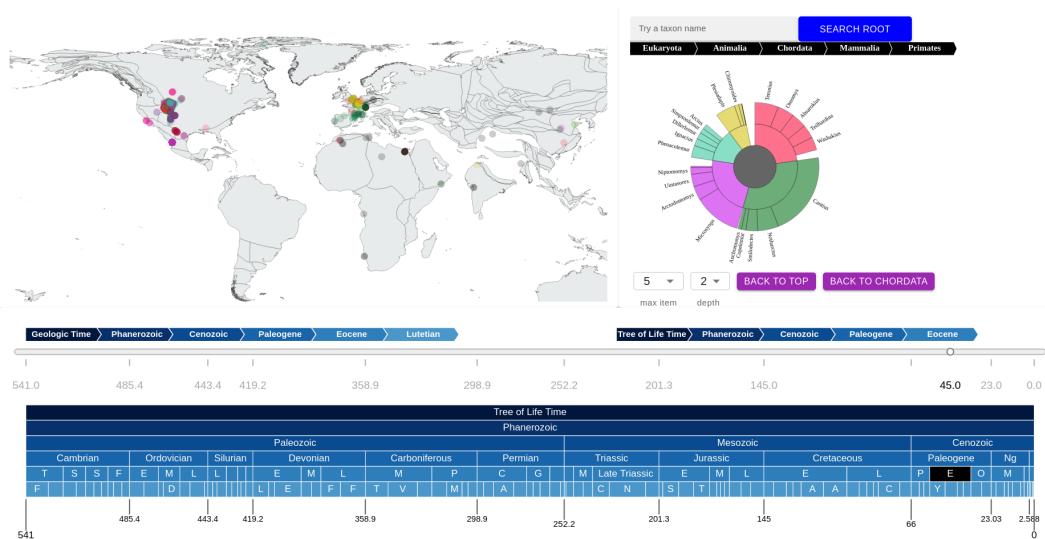


A visualization of biodiversity through time



Master Thesis
January 31, 2022

by Wuxing Dong, 18-946-038

Supervisors:
Prof. Dr. Renato Pajarola
Gaudenz Halter

Visualization and MultiMedia Lab
Department of Informatics
University of Zürich



University of
Zurich UZH

V
I
S
U
A
L
I
Z
A
T
I
O
N
A
N
D
M
U
L
T
I
M
M
E
D
I
A
L
A
B

VISUALIZATIONANDMULTIMEDIALAB

Abstract

The explosion of data in recent years has inspired interesting visualizations to communicate data-driven ideas in intuitive and memorable ways. The current project addresses the topic of biodiversity through time. With the collective effort of paleobiologists, online databases have been constructed to store locations and taxonomy identifications of fossil records collected all over the world. In the meantime, recent progress in Earth science has tracked how continents continuously shifted in the past 410 million years in high spatial-temporal resolution. Therefore, it is now possible to design a visualization that combines both data to illustrate how life and Earth together changed in the course of time.

This thesis purposed a complete workflow from data processing to data visualization in a web application. We clean all 1.3 million animal and plant fossil data in the PaleoBiology Database, and organize their taxonomic information into a tree of life. We reconstruct fossil locations at various ancient time points using a published simulation model on tectonic movement. The final result provides a rich and engaging visualization prototype that allows users to explore the tree of life, Earth geography, and fossil distribution in the world during Phanerozoic Eon. To our knowledge, this is the first interactive visualization that combines tectonic motion and tree of life across time, thereby providing users a holistic view of how different forms of life developed and where they lived in various periods of Earth history.

Contents

Abstract	ii
1 Introduction	1
1.1 Effective visualization of data	1
1.2 Background on biodiversity and tectonic change	2
1.3 Aim of the thesis	3
1.4 Thesis structure	3
2 Related work	4
2.1 Tree-based visualizations on biodiversity	4
2.2 Visualizations on tectonic shift	6
3 Problem statement	8
4 Technical solution	9
4.1 Overview	9
4.2 Data processing	10
4.2.1 Raw data parsing	10
4.2.2 Fossil point dataset preparation	11
4.2.3 Tree dataset preparation	13
4.2.4 Geospatial transformation of map and fossil data	14
4.3 Database setup	15
4.4 Back-end query processing	15
4.5 Front-end development	17
4.5.1 The global states	17
4.5.2 The user interface: time control component	18
4.5.3 The user interface: tree component	21
4.5.4 The user interface: map component	25
5 Implementation	26
5.1 Dependencies	26
5.2 Code structure	26
5.3 Data processing	27
5.4 Web development	28
5.4.1 Back-end	28
5.4.2 Front-end	29
6 Results	32
6.1 Data processing	32
6.2 Data visualization	35
7 Conclusion	39
7.1 Limitations and future Work	39

1 Introduction

1.1 Effective visualization of data

In the information age, the Internet has become a primary source to inform ourselves about the world. It is therefore a great opportunity for the scientific community to take advantage of the web and create high quality products to communicate their findings and ideas to the general public. Textbook-styled verbal explanations can be rigorous, however, they easily become tedious and are already made available by school and college education. Videos are based on visual learning and are quick to absorb, yet there is little engagement with the viewers. Interactive data visualizations provide an excellent way for users to understand scientific concepts. New knowledge is conveyed by direct visual impact that easily draws attention, and insights can be formed by actively exploring the data and observing the response of the presentation due to the users' own actions. This learn-by-doing strategy is considered to be beneficial for consolidating new knowledge [Sch95].

Although graphics play a major role to deliver scientific messages, there is a lack of formal theory on producing good graphics, and domain experts tend to rely on practical experiences that are guided by certain design principles [CHU07]. Increasing amount of studies have focused on concluding such principles [SI90, EM13, Mid20]. Chen and colleagues [CHU07] have summarized that three main aspects to pay attention to in a visualization project are its contents, the relevance in the context of the larger whole as well as the forms and aesthetics (the construction). Similarly, McCandless [McC14] argued that a good visualization requires the integrity of information, the interestingness of story (concept), the usefulness of goal and the beauty of the visual form.

Before embarking on producing a visualization, one should note the characteristics of human visual perception to improve the design strategies in practice. Cognitive psychology studies are aimed at examining human visual attention by exposing users to certain visual processing tasks. The results of those studies suggested some guidelines for more effective visualization. When the visual features of a certain type of data is expected or previously known, the accuracy and speed of visual perception is drastically improved [HW12]. Therefore, the authors suggested that a visual feature should be assigned to the same data dimension, or type of data [HW12]. It is also important to note that attention is limited and selective, therefore the amount and diversity of information should be kept within the limits of cognitive load [Alh18]. When many categories of data are present, compromises are needed to retain graphics complexity and only demonstrate a few categories at a time.

Through early years of research on how images are analyzed visually, it has been found that detailed vision is only achievable by actively focusing on a small portion of the visual field [IK01, NS71]. However, a few visual features, such as colors and curvatures, are subjected to pre-attentive processing, which can be effortlessly and rapidly detected by low-level visual processes [HE11]. Therefore, the appropriate use of limited but distinctive colors and simple shapes is a powerful technique. They are also the leading factors to make a visualization memorable [BVB⁺13].

With respect to the design process, it should be guided by the core messages to deliver. It is important to first prioritize what information to show, envision the final result, and then choose appropriate colors and geometries that fulfill the objectives [Mid20]. The most efficient geometry should be of high data-ink ratio, which is the ratio between the amount of ink allocated on the actual data to the total amount of ink a figure consumes [TGM83]. While striving for a high data-ink ratio, it is also important for a visualization to be simple, self-explanatory and straight to the point, such that the viewers are not overwhelmed with too much information.

The topic this thesis chooses to visualize is biodiversity through Earth history. There are biodiversity visualizations that are tree-based [BHP⁺12], map-based [JNZ⁺16], or plot-based [LMD⁺19]. However, there lacks a rich and interactive visualization on the Internet that illustrates biodiversity with both a dynamic tree of life and a map that locates species on the planet throughout millions of years. This thesis is intended to fill this gap. Furthermore, it is important to raise the awareness of the general public on ecological issues, in light of the current age of climate change. By exploring how different forms of life came to exist, thrive, and perish through a simple and interesting visualization, viewers could appreciate the inseparable relationship between life and its environment.

1.2 Background on biodiversity and tectonic change

Biodiversity is a result of evolution. Current views on species formation are based on modern evolution theory [Bow89] originated from Darwin's work. To summarize briefly, all forms of life are argued to share a common ancestor, which diversified over the course of evolution to form the current tree of life. An organism's genetic information is encoded in DNA or RNA, which is replicated over generations. However, recombination and random mutation events occur during the replication process, resulting in genetic variations, which in turn cause the difference in phenotype, *i.e.*, the observable traits of organisms. Individuals with traits that are more suitable to their particular environment are more likely to survive and pass their genes to their offspring, compared to those that are poorly adapted to the environment. This process is termed natural selection. Finally, a new species is formed during the process of speciation. It occurs when enough genetic variations are accumulated between the original population and the descendant population, such that organisms from the two groups are not capable of producing fertile offspring, namely, the two groups have formed reproductive isolation. A classic example of evolution is the adaptive radiation of finches on the Galapagos islands [Lac83]. The beak shapes of the finch species on different islands diversified to specialize in obtaining the dominant type of food on those islands.

It is believed that life emerged from 3.5 billion years ago [AS98]. The past 540 million years (the Phanerozoic Eon) have seen rapid development and diversification of life. Throughout evolutionary history, there are records of drastic changes in biodiversity occurring in a relatively short period of time. On one hand, radiations are rapid increase in biodiversity. An example is the Cambrian explosion, during which most phyla of multi-cellular organisms originated [AMB⁺01]. On the other hand, biodiversity experiences rapid declines during mass extinction periods. There is evidence of five mass extinction periods, one of which is the Cretaceous-Paleogene extinction event, when the dinosaurs were exterminated and mammals thrived due to their selective advantage given by their fur in the cooling environments [Cou02]. These periods of radiations or extinctions are caused by special events, plate motions, as well as changes in climate, such as temperature and oxygen level [But09, SS07].

Apart from biodiversity, the shape of the world also changes significantly throughout billions of years [Wil63]. Earth crust is composed of a number of tectonic plates undergoing constant motions. The heat generated from the interior of the planet causes the movement of the plates. Plate motion results in new landscapes and new continents, affecting lives inhabiting them. For instance, current continents are thought to belong to one super-continent called Pangaea [SHV⁺13], which gradually assembled approximately 335 million years ago (Paleozoic-Mesozoic) and drifted apart at around 175 million years ago (Triassic-Jurassic).

As the continents separate, populations from the same species are geographically isolated into their particular environments resulting from plate motion. Since organisms are under constant selective pressure, new species evolve in the newly created environments. Numerous studies support that tectonic movement plays an important role in driving biodiversity in various regions of the world. To list a few examples, Miocene tectonics contributed to the diversification of ungulate and carnivores in western United States [KF08]; the formation and break-up of Pangaea have driven global marine invertebrates diversity for the past 443 million years [ZFP17], and the beginnings and movements of tropical reef biodiversity hotspots can be predicted by modeling plate motion for the past 140 million years [LDG⁺16].

1.3 Aim of the thesis

This thesis aims at building a web application that visualizes the change in biodiversity through the course of Earth history. In the final product, users can view a customizable tree of life during a selected time period under a selected taxonomy. In the meantime, the fossil records associated with the displayed tree of life are shown on a map of the selected time period.

1.4 Thesis structure

Chapter [1](#) introduces the topic of data visualization and domain specific background information, and it claims the goal of the thesis. Chapter [2](#) analyzes some related work, indicating what could be learned and improved for the visualization to be built. Chapter [3](#) outlines the problems to be solved at each stage for building the web application. Chapter [4](#) discusses the technical solutions for solving the stated problems. Chapter [5](#) addresses how the technical solutions are implemented to result in the visualization product. Chapter [6](#) reports the result of the product. Chapter [7](#) concludes the project and suggests future work.

2 Related work

There already exists an abundance of visualizations on the Internet regarding to tree of life as well as plate motion. This section discusses some selected works on these two topics.

2.1 Tree-based visualizations on biodiversity

OneZoom (<https://www.onezoom.org/>, Figure 2.1) visualizes all known living things in a fractal tree, and it is targeted for both professionals and the masses [RH12]. Users can zoom into or out of different levels of the tree by scrolling, and they can search for a particular node by its name.

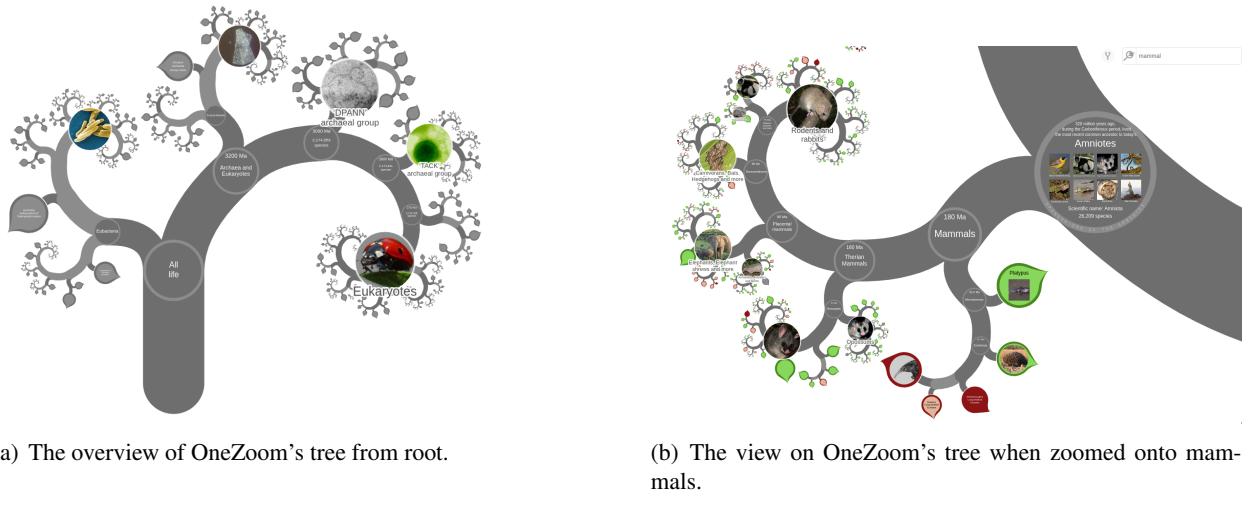


Figure 2.1: Tree of life implementation in OneZoom.

The fractal design gives the tree a nice appearance, with a high resemblance to an actual tree. Information such as images or the earliest time of a node is also well packed in the interactive image. There is a reset button to return to the root and a drop-down list that provides links to the nodes along the path from the current position to root. These functions are convenient for navigation and similar functionality should be implemented in this thesis. It is claimed that all categorized organisms, with a total amount of over two million, are included in the tree. Each species in the tree is incorporated with a phylogenetic classification. This leads to the fact that when users attempt to visit the root from a given species on a leaf node, they visit all of the nodes representing each of its ancestors throughout evolutionary history. Although how a certain organism evolved from the common ancestor of all life can be rigorously inspected, from the user experience's point of view, the tree is extremely deep. Apart from manually searching for a name, navigating to the desired location in this deep tree is mainly achieved by scrolling and dragging the item of interest to the center of the screen. This makes it difficult for viewers to relocate to a different section of the tree, or to gain an overview of where the current node positions in the entire tree of life. To reach the root of all life (Figure 2.1(a)) from the mammal (Figure 2.1(b)), users need to scroll through all upper layers of the tree, which have a similar-looking spiral structure that appears to be confusing and disorienting. A similar tree of life visualization is Lifemap (<http://lifemap.univ-lyon1.fr/explore.html>, [dV16]), which ameliorates the navigation difficulty by highlighting the ancestry path from root to a node (Figure 2.2). However, due to the sheer depth of the tree, users still need to travel deep into the tree by scrolling. Rather than demonstrating the exact ancestral paths, the purpose of the tree for this project is to organize each species

2 Related work

such that biodiversity can be viewed in an intuitive way. As a consequence, all nodes can be organized by the main standard taxonomic ranks, in the descending order of domain, kingdom, phylum, class, order, family, genus, and species. This limits the tree's depth to be eight and thus simplifies the navigation process. In addition, to avoid scrolling and dragging, the tree can take a fixed position. Users can select a node of interest as a root, and only the subtree that is rooted by the chosen node is shown. In the meantime, the location of the selected node in relation to the complete tree of life can be shown in a clickable breadcrumb structure that 1, provides an overview of the whole tree by demonstrating the path from the root of all life to the current root and 2, allows users to easily navigate to any of the upstream nodes in the tree by clicking.

