader-driven. This structure perfectly fits Martini Glass.

5.3.1 Martini Glass Structure. The Martini Glass has two parts. In the first part, the visualization is introduced through the use of text, annotations, animations or views, serving the single-path author-driven narrative. The second part is consist of single or multiple interactive components, the user can actively explore the visualization following whichever path he/she considers most interesting[1].

5.3.2 Author-Driven Narrative. On the first stage of our project, we are going to reveal the stories through a slide. Our narrative will follow a linear structure, and every point will be distributed with one page of slide. The slide can be controlled by a progress bar. We will mainly use text to tell our story, supplemented by a knowledge graph with annotations.

5.3.3 Reader-Driven Discovery. On the second stage, there will be lesser text and the viewers are put in charge. We will leave the knowledge graph for user to freely explore, in which every node is either a Pokemon or a property of Pokemon. The knowledge graph is interactive, selecting the nodes will show the detailed story of such property, and the clustering effect of the force-directed layout will tell us more interesting stories.

6 IMPLEMENTATION

6.1 Rendering a Large Node-Link Diagram

We are using d3 for visualizations, React and Ant Design for our interface. We have 1898 nodes and 70072 links to render for our node-link diagram, which is a relatively large number. At the beginning, we used SVG to draw the diagram similar to most of the d3 tutorials online, but it turns out to have very low frame rates for force directed simulation, pan and zoom animations. The reason is that every shape is an DOM element in SVG, so it can get slow when there are a large scale of shapes and animations.

To solve the rendering problem, we switched to HTML5 Canvas, which is a single element and we can use scripts to draw shapes on it. To use Canvas, we call an update function to redraw every nodes and links at every time the diagram changes(i.e. force simulation updates, expand a node, drag events, node size changes, etc.). It is fast but hard to interact with. For example, previously using SVG, we can easily add an on click event on each node DOM element, but in Canvas, we can only have the on click event on Canvas, and we need to check whether the click coordinates is within any node circle to know if we click on any node. The node expansion design is also beneficial for rendering besides making the diagram looks less chaotic since we have a array to track expanded nodes and only draw the small subset of edges that are related to them.

A better option is using GPU based rendering which is even faster, however, it requires extra works regarding WebGL or its related frameworks and it looks like HTML5 Canvas is enough for our scale of nodes and links.

6.2 Interactions Implementation

6.2.1 Pan and Zoom. We call d3 zoom for canvas to achieve pan and zoom for the diagram, when zoom event triggers, we will also call the update function to redraw the diagram.

6.2.2 Select a Node. A user can click on a node to select it and we will display detailed information on the right side bar, as introduced previously, we cannot append on click to nodes using Canvas, we can only append on click on Canvas. So we compare the click coordinate and the center coordinate of each node, if the user click within the circle range of any node, we will know that the user click on that node. We implemented the click callback within the d3 drag, so we do not have to append the listener ourselves.

6.2.3 Expand a Node. A user can double click a node to expand or recover it. The way to know which node a user has clicked is the same as node selection. We add a timer logic to distinguish single click and double click, essentially, if a user click the second time within 250ms we will set timer to true and consider it as a double click. To expand a node, we keep track of all expanded nodes in an array. When double click is detected, we will either add or remove the clicked node from this array. During the update function, we will iterate through it, drawing edges that have source or target nodes in it. We will also make nodes that are not source or target of drawn edges to be transparent.

6.2.4 Search a Node. We use d3 to select the search button and create a callback function when the button is clicked. The callback function will use d3 select to get the text in the search input bar, iterate through all nodes, adding the matched node indexes into the expanded nodes array, and display the detailed information.

6.2.5 Customization. Similar as search feature, we create a callback function when the slider input value is changed. The callback function will use d3 select to get the current value of the slider, changing the d3 force parameters, and restart the force simulation.

6.3 Storytelling

We apply slides to show the interesting and inspiring stories of Pokemon. The progress can be controlled by a pagination component. The slides are separated into two parts, the first part is a brief introduction for newbies, including the basic information about Pokemon games. There is an option to skip the first part. The second part are consist of 4 distinctive stories derived from the node link. They will be discussed in the next section as case study.

7 CASE STUDIES

After we finished our implementation, we evaluated it ourselves and found some interesting stories or usages. The following subsections are our example stories.