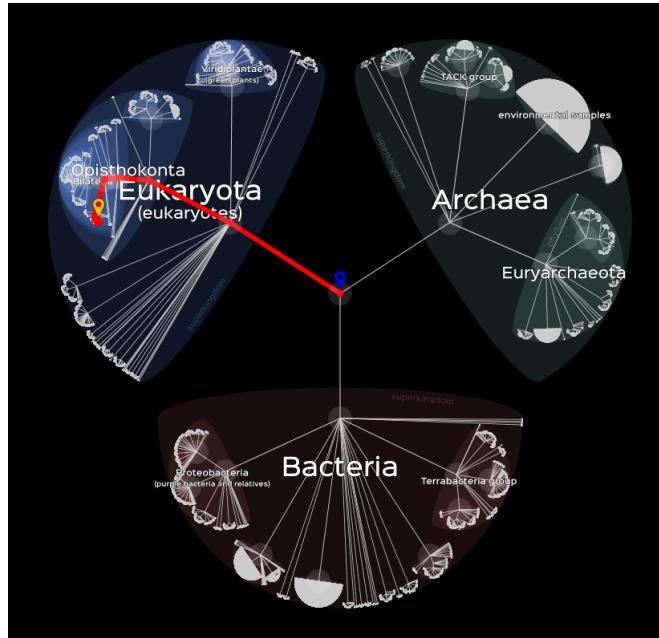


Figure 2.2: Lifemap enables a "path from root" functionality that highlights the ancestry path from *Homo sapiens* to the root. Although due to its small size, the name *Homo sapiens* is not longer visible.

A much more simplified version of the tree of life is Evogeneao (<https://www.evogeneao.com/en/explore/tree-of-life-explorer>, no publication), which is intended for school education. As shown in Figure 2.3, all species are clustered into clearly colored regions, giving the viewers a direct and memorable visual impact. A similar coloring and grouping design could be adapted in this project. With the tree of life arranged in a half-circle that is labeled with time and a few iconic events, users could get a view of how life radiated from the common ancestor into great diversity, and how this process is affected by special events on the planet. Since only a small subset of all organisms are shown, the tree is small and can be entirely shown, therefore always giving the audience an overview. However, this web application provides the little possibility of exploration due to its limited options for interaction, as the main functionality is to show the common ancestor of two selected items (Figure 2.3(b)).

Finally, there are a wide variety of tree of life visualized for computational biologists, such as AnnoTree [MCP⁺19], phylogeny.IO [JM19], T-BAS [CWM⁺17] and PhyloWidget [JP08], to list a few. These visualizations provide rich functionality for bioinformatic analysis that are well beyond the need of the general public and often make efforts to use. A typical example is iTOL (<https://itol.embl.deitol.cgi>, Figure 2.4, [LB19]). The design of arranging all leaf nodes on the edge of a circle and all non-leaf nodes in its interior utilizes space effectively, therefore providing a good hint for our implementation.

2 Related work

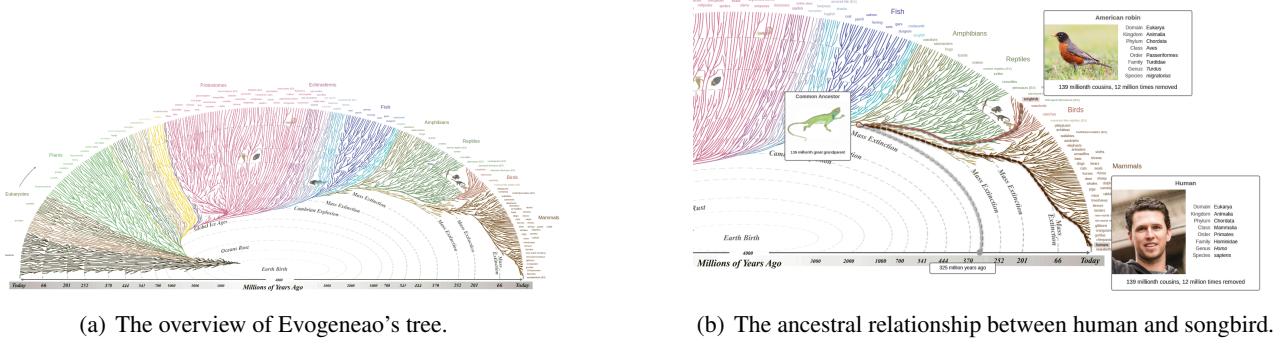


Figure 2.3: Tree of life implementation in Evogeneao.

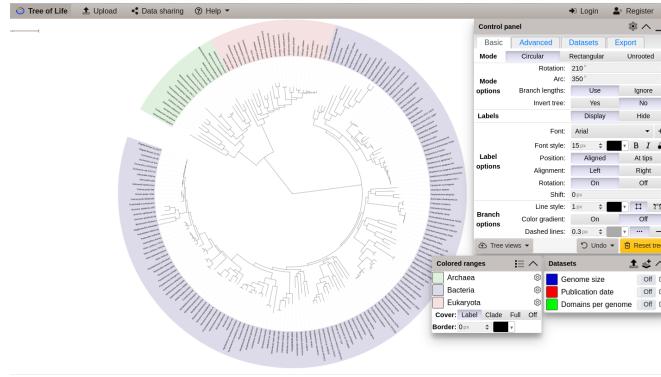


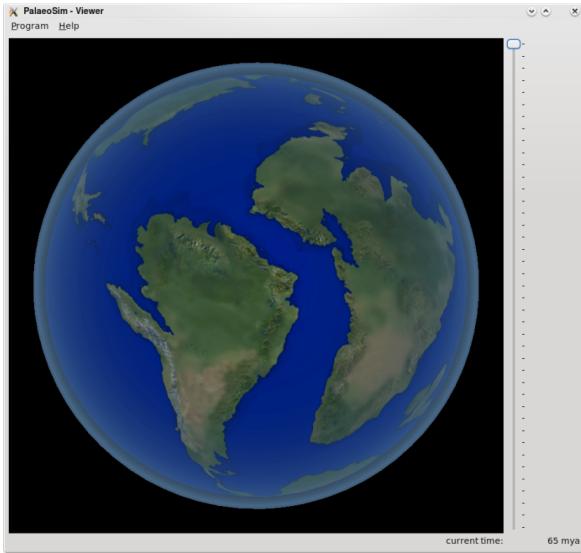
Figure 2.4: Tree of life in iTOL.

2.2 Visualizations on tectonic shift

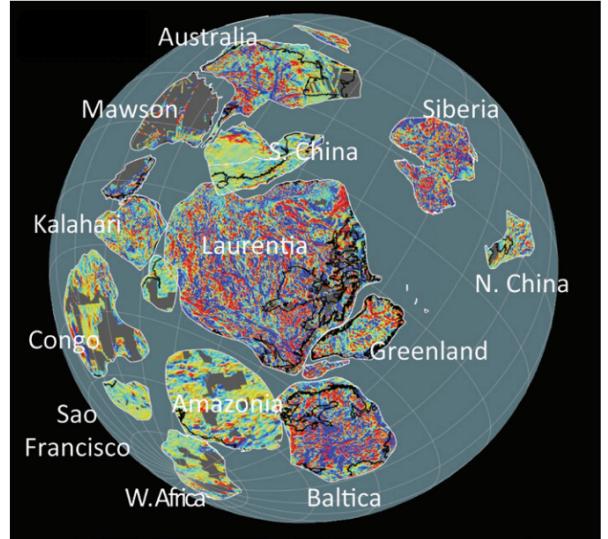
The current project involves visualizing paleogeography at various time points, since fossil points are displayed on a world map during the time period of interest. Rather than web-based, most visualizations on tectonic changes are commercial software products that are primarily intended for inspecting global petroleum and ore reserves. Many products targeting the scientific community fall into the category of GIS software [SWW10]. They are designed for analytic purposes. However, they are necessarily intuitive to operate or visually pleasing, such as GMAP [TS99], PaleoMac [Cog03], PLACA [MOAF05] and PPlates [SKR⁺07].

In more recent years, PaleoSim (Figure 2.5(a), [RLM10]) visualizes the ancient Earth by proposing a tectonic morphing model that simulates a smooth transformation of plates through time from image data. Users can select a time point or a particular plate and observe its shape throughout ancient history. However, this offers limited functionality, as it is not capable of tracking the location change of a given point on the map, which is needed for visualizing fossils. Nevertheless, GPlates (Figure 2.5(b)) is a next-generation software widely used for plate tectonics purposes [BMG¹¹] [WMLW12] [GTZ¹²]. In short, the software receives a tectonic simulation model that defines how the movement of plates occurs through time, and vector data that represents geospatial information in points, lines, or polygons. It can then assign all geospatial data to a plate-tectonic reference, before reconstructing the geography at a requested time point and exporting the reconstruction into files. In relation to our project, it provides a method to trace a fossil point by assigning it to a certain plate, such that how its position alters can be calculated with a given tectonic model. As a result, this tool is used for our project, to reconstruct both the world map and the fossil points at a given ancient time point.

2 Related work



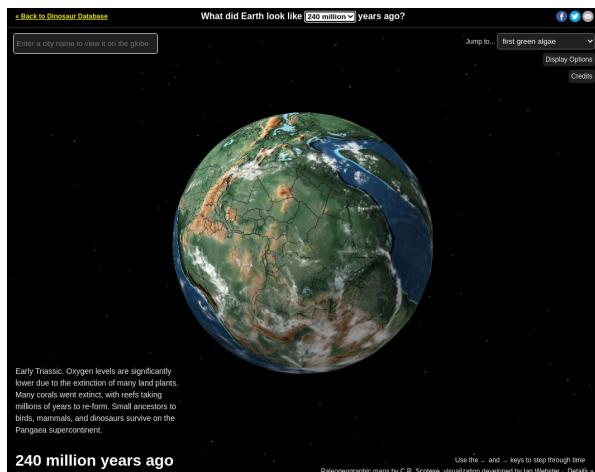
(a) PaleoSim visualization on South America and Africa 105 million years ago. The figure is directly adopted from [RML10].



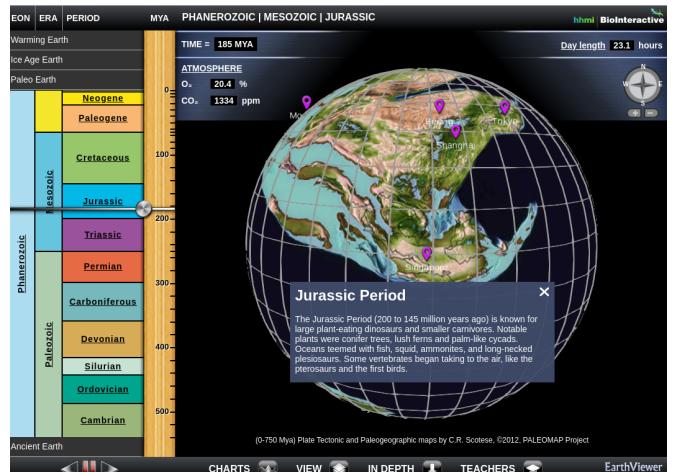
(b) GPlates visualization of ancient earth at 710 million years ago, based on a model by [LBC⁺08]. The figure is directly adopted from [WMLW12].

Figure 2.5: Example software-based visualizations on tectonic shift.

A few web-based projects on continental drift visualization can also be found. Two typical products are Ancient Earth (<https://dinosaurpictures.org/ancient-earth#240>, Figure 2.6(a), no publication) and Earth Viewer (<https://www.biointeractive.org/classroom-resources/earthviewer>, Figure 2.6(b), no publication). Both of them show the coastlines of the continents at a select time point on a three-dimensional globe. Ancient Earth allows users to pick a particular time point from a list of options and search for the ancient location of a city. Earth Viewer provides more detailed information on geological time, and selectable time points are incremented every five million years on the scroll bar. Time periods are organized in a hierarchical structure, which can be adapted in this project for easy navigation. When clicking on a time period, instead of directing the user to view that time period, a pop-up window containing a brief introduction appears, which is counter-intuitive. An improvement is to show the Earth of the selected time period upon clicking.



(a) Tectonic shift visualized in Ancient Earth.



(b) Tectonic shift visualized in Earth Viewer.

Figure 2.6: Example web-based visualizations on tectonic shift.

3 Problem statement

The data visualization project is divided into several sub-problems as follows.

- **Data processing**
 - Process the PBDB dataset to extract fossil records
 - Extract a tree of life from the taxonomy information of the fossil records
 - Trace the coordinates of each fossil record to ancient times
- **Database setup:** store the fossil data, coordinate data and the taxonomy data into a database
- **Back-end building:** provide a server for queries from the database upon front-end requests induced by users
- **Front-end building:** visualize interactive components for users to explore biodiversity through deep time
 - A time component that lists ancient time periods that users can select
 - A tree component that shows the tree of life constructed by the fossil records found in the selected time period, under the selected taxonomy
 - A map component that shows the shape of the continents in the selected time period, and the locations of the fossils that constitute the tree of life

4 Technical solution

4.1 Overview

This section explains the technical solutions to the problems stated in chapter 3. To summarize, the project is divided into data processing in section 4.2, database setup in section 4.3, back-end query processing in section 4.4 and front-end development in section 4.5.

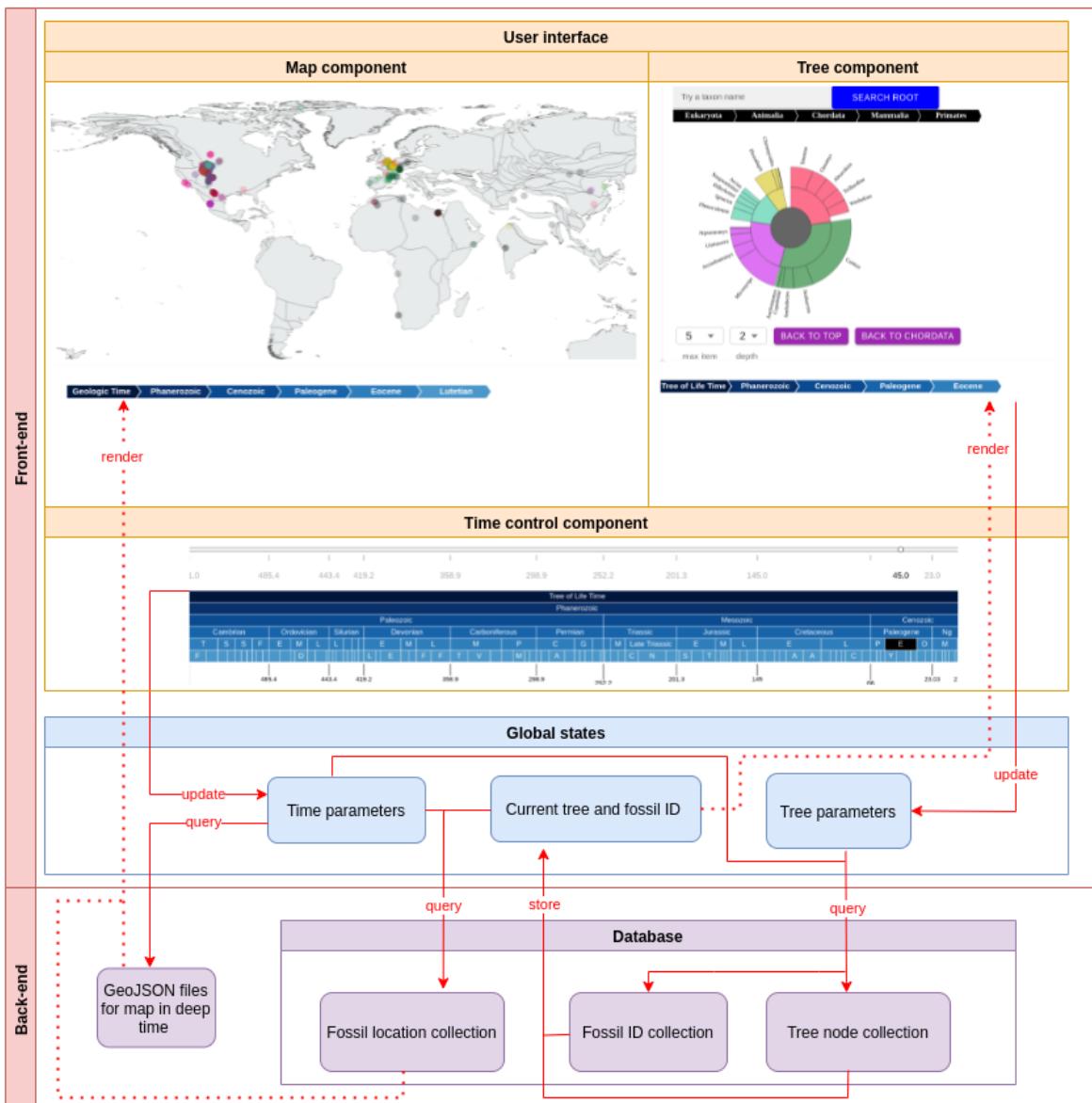


Figure 4.1: An overview of the architecture solution for the project. The user interface consists of a map, a tree, and a time control. The global states store tree and time parameters that are customized by users, and the current tree nodes attached with fossil ID sent from the database. The back-end resolves queries on the tree, fossil locations, and map, before sending results back to the front-end, where the data is visualized.

Diagram 4.1 proposes a structure of the designed visualization system. The front-end of the web application has a user interface and global states section that stores variables accessible to the components of the user interface. The back-end stores data either in the database or as static files, processes the front-end's data requests, and sends data back to either global states for further processing and storage, or directly to the user interface for visualization. Map, tree, and time control are the three components that users can view and interact with. The time control panel allows users to select a time to either view the tree of life and the related fossils, or to track the fossil location change due to plate motion. The *Time parameters* from global states section listen to user-induced changes from the time control panel, and trigger requests of map data, tree data (together with tree parameters stored in global states), or fossil location data (together with fossil ID information stored in global states). The map component receives 1, coastline boundary data from GeoJSON static files, and 2, fossil location data from the database, to render world map and fossils at a time point. The tree component shows the tree of life stored in *Current tree and fossil ID*. When users customize the tree view, the tree parameters are updated, and new data on tree nodes with attached fossil IDs is queried from the database. The data is returned to *Current tree and fossil ID*, which in turn processes the data and sends only the tree node information to the tree component to be rendered as the new tree.

4.2 Data processing

4.2.1 Raw data parsing

The data source for fossils and the tree of life is provided by the Paleobiology Database (PBDB) [PM16]. The raw data consists of two CSV files that contain animal and plant fossil records, respectively. Since the two files share the same data attributes, the data is concatenated into one file. Only the attributes that are relevant to this project are parsed, which are listed below.

Extracted attributes in the PBDB dataset:

- *occurrence_no*: the unique ID code for a fossil record
- *accepted_name*: the accepted name of a fossil record
- *accepted_rank*: the taxonomic rank in which the accepted_name belongs to
- *phylum*: the phylum name of the fossil record
- *class*: the class name of the fossil record
- *order*: the order name of the fossil record
- *family*: the family name of the fossil record
- *genus*: the genus name of the fossil record
- *max_ma*: the oldest estimated time of the fossil record, in million years
- *min_ma*: the earliest estimated time of the fossil record, in million years
- *long*: the longitude of the fossil record's found location
- *lat*: the latitude of the fossil record's found location

In addition, we append a new attribute to the dataset to keep note of which one of the two files a fossil record is originated from:

- *kingdom*: the kingdom name of the fossil record

The kingdom of "Animalia" and "Plantae" is assigned to each fossil record, depending on whether it is from the animal or plant file, respectively.

By observing the available attributes in the raw data, we can determine how they can be used for the project. The biodiversity of a given time period is represented by 1, a dynamic tree of life formed by the fossil records during that time period, and 2, the locations of the fossils on a world map of that time period. The *occurrence_no* uniquely identifies a fossil record. The *max_ma* and *min_ma* together describe during which time period(s) the organism was alive. The *lat* and *lng* are used to calculate the location of a fossil at an ancient time point. Most importantly, a tree of life is needed to host each fossil record on a specific node. To generate a tree structure, information on the parent-child relationship is essential. This relationship is represented in the names on each of the taxonomic ranks, in the descending order of *domain*, *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, *species*. The raw data lacks information on *domain* and *species*. However, since all records are either animals or plants, they all belong to the common domain of "Eukaryota". A fossil record has a species name only if it is identified down to the species level. Then, the value of *accepted_rank* is "species", and the species name is found in *accepted_name*. Therefore, the raw data has sufficient information to construct a tree of life with the eight above-mentioned taxonomic ranks as its levels.

The datasets on fossils and the tree are created from this parsed raw data. In the next steps, we first prepare the dataset that contains all fossil points, since the tree dataset is created based on it.

4.2.2 Fossil point dataset preparation

Multiple coordinates exist for each of the 1.3 million fossil records, as a certain location on the planet varies at different time points throughout Earth history. As a result, two datasets are created regard to fossil data. This section addresses the first dataset *fossil point*, which concerns the taxonomy and the range of age for each record, and sub-section 4.2.4 addresses *fossil location*, the second dataset that stores the coordinates of each fossil record at various time points.

The purpose of *fossil point* dataset is to accurately describe a given fossil record's age and its position in the tree of life, without considering its geological location. The parsed raw data already provides *occurrence_no*, which can serve as a fossil's unique identifier. However, one more attribute is required to describe its taxonomy, such that it can be attached to a particular node in the to-be-constructed tree of life. This attribute is essentially the unique identifier of a node in the tree of life, such that a given fossil record is attached to one and only one node in the tree. It is thus also the link between the *fossil point* dataset and the dataset for the tree. Although it is natural to consider *accepted_name* and *accepted_rank* in the parsed raw data for this purpose, they cannot be used for the following reasons. Firstly, *accepted_name* and *accepted_rank* does not correlate well with the names on the taxonomic ranks, as some *accepted_rank* values do not belong to any of the eight ranks that host the parent-child relationship in the tree. As a consequence, these nodes defined by *accepted_name* and *accepted_rank* cannot be placed in the tree, leaving their fossil records excluded from the tree as well. Secondly, the *accepted_rank* and *accepted_name* attributes are not unique. For instance, the genus "Banksia" is both an animal taxonomy and a plant taxonomy. In such cases, fossil records can be associated with an incorrect branch if only these two pieces of information are used. Thirdly, many names in the taxonomic ranks are missing. This leads to ambiguity when attaching a particular fossil to the tree of life when there exists a rank above the *accepted_rank* with missing names. An example is shown below.

occurrence_no	accepted_name	accepted_rank	kingdom	phylum	class	order	family	genus	...
3285	Tetradium	genus	Plantae	Rhophyta	NaN	NaN	NaN	Tetradium	...

Table 4.1: An example row of parsed raw data with missing information.

With only the *accepted_name* "Tetradium" on the *accepted_rank* "genus", we are uncertain about which class, order, or family to associate this fossil record to under the phylum of "Rhophyta". Extra knowledge in taxon-

4 Technical solution

omy is needed to retrieve the missing names on the path from "Rhodophyta" to "Tetradium" and this challenge is beyond the scope of the current project. The current solution only attaches the fossil record "3285" to the phylum of "Rhodophyta" and ignores its taxonomy below the phylum level.

To solve these three issues and provide a method for uniquely identifying a fossil record to the tree without ambiguity, the attributes *path_from_root* and *rank* are introduced.

- *path_from_root*: a comma separated string indicating the node's path from the root "Eukaryota" to the name on lowest possible taxonomic rank with no missing names in between
- *rank*: the lowest taxonomic rank of *path_from_root*

A path begins from the common root "Eukaryota" and terminates when reaching the lowest rank level "species", or encountering a missing name. For instance, the fossil record in Table 4.1 has "Eukaryota, Plantae, Rhodophyta" as *path_from_root* and "phylum" as *rank*. All names in *path_from_root* are guaranteed to be present in the tree since the names are taken from the same taxonomy information that is used to construct the tree. The path from the common root Eukaryota describes the exact location of a node in the tree, which serves as the node's unique identifier. In addition, an important search scenario is to retrieve all fossils associated with a taxonomy's subtree. Apparently, those fossils' *path_from_root* values start with the path of the taxonomy of interest. Therefore, they can be easily obtained by string matching this path using regular expression.

To calculate *path_from_root* and *rank*, the following attributes are used: *domain*, *kingdom*, *phylum*, *class*, *order*, *family*, *genus*, *accepted_name*, *accepted_rank*. Instead of iterating over each row in the parsed raw data, we first aggregate the rows when those values are identical and then perform the iteration, before joining the result with the original data (similar to an "inner join" operation in SQL), in order to avoid repetitive computation.

At each iteration on the aggregated data, we execute the following algorithm. We initialize the *path_from_root* as "Eukaryota" and visit each taxonomic rank from *kingdom* to *genus*, while appending a comma and the name on each rank to the end of the string and updating the current *rank*. The iteration terminates upon encountering the first missing name. At this point, both *path_from_root* and *rank* are those of the missing name's parent. If there are no missing names until visiting *genus*, the iteration finishes when the *accepted_rank* is also "*genus*". Otherwise, the *accepted_rank* could be either "species" or "subspecies". Under this circumstance, we append the value of *accepted_name* to *path_from_root*, update *rank* to be the value of *accepted_rank*, and terminate the iteration. The name in the path after *genus* can thus be either a species or a subspecies, which is acceptable since a subspecies name already includes the species name (*i.e.*, the species name of the subspecies "Argopecten circularis impostor" is simply "Argopecten circularis"). The implementation of this method can be seen in the function *append_path_and_rank()* in file *source/data_processing/2_prepare_fossil_point_data.py*.

It is important to note that the columns *path_from_root* and *rank* are precursors to the tree dataset since they represent all nodes with at least one fossil record attached. The unique values of the two columns are used for further processing in sub-section 4.2.3. In the meantime, the aggregated data with *path_from_root* appended are joined with parsed raw data, such that every fossil record has a path. Finally, the *fossil point* dataset is generated, by extracting the following attributes:

- *occurrence_no*
- *path_from_root*
- *max_ma*
- *min_ma*

4.2.3 Tree dataset preparation

The purpose of the *tree* dataset is to describe every node on the tree of life that can also be easily searched. The following attributes are included:

- *path_from_root*: a comma separated string indicating the path from the root Eukaryota to the name on lowest possible taxonomic rank with no missing names in between
- *parent*: the *path_from_root* of its parent
- *name*: the name of the node
- *is_leaf*: a boolean value that indicates if the node is a leaf node, *i.e.*, has no children
- *max_ma*: the earliest age in million years among all fossils in the subtree of this node
- *min_ma*: the latest age in million years among all fossils in the subtree of this node

For each node, we need a unique identifier, which is provided by *path_from_root* as discussed in the above sub-section 4.2.2. The last *name* of the path is also extracted as an attribute, such that the nodes can be searched via name. The node's *parent* attribute simplifies searching for all children of a given node. The *is_leaf* attribute indicates directly whether or not a node has children without searching for them. Lastly, *max_ma* and *min_ma* together provide a quick way to tell if a node exists during a given time period without searching for fossils attached to its subtree.

To prepare this dataset, we start from the precursor dataset from 4.2.2, which has two columns: *path_from_root* and *rank*. As a reminder, the *path_from_root* are the unique identifiers of all nodes in the tree with at least one fossil record attached. However, there are more nodes to include in the tree. For each row in this precursor dataset, we must also include each of the upstream ancestors in the tree. For instance, when a node "a,b,c,d" exists, its ancestors "a,b,c", "a,b" and "a" should also be included in the tree. Therefore, the first step is to add the missing ancestors, by iterating every *path_from_root*. Given a path, the *path_from_root* values of all its ancestors are simply strings sliced from the beginning of the path, until each of its commas. The *rank* value of an ancestor can also be computed by counting the number of commas in the path, as a "domain" has no comma in its path, and a "genus" has six commas. After the iteration, only the unique *path_from_root* and the corresponding *rank* values are kept. The implementation of this method is seen in the function `get_additional_nodes()` in file `source/-data_processing/2_prepare_tree_data.py`.

The *parent* and *name* values can be easily computed from a *path_from_root*. The common root "Eukaryota" has no parent, and any other node's parent is the sub-string of its *path_from_root* sliced from beginning until the last comma (*e.x.*, the parent of "a,b,c" is "a,b"). Similarly, the *name* of a node is the sub-string beginning from the first character after the last comma in its path (*e.x.*, the name of the node "a,b,c" is "c"). Next, we visit each *path_from_root* once again. If the path is in the set of all *parent* values, then this node is a parent (*i.e.*, not a leaf node), and its *is_leaf* value is set to be false, otherwise true.

Next, we calculate the *max_ma* and *min_ma* values for each node, using the processed *fossil point* data from 4.2.2, which now contains the path name and the age range information for each fossil record. We first create a reference dataset, by grouping the fossil data by the same *path_from_root* on two attributes: 1, maximum of *max_ma* values among the fossils with identical path and 2, minimum of *min_ma* values. Next, we iterate through this reference dataset. For each path, we extract all its ancestors using the same string splitting method as mentioned above, visit those ancestors in the tree dataset, and 1, update its *max_ma* only if it is less than the current *max_ma* of the path under iteration and 2, update its *min_ma* only if it is greater than the current *min_ma*

of the path under iteration. The implementation of this method is in the function `append_time()` in file `source/data_processing/2_prepare_tree_data.py`.

Lastly, it is found from the *tree* dataset that the nodes' names are not unique. There are in total 64 cases where a name has exactly one duplicate. This leads to ambiguity when a user searches for a name that is not unique in the tree of life. To solve this issue, we extract all repetitive names, and record an array of "unique names". The unique name designed for a repetitive name is a dash-separated string with the repetitive name itself on the left, and the first distinctive ancestor from the root on the right. For example, the unique name for the node "Eukaryota,Animalia,Arthropoda,Insecta,Megaloptera,Corydalidae,Corydalis" is "Corydalis-Animalia", and the unique name for the node "Eukaryota,Plantae,Spermatophyta,Magnoliopsida,Ranunculales,Fumariaceae,Corydalis" is "Corydalis-Plantae". This list of unique names provides the user the options to choose which node to view when searching for a repetitive name and the information will be passed to the database to search for the exact node of interest.

4.2.4 Geospatial transformation of map and fossil data

Apart from the tree, a major component of the visualization is a world map responsive to time selection, on which the locations of the fossils are also shown. Therefore, given a geological timescale, we need to calculate the world maps in all time periods. For each fossil point, its coordinates during those time periods are also calculated and stored in the *fossil_location* dataset.

The timescale data is provided by PBDB. Similar to the tree of life, it is also hierarchical, with five layers in total. From the highest layer to the lowest layer, the time periods are in the geochronologic units of Eon, Era, Period, Epoch, and Age. The highest layer Phanerozoic (an Eon) ranges from 541 million years ago to the present, and the intervals on the fifth layer (Age) often spans only several million years. Nevertheless, plates move constantly, such that the location of a given point at the end of a time interval varies from the location at the start of the interval. Our solution is to represent the geology of a time period using the middle value of a certain Age interval. When the user selects a time interval in the fifth layer, the coordinates are transformed to that of the interval's middle time point rounded to an integer. When a higher-layer time interval is selected, we first find the Age in which the middle value of the selected interval falls, before transforming the coordinates to the integer time point in the middle of that Age.

Both the global coastlines boundaries of modern-day and the model to transform a modern location to an ancient location are provided by Matthew *et al.* [MMZ⁺16]. In this paper, they proposed a high-resolution model on the movement of plates from the late Paleozoic (410 million years ago) to nowadays. This model transforms the global coastlines or points to a desired time point no earlier than 410 million years ago. In the parsed raw PBDB data from sub-section 4.2.1, we extract the *lat* and *lng* attributes as the modern coordinates of a fossil point, and *occurrence_no* as the ID for each fossil point. For a given point on the planet, the model assigns it to a plate, before calculating its ancient position as it relocates along with the movement of its assigned plate. Finally, the map data is stored as GeoJSON files, and the ancient locations of each fossil point are stored in the *fossil_location* dataset with the following attributes:

- *occurrence_no*: the ID of a fossil record
- *mya*: the time point as an integer in million years
- *lat*: the latitude of the fossil record in the corresponding time point
- *lng*: the longitude of the fossil record in the corresponding time point

4.3 Database setup

The fossil and tree information is stored in a database, from which requested data is sent for visualization according to queries generated from user interactions. The three prepared datasets are uploaded to the database, with slight renaming and reorganizing in the following manner:

- *fossilLocation*: It stores the *fossil location* dataset, which contains the locations at different time points for each fossil record. It has the following fields: *id* (*occurrence_no* from *fossil location*), *mya* (*mya* from *fossil location*), *coordinates* (an array of [*lat,lng*] from *fossil location*).
- *fossilPoint*: It stores the *fossil point* dataset, which contains taxonomy and range of age for each fossil record. It has the following fields: *id* (*occurrence_no* from *fossil point*), *maxma* (*max_ma* from *fossil point*), *minma* (*min_ma* from *fossil point*), *pathFromRoot* (*path_from_root* from *fossil point*).
- *TreeNode*: It stores the *tree* dataset, which contains each node in the tree of life. It has the following fields: *pathFromRoot* (*path_from_root* from *tree*), *maxma* (*max_ma* from *tree*), *minma* (*min_ma* from *tree*), *parent* (*parent* from *tree*), *name* (*name* from *tree*), *isLeaf* (*is_leaf* from *tree*).