7.1 A quick filter for Pokemon that fits into a team

A quick filter rather than iterate through a Pokedex database can benefits for advanced Pokemon players who want to find a Pokemon to his team. For example in Figure 2, solar power is a useful ability for attackers under sunny weather, it will increase a Pokemon's special attack during harsh sunlight. A player may want to look for an attacker with this ability to collaborate with another Pokemon who can change the weather and brings sunlight. Traditionally, the player needs to search for this ability in Pokedex and iterate through every Pokemon that has the ability in the list. Here, when changing the node encoding to attack stats($max(attack, special\_attack) + speed$), the user can easily filter out several Pokemon candidates that have better attack stats than others because they have bigger node size. This can be more helpful when we have a long list of candidates.

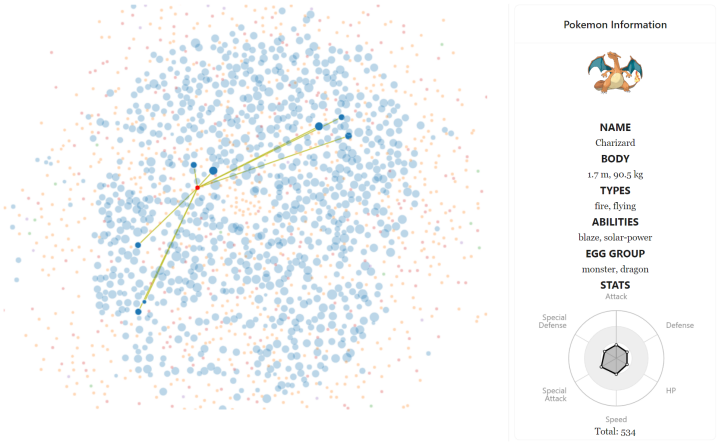
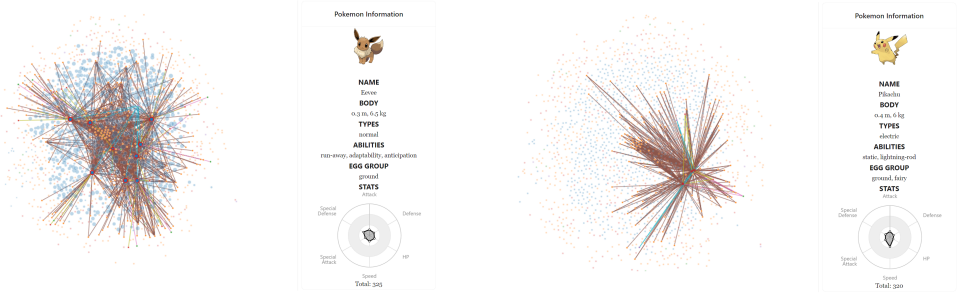


Fig. 2. The nodes link to ability Solar Power.

7.2 The different evolution pattern of Pokemon

When going through the evolution, most Pokémons retain their original types and moves, so their nodes in the Nodelink graph are near. For example in Figure 3, we can see the links of Pichu, Pikachu and Raichu shares most of moves and types. However, some Pokémons don't follow this rule. For example, Eevee is a Normal type Pokémon, but it can evolve into one of eight different Pokémons, none of them retain Normal type. We can see the nodes of Eevee and Eevee family are relatively far from each other.



(a) Eevee and its family.

(b) Pikachu and its family.

Fig. 3. Difference evolution pattern between Pikachu and Eevee.

7.3 Dragon or not?

Charizard is a draconic, bipedal Pokémon. It looks like a dragon and hatched from dragon eggs, so is it a dragon type Pokemon? However, it isn't a Dragon type Pokémon. But even it isn't offcially a dragon, it still has a close relationship with dragon type Pokemons. For example in Figure 5, through the node link we can affirm that Charizard can master the move "Dragon Tail", which is considered to be the distinctive move of Dragon type Pokemons.

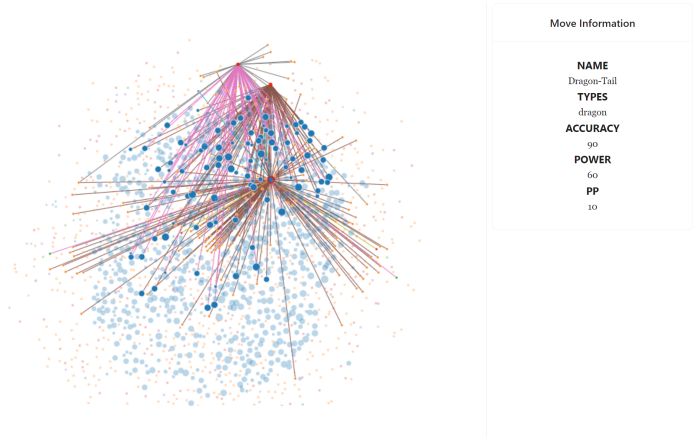


Fig. 4. Charizard and dragon type.