An entity-relationship is shown in Figure 4.2. *FossilLocation* is linked to *FossilPoint* via *id*, the unique identifier for a fossil record. An instance of *FossilPoint* can relate to many instances of *FossilLocation*, because a fossil point has many locations at various time points. An instance of *FossilLocation* relates to one and only one fossil record in *FossilPoint*. An instance of *FossilPoint* is uniquely identified by its *id*. *FossilPoint* and *TreeNode* are linked via *pathFromRoot*. One fossil record is associated with one and only one node in *TreeNode*, however, a tree node can have zero or many fossil records. Indeed, a tree node of a high taxonomic rank has no fossil attached to it, when all fossils under this taxonomy are identified with a lower rank, thus are attached to its descendants. For instance, the root node "Eukaryota" has no fossil attachment, since, for any given fossil, we have at least identified it as an animal or a plant. Finally, an instance of *TreeNode* represents a certain node in the tree of life, therefore is uniquely identified by its *pathFromRoot*.

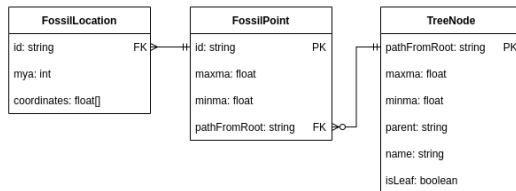


Figure 4.2: An entity relationship diagram of the database.

4.4 Back-end query processing

The back-end is responsible for processing the data queries requested by user actions, and sending the result to the front-end. How the queries are resolved is explained in this sub-section. There are two main queries to carry out: 1, given a time range and tree parameters, obtain a tree of life with fossil IDs attached to each node, and 2, obtain the locations of the fossil records given their IDs and a time point.

The first query returns a fossil-attached tree under a requested root node. The following parameters are passed to the database:

- *searchTerm*: the root node's name or *pathFromRoot*
- *maxma*: the start of the selected time period
- *minma*: the end of the selected time period

4 Technical solution

- *maxElement*: the maximum number of siblings to retrieve under each node
- *depth*: the number of tree layers to retrieve

The resulted tree should be rooted by the requested node, with a depth no greater than the parameter *depth*. The number of children under each node is no greater than the number of *maxElement*. The IDs of fossils that exist during the time period are also attached to each node. The returned tree is an array of nodes with the following fields:

- *pathFromRoot*: the unique identifier of the node
- *name*: the node's name
- *parent*: the *pathFromRoot* value of its parent
- *isLeaf*: a boolean value indicating if the node is a leaf node with respect to the returned tree
- *fossils*: the IDs of fossil records existing during the requested time period that either are attached to this node (if the node is not a leaf node) or belong to the node's subtree (if the node is a leaf node)

To retrieve the tree, we first find the requested root node by searching for either its *name* or *pathFromRoot* in *TreeNode*. The *name* is passed from user input in a search bar and the *pathFromRoot* is passed from a clicking event on the tree graph. Since the first letter of all names in *TreeNode* are capitalized, we capitalize the first letter in *name* in case it is not, before performing the search. If the *searchTerm* is a dash-separated name, then it is one of the unique names selected by users from the front-end's search bar. Then we only return the root node whose *pathFromRoot* contains the name to the right of the dash sign. For instance, if the *searchTerm* is "Corydalis-Animalia", two results can be returned when searching by the name "Corydalis". However, we only return the result whose *pathFromRoot* contains "Animalia", leaving the other "Corydalis" that is in fact under "Plantae".

The *parent* value of the root node is set to null since although it may have a parent in the complete tree of life, it acts as the root of the tree to be returned. The fossil IDs attached to the root node are also appended. We then recursively search for its children by using the *parent* field and append the children nodes to the result. The recursion terminates until reaching the requested depth or encountering nodes with no children left. When processing a certain node during the recursion, we compute the *isLeaf* value in the following way. If this node has no children in the complete tree of life (*i.e.*, this node's *isLeaf* value in database's *TreeNode* is already true), then it is certainly a leaf node in the tree under construction. However, if the iteration is on the maximum depth, then the node is a leaf node with respect to the tree under construction, whether or not it has children in the complete tree of life. When it is a leaf node, we search for the IDs of all fossils belonging to its subtree. This can be easily performed with string matching the *pathFromRoot* of the fossil records in database's *FossilPoint* using a regular expression since all desired fossils' paths begin with the *pathFromRoot* of the current node. When the node is not a leaf node, we only attach fossils that are strictly associated with this node, by exact string matching of *pathFromRoot* between this node and the fossil records. To improve efficiency, the sibling nodes are filtered before continuing to the next recursion level. A count value is computed for each sibling node, which is the number of fossil records belonging to its subtree that is present during the time period. Then the siblings are ranked in descending order, and the top *maxElement* number of siblings are kept.

The second query receives a time point of interest and an array of nodes with fossil IDs attached, which is a result of the first query. However, only the relevant fields are passed:

- *name*: the node's name
- *fossils*: the IDs of fossils attached to the node (if it is a lead node) or its subtree (if it is not a leaf node)

From *FossilLocation*, we retrieve the fossils' coordinates by their IDs and time point, and organize the result in an array. Each item has the following fields:

- *id*: the fossil record's ID
- *coordinate*: the fossil location at the time point, in the form of [latitude, longitude]
- *name*: the name to which the fossil record is associated in the current tree
- *mya*: the requested time point

4.5 Front-end development

This section introduces the design solution for the front-end, which includes a global states section that locally stores variables to which different components in the user interface have access, and the user interface where viewers interactively explore the visualized data. The user interface is in turn divided into a time control component, a map component, and a tree component.

4.5.1 The global states

The global states serve as the intermediate agents between the user interface and the back-end, and between different components of the user interface. As shown in Figure 4.1, global states include *Time parameters*, *Tree parameters* and *Current tree and fossils*.

Time parameters. Time parameters store a time period and a time point. Upon a time-related request from the user interface, certain values of time parameters are updated, and new data queries are triggered. When only the time point is changed, new fossil location data is queried using time point and the fossil IDs stored in *Current tree and fossil ID*, and new map data is loaded from static files in the back-end. When the time period is changed, a time point that represents the period is also calculated. Data on a new tree and attached fossil IDs are queried in regarding the time period, before storing the new result by updating *Current tree and fossil ID*. Due to the updated time point value, data on the map and fossil location is also queried.

Tree parameters. Tree parameters keep the record of how the tree is customized by the users. It includes the requested depth, the maximal number of children to retrieve under a given node, and the name or the path of the node to search for. When any of the parameters are changed as a result of user actions, combining with the time period information in *time parameters*, a new data query is sent to compute a customized tree of interest formed by fossils found in the time period, and the returned data containing both the tree nodes and the attached fossil IDs is stored in *Current tree and fossil ID*. Lastly, in the user interface, when a user hovers on a node in the tree graph or a fossil point on the map, the identity of the node is recorded as a parameter, for both tree and map to be aware of, which is required for the communication between the two components. The tree parameters are also responsible for the communication between map and tree components. When a user hovers the mouse on either a node in the tree graph or a fossil point attached to a certain node on the map, a parameter also records the identity (*pathFromRoot* value) of the tree node. As a result, the information on which node is being highlighted is shared with the other component, which also elicits a highlighting effect as a response.

Current tree and fossil ID. This global state variable stores information on current tree nodes with fossil IDs attached. It mainly serves two purposes. When users are interested in viewing how the same fossil points move as a result of plate movement through time, they change the time point value in *time parameters*, triggering a data query for new fossil locations with the pre-stored fossil ID values. Under this circumstance, the fossil IDs remain constant, it is thus more efficient to store them locally than re-querying upon a new time point. In addition, the colors of nodes shown in the tree component are consistent with the colors of their attached fossil points shown in the map component. As a result, upon receiving new tree and fossil ID data from the database, instead of sending directly to the tree component in the user interface for rendering, we compute the colors before storing this information in *Current tree and fossil ID*. Thereby, the color values are calculated only once and are shared by both map and tree components when they render fossil points and the tree respectively. The coloring of the

tree node is determined as follows. The current root node is in black. Its descendants are assigned with one of the five distinctive colors. The assignment initiates from the children of the root node, by visiting each node in a breath-first-search manner. A color is removed from the palette when it is assigned to a node. After all five colors are used, if the new node being visited is still a child of the current root, then the last of the five colors is assigned to this node; otherwise, we assign the same color as its parent, which must already have one of the five colors, as the parent has been visited and assigned with a color in the breath-first-search process.

4.5.2 The user interface: time control component

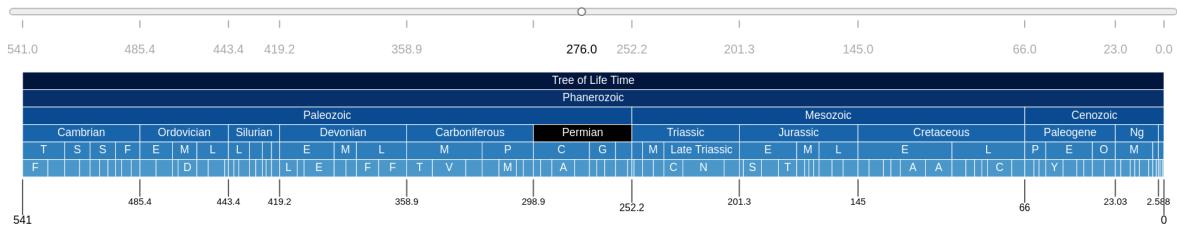
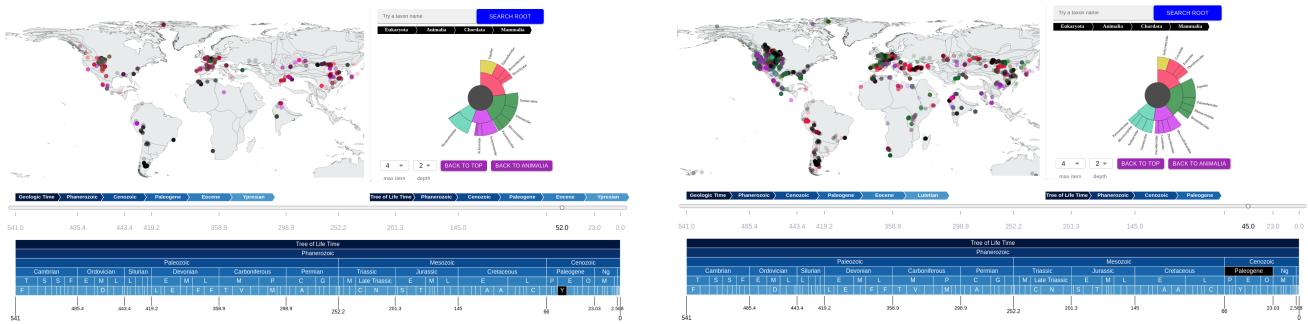


Figure 4.3: An overview of the time control component. A timescale table displays all time periods of the Phanerozoic that are organized hierarchically in five layers. When selecting a time period, *i.e.* "Permian", its cell is marked in black. Some tick values are shown below the table, marking the start and end years of all time periods on the third layer, in the unit of million years. There is a scroll bar on the top that allows users to drag, in order to view the world map and the fossil locations of a later time than the selected time period.

The time control component shown in Figure 4.3 sits at the bottom of the user interface. It allows users to control the map and tree components by either clicking in the timescale table or dragging the scroll bar. The first functionality is shown in Figure 4.4. Users can pick a time period of interest to view the tree and fossil points during that period by clicking a cell in the timetable. Only a time period on or below the third layer (Period) can be selected, in order to avoid requesting fossil data spanning hundreds of millions of years at a time, which may exceed the size of the heap memory. When a time period is selected, a time point at which the world map and the fossil locations will be queried is also calculated. This time point is intended for representing the geology during the time period. Due to the hierarchical nature of the timescale, a certain time point must belong to a time period in the fifth layer (Age). We intend to show the name of this Age, since it is of the highest detail. When an Age is selected, the time point is simply the middle value between the start and end year, rounded up to an integer (Figure 4.4(a)). Otherwise, the middle value of the time period is calculated, and we search for the Age where this calculated middle value falls into. Then, the middle integer value of that Age is used as the time point (Figure 4.4(b)). The second functionality is shown in Figure 4.5. Users can pick a time point that is later than the selected time period to view how fossil locations are shifted with the continents, by dragging the scroll bar on the top. Those time points are the middle values of all Ages later than the current time point, rounded up to an integer.

The entire Phanerozoic Eon spans over 500 million years, whereas certain time periods, especially those of lower layers, are only a hundredth of that length. Without dynamic scaling of the timetable, it is difficult to have a clear view or easy accessibility of the time periods in lower units. A solution is demonstrated in Figure 4.6. When hovering on a time period on the third (Period) layer or lower, a zooming effect takes place, where all time periods from the Period layer and below are elongated, such that the Period cell occupies 90% length of its immediate ancestor (an Era cell). As a result, labels indicating the start and end year of the shortest time periods on the Age layer can also be shown. To further highlight the time period, when hovering on it, the opacity of all other time periods that are not in the path to the common ancestor "Phanerozoic" is decreased, and a small pop-up note also shows its name as a path from the common ancestor.

4 Technical solution



(a) The time period "Phanerozoic -> Cenozoic -> Paleogene -> Eocene -> Ypresian" is selected. Since this is a time period in the lowest layer (Age), the timelines below both map and tree show the name of this time period. The map in fact shows the continents and fossil locations at the middle point of this period.

(b) The time period "Phanerozoic -> Cenozoic -> Paleogene" is selected. This time period is not in the lowest layer (it is a Period), but its middle value falls into "Phanerozoic -> Cenozoic -> Paleogene -> Eocene -> Lutetian", a time period on the lowest Age layer. Therefore, in the map component, Lutetian is the Age that represents the selected Paleogene, and its name is shown beneath the map on the left. We then take the middle value of Lutetian as the time point at which the continents were shaped and the fossils were located.

Figure 4.4: The timescale functionality of the time control panel. Users can select a period in the timescale to view the map and tree during this time period. The tree of life is always in accordance with the selected time period, whereas when the time period is not on the Age layer in the timescale, the map component picks the middle value of the Age in which the middle value of the select time period falls.

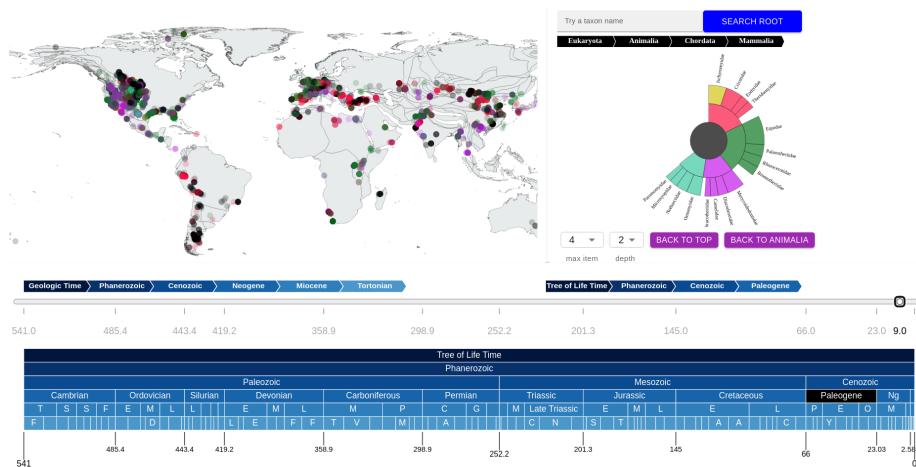


Figure 4.5: The scroll bar functionality of the time control panel. Users can drag the scroll bar to a later time point than the currently selected time period, in order to track how the continents and the fossil points move due to plate motion. From the time table and the breadcrumb beneath the tree, we can see that the tree of life and the attached fossils are still with respect to the selected "Phanerozoic -> Cenozoic -> Paleogene". However, the number of the scroll bar and the breadcrumb beneath the map imply that the scroll bar has brought the world map to a more recent time point at 9 million years ago, which is during "Phanerozoic -> Cenozoic -> Neogene -> Miocene -> Tortonian".

4 Technical solution

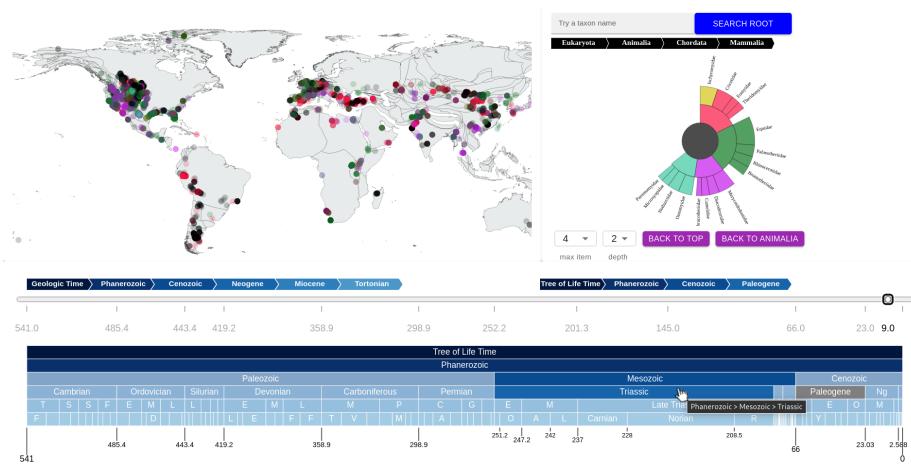


Figure 4.6: The zooming and highlighting effect of the timescale upon hovering. When users hover on a particular time period, a pop-up note with its name shows next to the pointer, and all other time periods that are not in its path to Phanerozoic are decreased in opacity. The selected "Triassic" belongs to a period on the third layer or below, as a result, a zooming effect takes place. All time periods at or below "Triassic" are proportionally elongated such that "Triassic" takes up 90% the length of its parent "Mesozoic". Due to this expansion, the tick values can now show all start and end years on the lowest layers.

4 Technical solution

4.5.3 The user interface: tree component

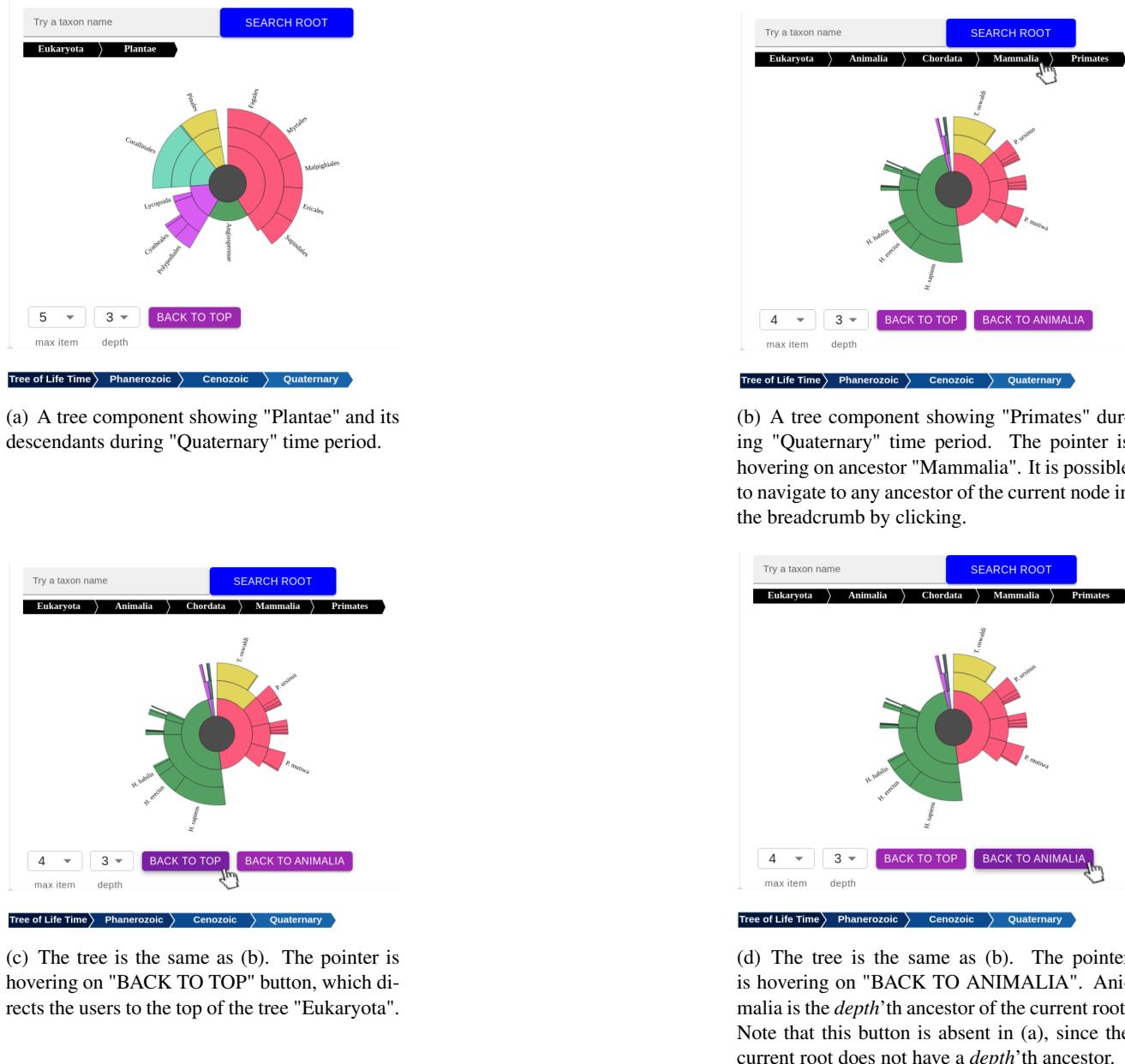


Figure 4.7: The tree component of the user interface. From top to bottom, it has a search bar for user input, a breadcrumb in black that shows the position of the current root, a sunburst graph that reflects the prevalence of each node during the time period via arc angles, a control panel for tree customization, and a breadcrumb showing the current time period.

The tree component is on the right of the user interface. It shows the current tree of interest, while providing options to customize the tree. A typical tree component is seen in Figure 4.7(a). A search bar is located on the top, where users can search for a name. Below, a line of breadcrumb indicates the location of the current root in the complete tree of life, by showing all of its ancestors in a path from the common root. Each ancestor can be visited by clicking (Figure 4.7(b)). Since the ancestor nodes are upstream of the current root, they are colored in black, to distinguish from the current root's descendants in other colors. The tree is rendered as a sunburst graph from the global state variable *Current tree and fossils* addressed in sub-section 4.5.1, which contains information

4 Technical solution

on a node's name, color and fossils.

To demonstrate the prevalence of a certain taxonomy during the selected time period, the angle of each node's arc reflects the number of fossils attached to its subtree. The current node under inspection is the root of the tree being displayed, as a consequence, it is a full circle, since it includes all fossils in its subtree. The arc angles of sibling nodes are proportional to the ratio between the number of fossils in their respective subtree, and the number of fossils included in their parent's subtree. To calculate the number of fossils under a given node's subtree, we recursively sum up the fossil counts from leaf nodes to the current root node, by the number of fossils attached to each node. Note that when performing the sum up, if the node is a leaf node, we sum up by all fossils under its subtree, however, when the node is not a leaf node, we sum up by the fossils that are attached to the node. This is because if we still sum up by the fossils belonging to its subtree, we are double-counting the fossils, as each of its children is already summed up by the fossils under their respective subtree. *Current tree and fossils* conveniently stores fossils attached to a node (for non-leaf nodes), or fossils associated with the node's subtree (for leaf nodes). Therefore, when performing the sum up at the front-end, no further processing is needed to distinguish if we are counting the fossils with respect to only a node, or its subtree.

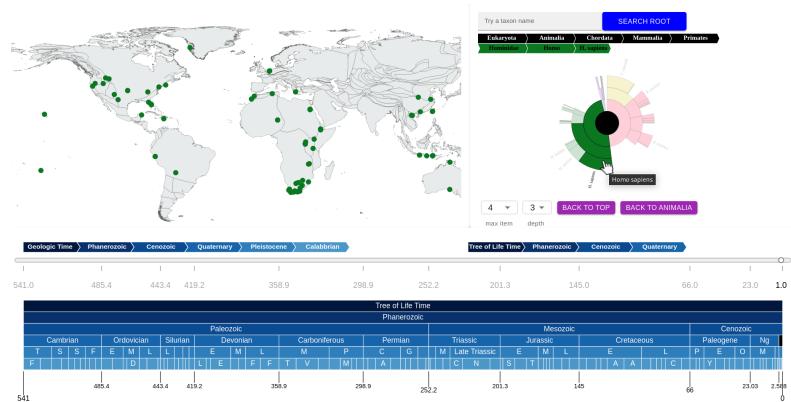
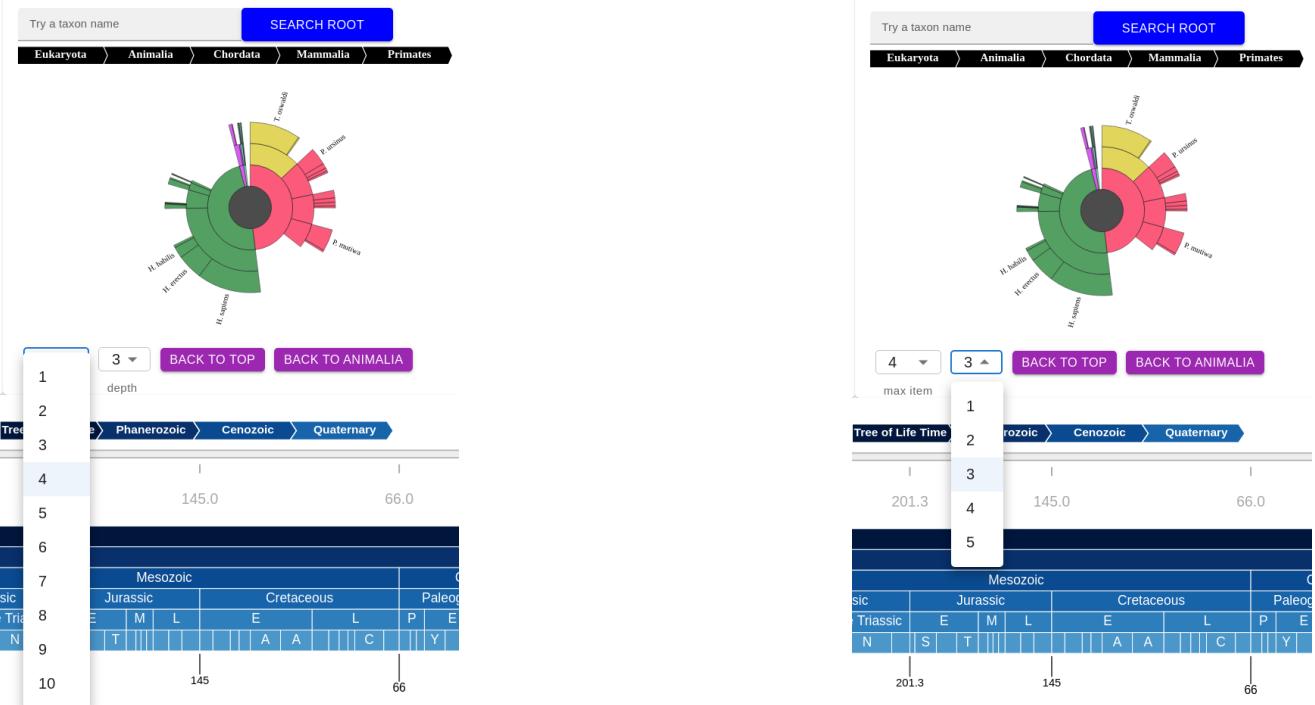


Figure 4.8: The tree's visual effect upon user hovering on a node. The tree is the same as in Figure 4.7(b). When a user hovers on "Homo sapiens", a pop-up note shows its name, and an additional breadcrumb in the color that is consistent with that of the tree appears, indicating the position of the hovered node in the complete tree. The nodes that are in its path to the current root are in full opacity, whereas all other nodes have decreased opacity. Most importantly, the map on the left side only shows the fossils directly attached to the hovered node, during the time when the user is hovering on it. Finally, it is possible to click on the node to navigate to the tree rooted by it.

To highlight a certain node of interest, we provide the following visual aids (Figure 4.8). First, a second line of breadcrumb indicates its position from the current root node. Second, similar to the time control component highlighting effect, the opacity of other nodes is decreased, and a pop-up note informs the user of the node's name. When a node's name is too long or a node's arc is too narrow to display on the tree, the pop-up note and the dynamic breadcrumb provide an opportunity to view this name upon hovering. Finally, the hovering effect is communicated to the map component, such that only the fossil points attached to the hovered node are shown and in full opacity, to inform the users where their taxonomy of interest inhabited in the world during the selected time period.

To enable users to customize and navigate the tree, we designed a small control panel below the tree graph. If the current node is not the common root "Eukaryota", the *Back to top* button directs the user to it (Figure 4.7(c)). Clicking on the "Eukaryota" in the breadcrumb above the tree graph provides the same functionality. If the distance from the current root to "Eukaryota" is greater than the current *depth*, a second button redirects the users to upstream of the current node that is its *depth*'th ancestor. As an example, in Figure 4.7(d), the *depth* value is 3, and current root is "Primates". From the breadcrumb in black, we can see that the third ancestor of "Primates"

4 Technical solution



(a) Upon clicking on *max item*, a drop down list appears, prompting the users to select a number from 1 to 10, which is the maximal number of siblings to shown under a given node.

(b) Upon clicking on *depth*, a drop down list allows users to customize the number of depth to show with respect to the current root, which is a number from 1 to 5. This number also determines which upstream ancestor to navigate to in the second "BACK TO" button.

Figure 4.9: The drop down lists in the tree's control panel.

is "Animalia", hence the button in Figure 4.7(d) that provides a quick way to navigate to it. In Figure 4.7(a), however, the current root is "Plantae", and it does not have a third ancestor, since its first ancestor is already the common root "Eukaryota", hence there is only a "BACK TO TOP" button. To the left of the buttons, the *max item* adjusts the maximal number of elements to display under each node (Figure 4.9(a)), and the depth adjusts the tree's depth (Figure 4.9(b)).

Finally, when the name that the user is searching is not unique, an auto-complete function asks the user to select one of the nodes to navigate to (Figure 4.10). Note that the unique names in the drop-down list are the result from subsection 4.2.3: the name to the right of the dashed line is the first distinctive ancestor between the two nodes.

4 Technical solution

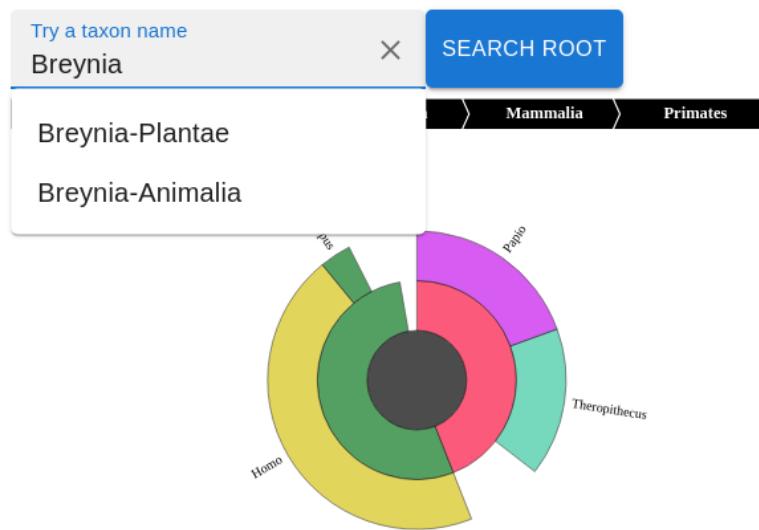


Figure 4.10: The auto-complete function prompts the users to select a node to view when there are two tree nodes sharing the same search name.