7.4 Relationship between types

The design of Pokemon types are based on natural matter, such as water, ice, rock, etc., or real-life creature properties, such as grass, poison, flying. Are the relationship between these types also follow the physics rules? Through our node link graph in Figure 5, we can see that some types that supposed to be more close to each other truly have shorter distances between each other, just like our real world.

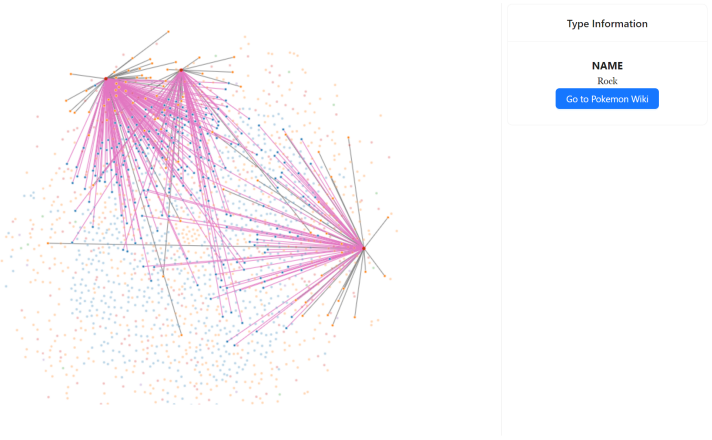


Fig. 5. Distance between types.

8 FUTURE WORKS

We developed this decent prototype within limited time, however, it is not that comprehensive at the moment. Some of the directions of future works can be as follows:

(1) Taking more data into account: For example, adding different forms of Pokemon, regions that a Pokemon appears, etc.

(2) Compare multiple selected Pokemon: We are showing information of one selected node under our current interface design, it might be helpful for users to compare Pokemon side by side with multi-selection.
(3) More interactions for Canvas based diagram: It is not easy to add interactions on Canvas base node-link diagram, the click events are basically based on the d3 drag events. However, if we can add more features like showing the name of expanded nodes without clicking, it might be better than just display a bunch of colorful nodes and links.
(4) More customization options: We provide several customization to demonstrate the ability to customized the encoding and layouts, we can further provide more of them, e.g. customization of color encoding, more options for d3 force, more options for node sizes, etc.
(5) More helpful statistics in detailed view: Currently our detailed view only display the basic information of a selected node, but ideally, it can be helpful if we can put some statistics such as the rank of the Pokemon's stats among all Pokemon.
(6) Transfer this approach to other data: Another future work can be experimenting whether this visualization approach can benefit other data besides Pokemon games.

9 CONCLUSION

In conclusion, we have introduced an interactive knowledge graph Pokedex visualization system. Our node-link diagram has a relatively large number of nodes and links, switching to Canvas instead of SVG and the node expansion interaction design make the rendering smoother. Also, we introduced color encoding based on the category of nodes and links. Apart from that, our system have selected node details display, searching feature, and customization options for layout and node size encoding. In terms of storytelling, our work follows Martini Glass structure, providing both example storytelling and a explorable way for users to explore the data. We hope our system and derived stories can benefit Pokemon players.

ACKNOWLEDGMENTS

All members contributed to this project equally. This project is supervised by Professor Kwan-Liu Ma, and teaching assistant Keshav Dasu.

REFERENCES

[1] Jeremy Boy, Jean-Daniel Fekete, and Françoise Détienne. 2015. Storytelling in Information Visualizations: Does it Engage Users to Explore Data? (04 2015).
[2] Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a Definition of Knowledge Graphs. In *International Conference on Semantic Systems*.
[3] Stephen G. Kobourov. 2012. Spring Embedders and Force Directed Graph Drawing Algorithms. *ArXiv* abs/1201.3011 (2012).
[4] Kwan-Liu Ma Oh-Hyun Kwon, Jia-Kai Chou. 2011. friendship and aggression network analysis. https://kwon.io/bullying/
[5] Edward Segel and Jeffrey Heer. 2010. Narrative Visualization: Telling Stories with Data. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1139–1148. https://doi.org/10.1109/TVCG.2010.179