4.5.4 The user interface: map component

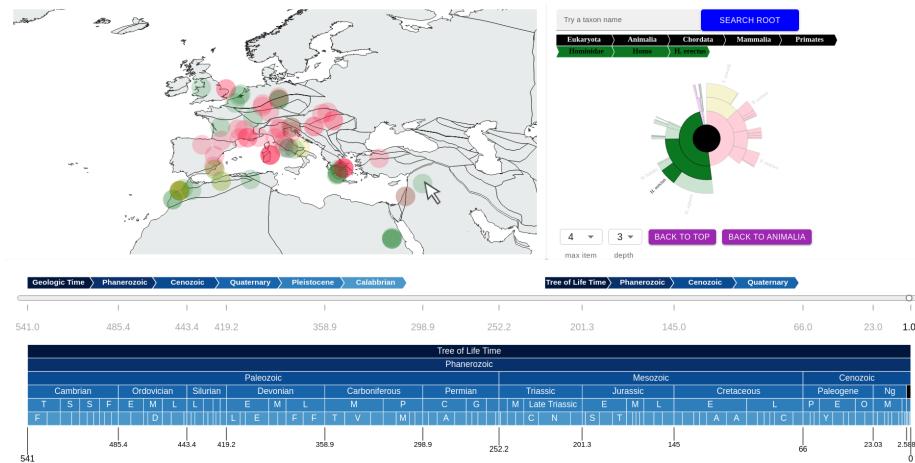


Figure 4.11: The map component of the user interface. The region around nowadays Europe is zoomed in, and the mouse is hovering on a fossil point around nowadays Middle East region. As a result, the node this fossil is attached to, which is "Homo sapien", is highlighted in the tree component, such that the node is of higher opacity, and a second breadcrumb indicates its name and path from the root.

The map component is on the left of the user interface. It shows the world map with fossil points at the select time point. The geological data on coastlines are provided from static files in the back-end, and the fossil coordinates are rendered directly from database's *FossilLocation*, whereas their pre-computed colors are loaded from front-end in *Current tree and fossils* discussed in 4.5.1. An overview of the map component is shown in Figure 4.11. The zooming and panning functionality enables users to view regions of interest in greater detail, and when hovering on a fossil point, the tree component shows which node it attaches to, by highlighting the node and providing a second breadcrumb that indicates its position, in a similar way as hovering on a node in the tree (Figure 4.8). Lastly, often fossil points are of very close proximity, or even have identical locations, leading to the fact that the points can easily be on top of each other. As a solution, we decreased the opacity of the fossil points in order to reflect all fossil records' presence — a point of higher visibility implies that there are multiple records found at or close to the location.

5 Implementation

This chapter explains how the technical solutions introduced in chapter 4 is implemented to produce the web application. First, the stacks used in both data processing stage and the web development stage are listed in section 5.1. Then, section 5.2 describes the code structure. Next, section 5.3 addresses how the data processing is implemented. Finally, section 5.4 explains the steps to realize the web application's structure shown in Figure 4.1.

5.1 Dependencies

At the data processing stage, Python3 and Pandas library are used to process the raw PBDB data to generate data on both tree and fossil information. GPlates is used to transform modern-day coordinates of both fossil records and global coastlines to ancient time points. At the web development stage, MongoDB serves as the database to store the tree and fossil data. Mongoose is used as the object document modeling (ODM) library for MongoDB and Node.js, as the back-end is written in JavaScript. GraphQL is the query language used for requesting data. The Apollo graph platform connects the front-end and the back-end, and it consists of Apollo Server, which communicates the GraphQL queries to the back-end, and Apollo Client, which manages GraphQL queries induced from actions at the front-end user interface. Axios is used to load world maps in GeoJSON by accessing a file's URL. Finally, React is the framework in which the front-end is implemented, and MaterialUI is used to style the website. d3.js is the library used for generating and manipulating the visualized data that is SVG-based. Finally, Webpack is used as the module bundler.

5.2 Code structure

The code for data processing, back-end and front-end are structured as follows:

- **app/**: includes the code for building the web application.
 - **server/src/**: includes the back-end code to run the server (*/index.js*), to define the Mongoose schema (*/models.js*), to define the GraphQL schema (*/schema.graphql*), and to upload the processed datasets to the database (*/insertBiodiversityData.js*).
 - * **resolvers/**: includes the code for resolving the queries from the front-end, *i.e.*, retrieving and processed desired data from MongoDB before returning the result.
 - **src/**: includes the code for front-end.
 - * **components/**: includes the code for different components and sub-components that the front-end framework is assembled from. The user interface consists of time control, tree and map components, and there is also a component to store global states and a component for data fetching.
- **data_processing**: includes the code for processing raw data to generate the tree and fossil data, the code to further organize fossil data to prepare it for coordinate transformation through time, and the code to plot the graphs in chapter 6.

5.3 Data processing

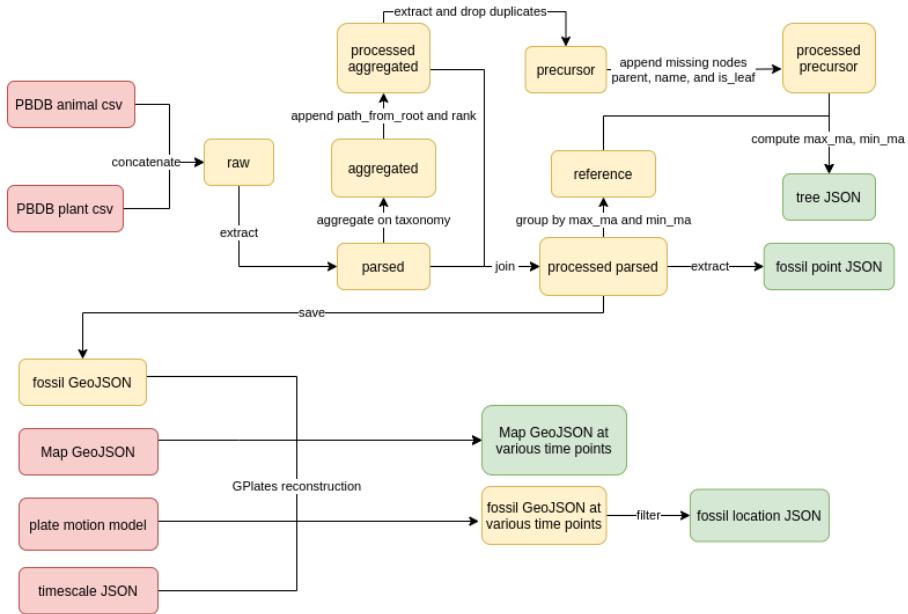


Figure 5.1: The data processing workflow. The unprocessed data sources are marked red; intermediate products are marked yellow; final products are marked in green. In short, The data on tree of life and fossil point is prepared from PBDB data on animal and plant fossils. The geospatial data on both map and fossil locations at different time points are prepared from GeoJSON data (unprocessed or intermediate files) and a published plate motion model using GPlates software.

Prior to developing the web application, we must obtain the necessary data to visualize, by implementing the strategies in section 4.2. Pandas, a standard Python library specialized in data science tasks is used for this purpose. The workflow is summarized in Figure 5.1. First, we concatenate the PBDB [PM16] data files for animal and plant fossil records into one file, the raw data. Then, we parsed the relevant attributes (listed in 4.2.1) to form the parsed data. The rows in parsed data with identical values on all taxonomic ranks are aggregated, such each row in aggregated data has unique taxonomy information. Then, the *path_from_root* and *rank* values for each row can be calculated. This processed aggregated data with appended information now represents all tree of life nodes with at least one fossil record attached. It is further processed in the following two directions. On one hand, it is joined with the parsed data, such that every fossil record in parsed data would also have the appended information. We then extract the *path_from_root*, *occurrence_no*, *max_ma* and *min_ma* from the parsed data to generate the *fossil_point* dataset, which is saved in a JSON file for uploading to database. On the other hand, we extract only the newly added *path_from_root* and *rank* in the processed aggregated data, before saving the unique rows in a precursor dataset in a CSV file. This precursor dataset is responsible for the preparation of the *tree* dataset, as each of its rows is a node in the tree of life. Note that the precursor dataset is still missing the tree nodes that have no fossil records attached. We first iterate it to include all missing nodes in the tree of life as mentioned in sub-section 4.2.3. Then, we compute the *parent*, *name* and *is_leaf* information for each row. The precursor data is now processed. Finally, we append *max_ma* and *min_ma* values for the rows. To do that, we use the processed parsed data again. As a reminder, a row in the processed parsed data represents an individual fossil record. Therefore, we group the rows with identical *path_from_root* (*i.e.*, the fossils attached to the same tree node) by 1, the maximum of all *max_ma* and 2, the minimum of all *min_ma*. We iterate through this new reference data, and for each tree node in the processed precursor data, we visit all its ancestors to update their *max_ma* and *min_ma*, as explained in 4.2.3. The dataset on the tree is ready to be saved in JSON and uploaded to the database.

Concerning geospatial data processing, GPlates is used to reconstruct all modern coordinates to ancient ones. We export the processed parsed data that contains modern coordinates of each fossil record into GeoJSON format, before loading the data into the software. In addition, we also load modern plate boundaries data as GeoJSON and the simulation model that defines the kinetics of plate motion since the past 410 million years, which are both available in the work by Matthew and colleagues [MMZ⁺16]. The timescale data is provided by PBDB [PM16]. As a reminder, it is a hierarchical structure with five layers, and we trace all modern locations to a particular fifth layer (Age) time period. As a consequence, all ancient time points we are reconstructing are essentially the middle integer values of all Ages that are no earlier than 410 million years. GPlates assigns all fossil points to the plate that they belong to, and reconstructs those locations together with plate boundaries, using the plate motion kinetics model. All results are exported in GeoJSON. The plate boundaries files describe the world map at a particular time point. We store them as static GeoJSON files that can be directly rendered at the front-end upon requesting a certain time point. With respect to the reconstructed fossil location data, further filtering is executed. For a given fossil record, it is in fact not necessary to keep all its ancient locations. At a time period before the earliest estimated existence of a fossil, it will never appear on the map. If a time period spans longer, then it can cover more fossils. The grossest time periods that users are capable of selecting are on layer three (Period). If a time point is earlier than the start of the Period to which the fossil's *max_ma* belongs, then we can filter out its location at this time point. After filtering, we extract each fossil's ID (*occurrence_no*), coordinates, and time point in million years and store them in the *fossil_location* dataset in JSON files for uploading to the database. We use JavaScript for geospatial data processing due to its convenience when working with JSON and GeoJSON files.

5.4 Web development

5.4.1 Back-end

The back-end provides a database to store the data prepared in section 5.3, and an API to handle the data traffic with the front-end. Although a relational database would also suffice for this project, MongoDB is used as the database of choice. It is relatively convenient to represent our prepared data in JSON with this document-oriented database system. More importantly, as a NoSQL database, the lack of data normalization enables higher horizontal scalability (*i.e.*, no rigid schema). This gives a faster iteration time in the early stage of development which is highly appreciated. A lot of trials and errors are needed to decide the appropriate fields to include for both tree and fossil data. The *tree*, *fossil_point* and *fossil_location* JSON files from section 5.3 are uploaded to MongoDB database into collections *TreeNode*, *FossilPoint* and *FossilLocation* respectively, as described in 4.3.

When building the API between the data requesting front-end and the database, GraphQL is chosen as the data query language. A traditional REST API returns fixed data structures from the server, as a result, it may need to access multiple endpoints from the server, leading to over-fetching (downloading data attributes that are not actually needed) and under-fetching (accessing another endpoint since some attributes are not included in the returned result). Therefore, the APIs are designed according to the needs of the UI, such that a particular request corresponds to one endpoint from the server. However, a change of the front-end's data needs requires the corresponding back-end adjustment. GraphQL allows more rapid iterations during early development. With GraphQL, a single data query that declares all the desired fields of typed data is sent to the back-end, which returns a JSON object with only the required information. A data query is processed by a JavaScript resolver function that fetches data modeled by the ODM mongoose. The algorithms of the resolver functions are described in section 4.4. Now, any change in the data needs would only require re-writing the relevant data queries and changing the resolver functions, which is relatively more productive. There are two possible queries, *getTreeWithFossils*, which retrieves the tree with fossil IDs attached to each node, and *getFossilLocations*, which retrieves the fossil coordinates data at a requested time point. The GraphQL system is implemented by the Apollo platform. It provides the Apollo server, which connects the data from MongoDB, and Apollo Client, which sends front-end data queries and returns the data to UI. An overview of the API structure is shown in Figure 5.2. Finally, the world map at a certain time point is stored as a GeoJSON file accessed by its URL, and it can be loaded by sending an HTTP

5 Implementation

request using Axios.

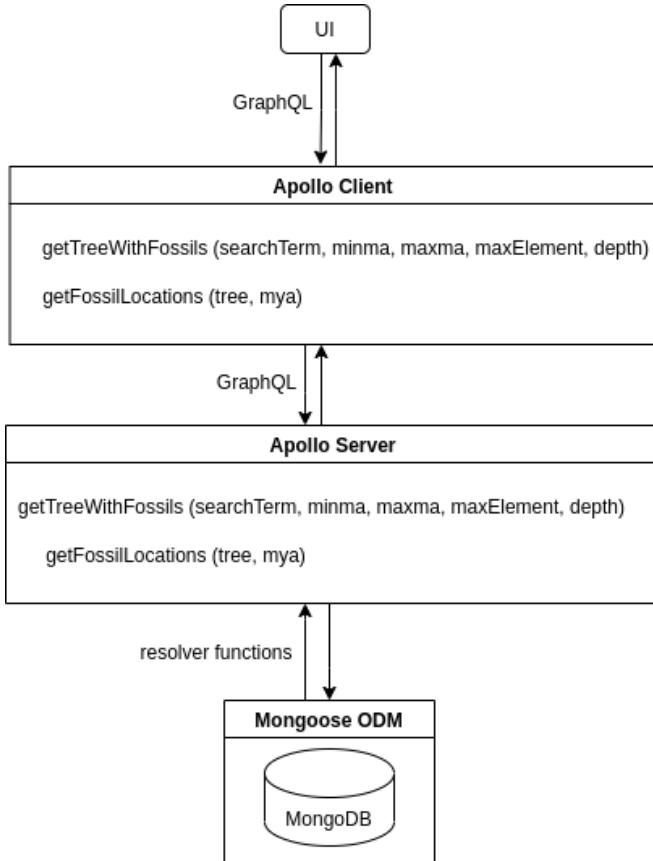


Figure 5.2: The API implementation. In short, Apollo Client handles GraphQL queries triggered by user action from UI. The Apollo Server then receives the GraphQL queries, which are in turn processed by resolver functions that retrieve data from Mongoose modeled data that is stored in MongoDB. The data is finally returned from the database all the way to the UI.

5.4.2 Front-end

The front-end design from section 4.5 is realized in a React framework. Conceptually, the front-end consists of the global states and the user interface, as seen in Figure 4.1 in chapter 4. The implementation of global states are listed in Figure 5.3. All global states can be classified into three categories. For time parameters, *mya* records the time point at which to show the world map and fossil locations, while *maxma* and *minma* together define the time period to retrieve the fossils and to form the tree of life. For tree parameters, *searchTerm* is the *name* (when users request a node by typing in search bar) or *pathFromRoot* (when users request a node by clicking) of the root node of the tree of interest; *depth* and *maxElement* are the number of layers and the max number of siblings under each node when retrieving the tree of life. Finally, *mapFocusNode* and *treeFocusNode* are the tree node identified by *pathFromRoot* on which a user is hovering in the map (users can highlight a tree node by hovering on a fossil point that is attached to it) and tree component, respectively. They are implemented to communicate the highlighting effect in between map and tree components (Figure 4.8 and Figure 4.11). The variable *currentTree* stores all nodes of the current tree of life in an array of objects. Each object contains the information on a node's name, path, parent, color, and whether or not the node is a leaf node. The IDs of the fossils that are attached to the node (for a non-leaf node) or associated with its subtree (for a leaf node) are also stored in an array in the object. In React, variables can only be passed down from parents to children via props. Since our components are eventually assembled in a tree-like structure (Figure 5.4), global states need to be passed down each layer

5 Implementation

to reach the component that actually needs them, leading to cumbersome code. Therefore, all global states are implemented as state variables in a React component *GlobalStateContext*. A certain component gains access to the desired global states by directly importing them from *GlobalStateContext* using the React hook *useContext()*.

Time parameters	Current Tree and fossil ID	Tree parameters
mya: Int maxma: Float minma: Float	currentTree: {name: String, pathFromRoot: String, parent: String, isLeaf: Boolean, color: String, fossils: String []} []	searchTerm: String depth: Int maxElement: Int mapFocusNode: String treeFocusNode: String

Figure 5.3: A list of all global states. The front-end's global states falls into three categories: time parameters, tree parameters and current tree and fossil ID. While time parameters and tree parameters include several variables of primitive types, current tree and fossil ID only has one variable, *currentTree*, that is an array of complex objects.

Next, we introduce how the components of the user interface are assembled and how they could interact. An overview of the user interface is shown in Figure 5.4. The user interface is presented by *App*, which has one *Header*, one *Footer* and one *TimeControl*. The *TimeControl* has *mya*, which is needed to show the position of the current time point in the scroll bar (Figure 4.3). By clicking on a selectable time period, both time point and the *maxma* (the start of the time period) and *minma* (the end of the time period) will be updated, triggering both a *getTreeWithFossils* query for the tree, and a *getFossilLocations* query for the map. Since both map and tree requires data traffic from the back-end, they both inherit from a more general *DataFetcher* component for higher code re-usability. *DataFetcher* is responsible for sending queries via *useQuery()* that passes the appropriate *queryVariables* depending on its *queryName* (*i.e.*, whether it is *getTreeWithFossils* or *getFossilLocations*). The exact *queryVariables* for each query is addressed in Figure 5.2. The *data* in *DataFetcher* hosts the returned data from the database, and is passed down to its children for further processing and rendering.

Tree receives the tree node data, and calculates the colors for each node, before updating the result in *CurrentTree*. *TreeGraph* renders the tree stored in *CurrentTree* in a sunburst graph. The *mapFocusNode* alerts the tree graph to provide a highlight effect for the node whose fossil a user is hovering on in the map. In the meantime, when a user hovers on a node in the tree graph, it can alert the map component to highlight its fossil point by *setTreeFocusNode()*. When users are intended to navigate into a node, by clicking on it from the graph or the breadcrumb above it (Figure 4.7), *setSearchTerm()* is called, which in turn triggers new queries from the database.

TreeSearchName renders the search bar above the tree graph (Figure 4.7). It keeps notes of what a user is typing in *searchNameBuffer*, and upon clicking on the search button, *setSearchTerm()* updates the name of the node of interest, which also triggers new queries. *TreeGraphCustomize* renders the small control panel below the tree graph (Figure 4.7). It shows the current *maxElement* and *depth*, while updating these values when users change them in the drop down lists (Figure 4.9). When users are clicking either of the "Back to" buttons, *setSearchTerm()* is responsible for updating the global state variable *searchTerm* with the *pathFromRoot* value of the clicked node and re-directing the users to a different tree.

Lastly, *map* renders the world map in GeoJSON, and the fossil locations returned to the *data* inherited from its parent *DataFetcher*. It acknowledges the colors of the fossil points from *CurrentTree*, and the fossil points to highlight (as a response to user hovering in the tree component) from *treeFocusNode*. *setMapData()* uses Axios to load the correct GeoJSON map data according to the current *mya*, before storing it in *mapData* for rendering. When a user hovers on a fossil point, its attached node is updated by *setMapFocusNode()* to communicate this information to *TreeGraph*.

5 Implementation

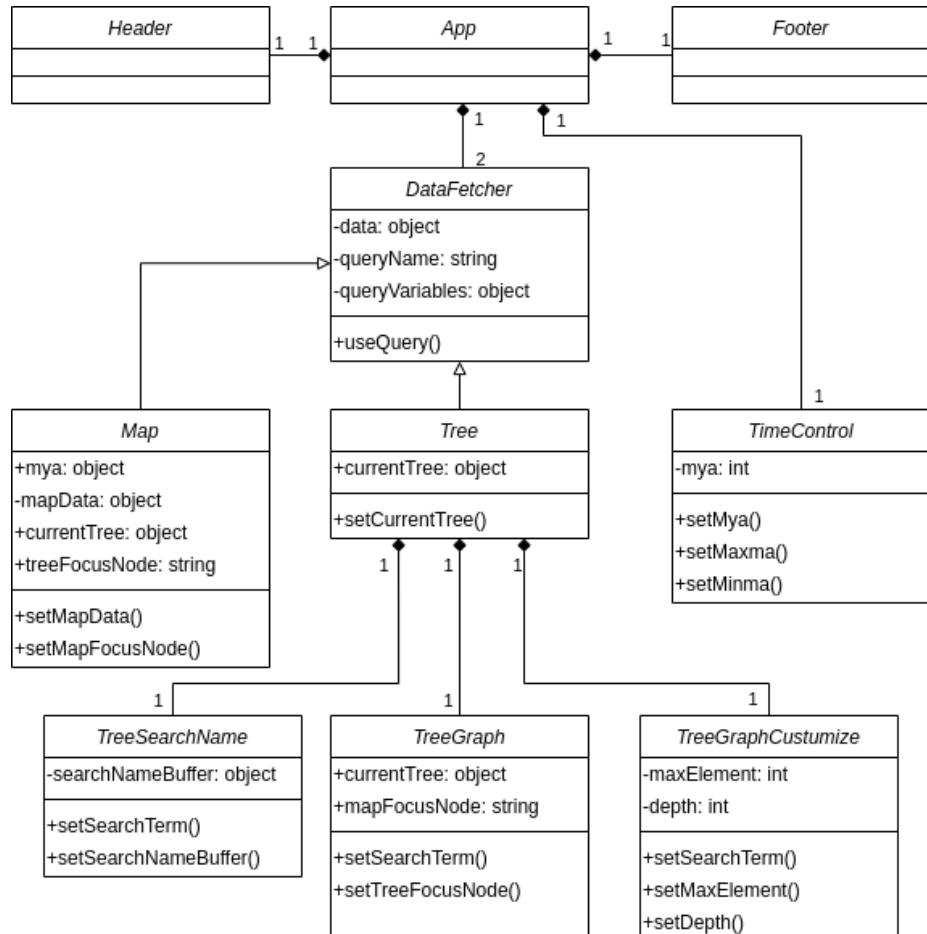


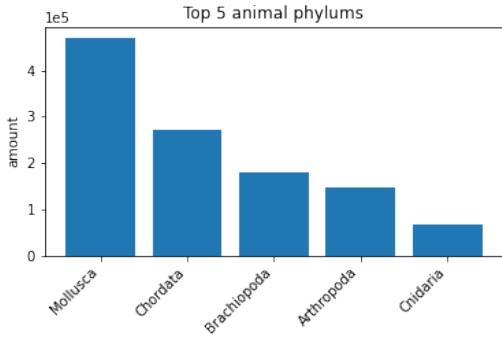
Figure 5.4: The implementation of the user interface by assembling different components. The user interface is the `App` components, which is assembled by a `Header`, a `Footer`, a `TimeControl` as well as `Map` and `tree`, which are both children of a more general `DataFetcher` component. The `Tree` is further broken down into three sub-components. For each component, the properties that belong to global states are marked with a "+", since they are accessible by all components.

6 Results

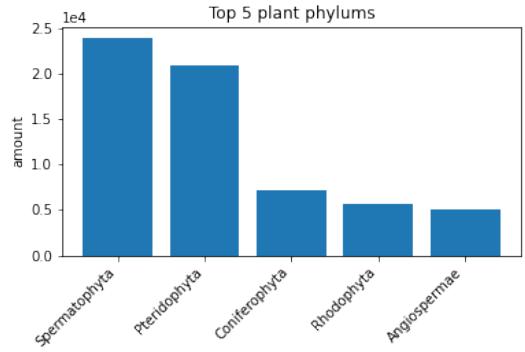
In this chapter, the results are organized into two sections. Section 6.1 shows some characteristics of the tree and fossil data revealed after data processing. Section 6.2 uses some case studies to show how the web application may serve as a useful tool to explore biodiversity through time. Due to the visualization nature of the product, the results in the second section are demonstrated as screenshots.

6.1 Data processing

There are in total 1346120 fossil records, of which 1240202 (92.13%) are animals and merely 105918 (7.87%) plants. Figure 6.1 shows a breakdown of the top five animal and plant phyla with the most number of fossils identified under them. Over the entire evolutionary history, in regard to animals (Figure 6.1(a)), Mollusca (soft-bodied shell-covered invertebrates) has the highest number of fossils found, followed by Chordata (the phylum that mammals belong to), Brachiopoda (marine invertebrates with bivalve shells), Arthropoda (invertebrates with an exoskeleton and segmented body, such as scorpions) and Cnidaria (animals possessing cnidocytes, such as jellyfish and corals). In plants (Figure 6.1(b)), Spermatophyta (seed plants) has the most fossils under it, followed by Pteridophyta (spore dispersing vascular plants), Coniferophyta (cone-bearing plants such as pine trees), Rhodophyta (red algae) and Angiospermae (flowering plants).



(a) The animal fossils identified under each phylum is ranked by amount in descending order and the top 5 phyla are displayed.

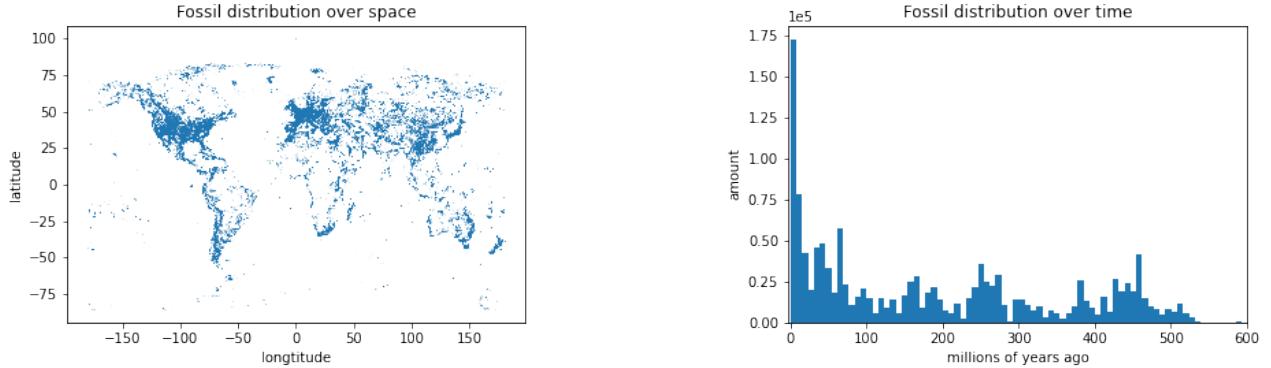


(b) Similar to animal fossils, the top 5 plant phyla with the most amount of fossils identified under.

Figure 6.1: The top 5 animal and plant phylum with most fossils.

The fossil distribution in time and space is shown in Figure 6.2. Fossil records are not equally distributed over the planet (Figure 6.2(a)). All fossils form a clear shape of the continents, indicating that proportion-wise, very few fossils are found in oceans. Furthermore, several fossil-rich areas exist, including the vast majority of North America, entire Western Europe, and most of East Asia, as well as the coastline regions of South America, Africa, and Australia. This may be due to easier fossil accessibility in those regions. The result implies a strong bias of biodiversity information favoring such regions.

6 Results



(a) Space distribution of fossils. The longitude and latitude values of every fossil is shown on the x-axis and y-axes, forming shapes of the continents.

(b) Time distribution of fossils during the past 600 million years. The middle value between *max_ma* and *min_ma* is computed to represent a fossil's age, and only data in the past 600 million years is shown.

Figure 6.2: Fossil distribution in time and space.

The oldest fossils in the dataset date back to 1000 million years ago. However, half of the fossils discovered are estimated to be at most 160 million years old. Each fossil is dated to an age range, with *max_ma* value indicating the maximal possible age and *min_ma* indicating the minimal. Their middle values are calculated and plotted in Figure 6.2(b). An obvious bias favoring very recent (less than 10 million years ago) can be seen. In more ancient times, the fossil amount distribution tends to fluctuate. Although fossil amount is more of a measure in biomass rather than biodiversity, this fluctuation is also expected, due to the numerous mass extinction and explosion events throughout evolutionary history. The age range distribution is demonstrated in Figure 6.3. There are only 444 fossil records with identical *max_ma* and *min_ma* (*i.e.*, dated to a precise time point). The average age range of a fossil record is 6.80 ± 7.59 (mean \pm sd) million years.

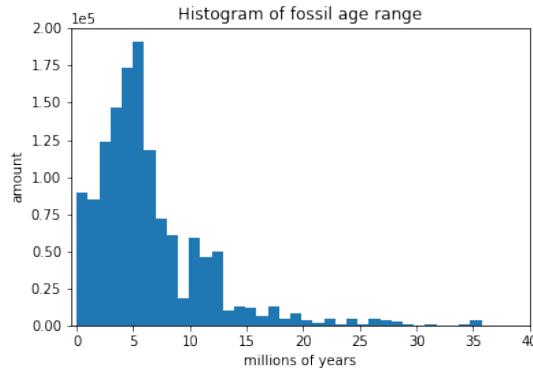


Figure 6.3: The fossil amount distribution over different ranges of age. The range of age is the difference between a fossil record's *max_ma* and *min_ma* values.

All fossils are organized into a tree of life with 176369 nodes, of which 147315 have at least one fossil attached. The fossil-free nodes are branch nodes that serve as the ancestors of lower-ranking nodes with attached fossils. The tree has one domain (the root "Eukaryota"), two kingdoms ("Animalia" and "Plantae"), 60 phyla, 159 classes, 1061 orders, 7305 families, 47828 genera, and 119701 species. The top five nodes with most fossils directly attached are shown in Figure 6.4. The class "Gastropoda" is commonly known as snails and slugs, and the class "Saurischia" is a major category of dinosaurs. It can be seen that none of the nodes are on the species level, and most of the nodes are animals, as animal fossils, in general, outnumber plant fossils.

6 Results

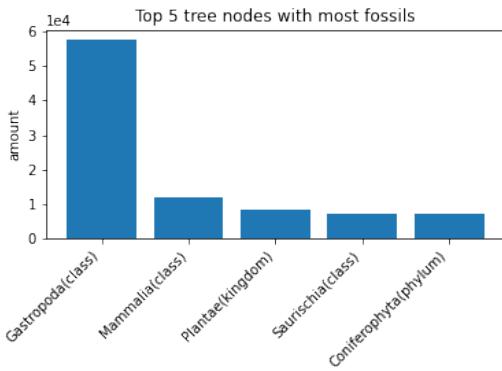
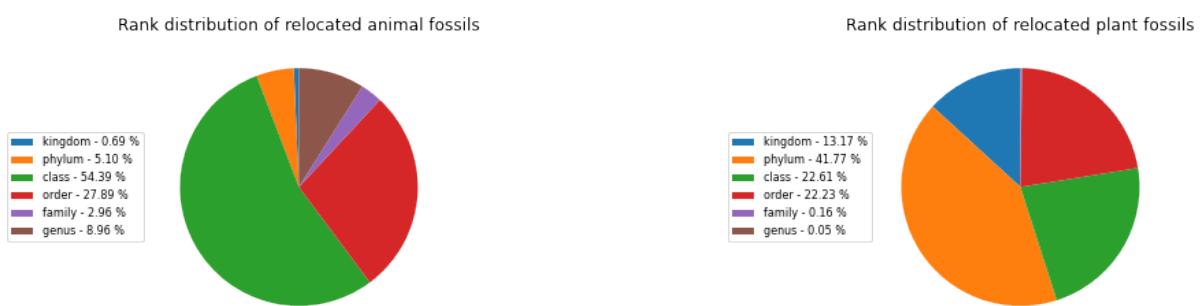


Figure 6.4: The top five tree nodes with most amount of fossils attached to. Note that this is different than in Figure 6.1, which includes the fossils that are under the subtree of a certain phylum node.

As a reminder, when producing the tree of life using the method discussed in section 4.2.2, each node is uniquely identified by a *path_from_root* string with no missing names in between. However, missing names in taxonomic ranks are very common in the raw data. As a consequence, the identified ranks of many fossils are "relocated" to a higher one, namely, the one before the first missing name. For instance, if a plant fossil is missing its phylum name but does have names on lower ranks, then it is still relocated to the kingdom node of "Plantae". Therefore, this fossil record loses some detailed taxonomic information. There are a total of 209257 (16.87%) animals and 58874 (55.58%) plants that suffer from this circumstance. Therefore, missing taxonomic information is a severe issue in plant records. Figure 6.5 shows that among all relocated animal fossils, most of them are settled on the class (54.39%) and order (27.89%) level, whereas most of the plant relocated fossils settle on the phylum (41.77%), class (22.61%), and order (22.23%) level.



(a) The distribution of taxonomic ranks on which relocated animal fossils are finally identified. When a fossil has taxonomic information on lower ranks but lacking names on at least one of the higher ranks, it is relocated to the rank before the first missing name.

(b) Similar to animal fossil, the distribution of ranks on relocated plant fossils.

Figure 6.5: The rank distribution in fossil relocation.

6.2 Data visualization

The use scenarios of the web application can be determined by a user's own need and curiosity. However, the combination of map and tree mainly enables users to observe the wax and wane of a selected taxonomy across time, as well as the change in dominance and spatial distribution of the organisms under this taxonomy. An example is seen in the development and sudden disappearance of Avetheropoda over a span of 150 million years. Avetheropoda ("bird theropods") is an order that covers dinosaurs that are relatively closely related to modern birds, as well as predatory dinosaurs, including the famous *T.rex*. To begin with, we can type "Avetheropoda" in the search bar, and collect all fossils during the Early Jurassic, as seen in the breadcrumb time indicator below the tree graph in Figure 6.6(a). During this time period, there are only a few fossils covering two families. Forwarding to Middle Jurassic, Avetheropoda has diversified greatly, as seen in the tree graph of Figure 6.6(b). Their fossils also tend to be more well spread around the world. Late Jurassic sees an even greater amount of Avetheropoda fossils (Figure 6.6(c)). Its biodiversity in the Late Jurassic is characterized by a dominant genus (Figure 6.6(c)). This is distinctive from Middle Jurassic, during which no particular families or genera greatly outnumber others. A visual inspection of the arc angles of the tree reveals that this dominant family alone accounts for around one-third of all Avetheropoda fossils, and most of the fossils in this family belong to only one genus. When hovering on the node, we can see from the dynamic breadcrumb above the tree graph and the pop-up note that this is the genus *Allosaurus* from the family Allosauridae, whose fossils are found mostly in the Western United States (Figure 6.6(d)).

Moving on to the Cretaceous, Avetheropoda is experiencing a prosperous period. In both Early (Figure 6.7(a)) and Late Cretaceous (Figure 6.7(b)), Avetheropoda is widely spread all over the world, with even more fossils found compared to the epochs of Jurassic. This continues to Danian, the first Age (a layer-five time period) after Cretaceous (Figure 6.7(c)). *Tyrannosaurus*, the genus of the iconic dinosaur *T. rex*, is highlighted by the red arrows in Figure 6.7(b) and Figure 6.7(c). The relatively wide arc angle implies the prevalence of *Tyrannosaurus* during these times. The scroll bar at the bottom of Figure 6.7(c) shows that the middle value of Danian is 64 million years ago. However, when entering the next Age, Selandian, whose middle value is 60 million years ago, all Avetheropoda suddenly disappeared, except for one single fossil found on an island in the nowadays south Pacific ocean (Figure 6.7(d)). This abrupt change brought an end to Avetheropoda and also other dinosaurs, and it reflects the well-known event of a mountain-sized asteroid with a diameter of 12-kilometer hitting Earth 66 million years ago. As a result, the Age Selandian that is immediately after this incident sees few dinosaur fossils.

A second case study inspects Dictyopteridiales, an order of plant present in the Permian and Triassic, that is characterized by its tongue-shaped leaves. In Figure 6.8(a), we access Dictyopteridiales fossils during the entire Permian period. A genus *Glossopteris* that is marked by green points on the map can be seen dominating the southern hemisphere, while other genera are far separated from it and are inhabiting in the northern hemisphere (Figure 6.8(a)). We then keep the tree of life's time in Permian (as seen in the time tables in figures of 6.8). Meanwhile, we use the scroll bar above the timetable to view how the same Permian fossils shift their positions through time, at an interval of around 100 million years (Figure 6.8(b), Figure 6.8(c) and Figure 6.8(d)). It can be easily seen that as the supercontinent (termed "Gondwana") drifts apart since Permian, the *Glossopteris* fossils marked in green are gradually separated into different continents of the modern world. Although they are found in South America, the southern part of Africa, Australia, and India in the modern world (Figure 6.8(d)), they were in fact joined together during the time they were alive (Figure 6.8(a)). Indeed, the fossil distribution of *Glossopteris* is an example of how fossils could provide evidence supporting the continental drift theory [Sch70].

6 Results

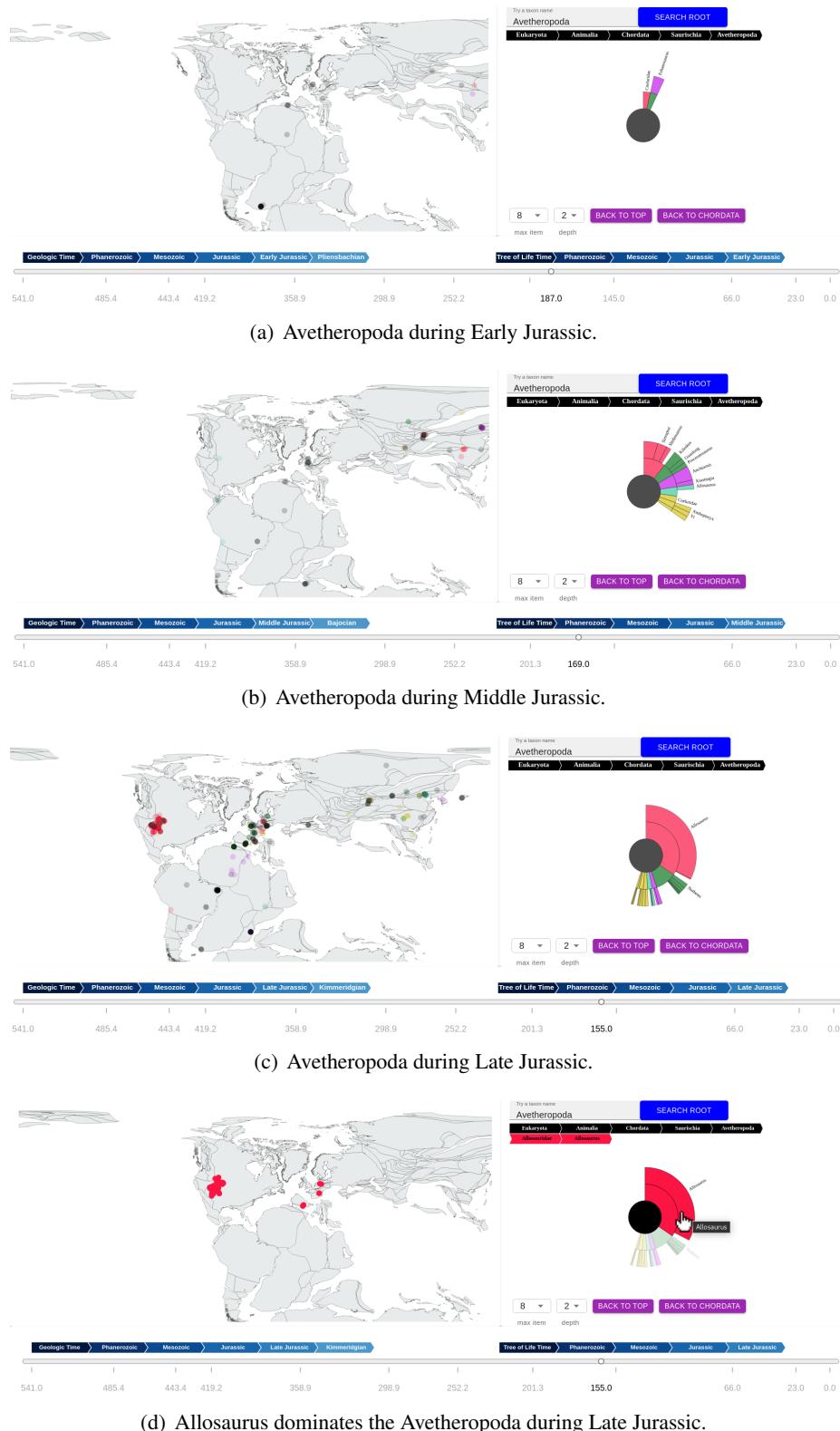
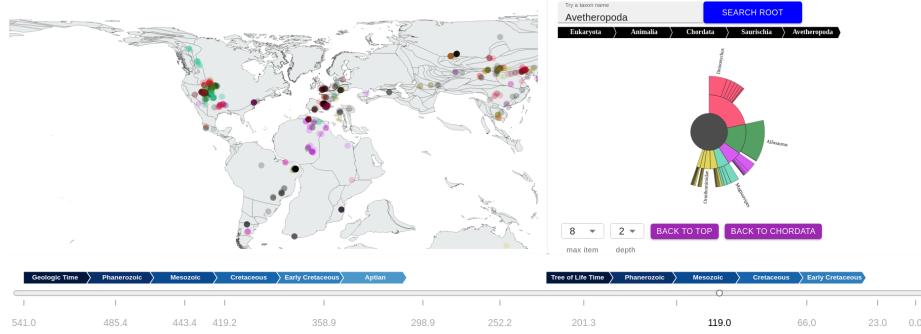
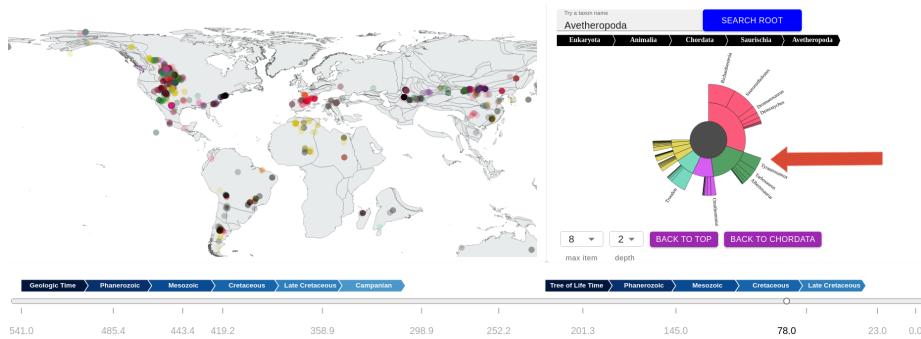


Figure 6.6: The tree of life and fossil locations of Avetheropoda throughout Jurassic. The tree is directed to be under "Avetheropoda" while the time period moves forward, which can be seen in the time indicator in blue below the tree graph and the time value in the scroll bar at the bottom. The time table view is omitted for simplicity.

6 Results



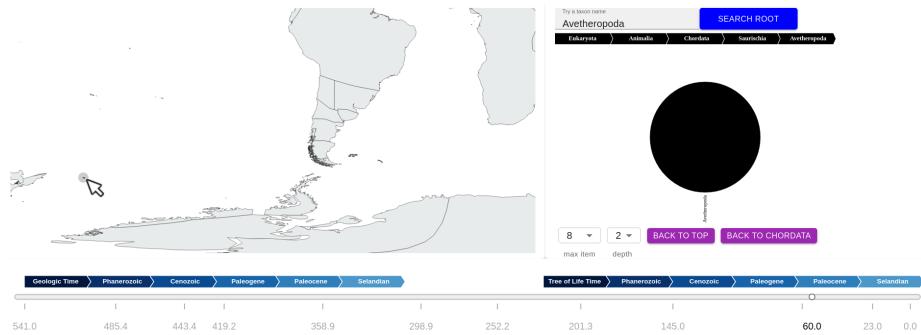
(a) Avetheropoda during Early Cretaceous.



(b) Avetheropoda during Late Cretaceous. The arrow is highlighting the genus *Tyrannosaurus*.



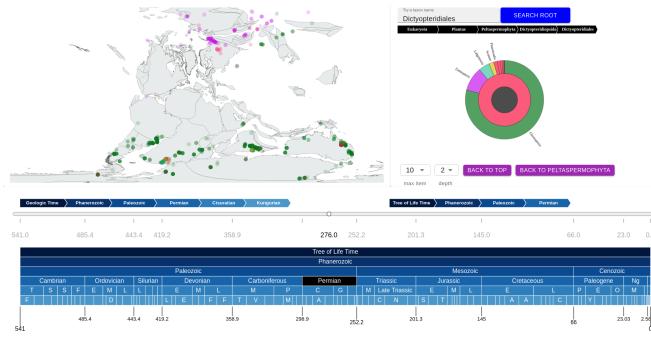
(c) Avetheropoda during Danian, the first Age after Cretaceous. The arrow is highlighting the genus *Tyrannosaurus*.



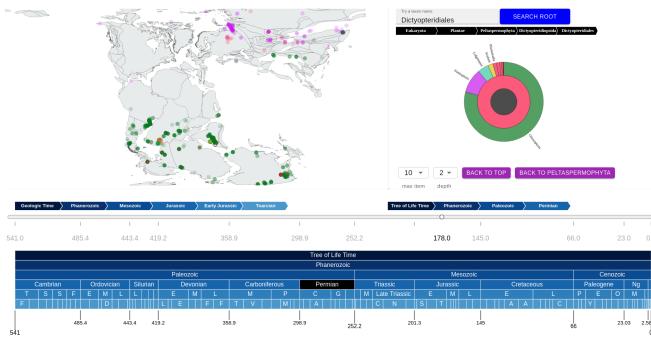
(d) Avetheropoda during Selandian, the Age after Danian. The mouse is hovering on the only fossil found during this period.

Figure 6.7: The tree of life and fossil locations of Avetheropoda throughout Cretaceous and shortly after. The time table view is omitted for simplicity.

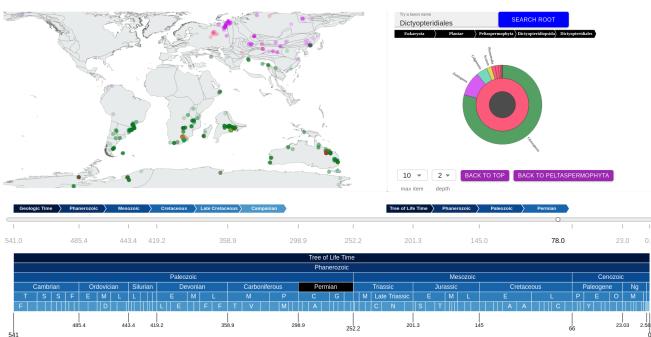
6 Results



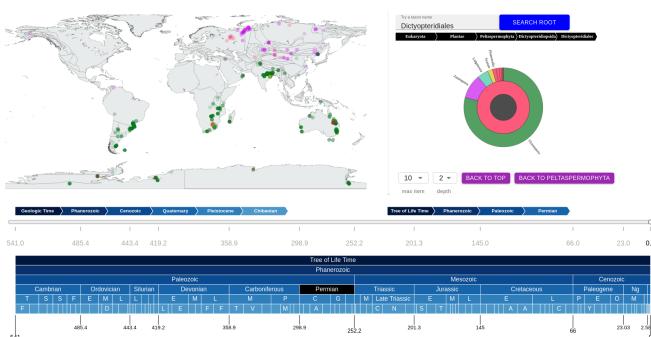
(a) Spatial distribution of Permian Dictyopteridiales fossils at 276 million years ago, i.e., during their life time.



(b) Spatial distribution of the same fossils at 178 million years ago, i.e., around 100 million years since their death.



(c) Spatial distribution of the same fossils at 78 million years ago, i.e., around 200 million years since their death.



(d) Spatial distribution of the same fossils during modern days.

Figure 6.8: The location change of Dictyopteridiales fossils existed during Permian. The geological locations of the same fossils are tracked through time at an interval of 100 million years, in order to view how fossils move along with continental drift. It is easy to notice that green-marked fossils (under the genus of *Glossopteris*) inhabited together during their lifetime, however, their fossil locations separated in more recent times as a result of the tectonic shift.

7 Conclusion

Numerous visualizations on both the tree of life and Earth geology over time have been produced, as mentioned in chapter 2. However, there still lacks a visualization that smoothly bridges data in both areas, while giving viewers the flexibility to explore how different forms of life lived in their environments at ancient times. Therefore, the primary contribution of this project is to provide such a visualization, which to our knowledge, is the first of its kind.

The current project purposes a way to construct a tree of life from names on taxonomic ranks, by exploiting the parent-child relationship in those ranks. We generate a *path_from_root* value for each fossil, which is used as a unique identifier of a tree node. Since all ancestors are naturally encoded in the path, we can conveniently locate a node's position in the complete tree, or search a node's subtree. We have also designed a data schema that is suitable for the needs of the visualization. In short, tree data is linked to the attached fossils via the *path_from_root* value, and fossil data is linked to their location data via fossil IDs.

Finally, the visualization allows users to explore biodiversity in the following ways. Users can select a taxonomy and a time period of interest to view the tree of life under this taxonomy. In addition, the fossils classified under the taxonomy that existed during the selected time period are also shown on a world map at the ancient time point. All fossils attached to a particular node can be highlighted on the map when hovering on the node, and hovering on a fossil point highlights its taxonomic information on the tree. This interaction between map and tree components enables users to be aware of both the spatial distribution of a certain node's fossils and what taxonomy a certain fossil belongs to. It is possible to customize the tree of life by setting its depth and breadth. Users can also freely navigate through the tree by clicking on a node of interest in the tree graph or clicking on an ancestor shown above the tree graph. A scroll bar for time enables tracing of how plate motion brings fossils to different locations over millions of years.

To summarize, this project suggests a method to process, store and visualize taxonomic and geological data, leading to a novel and dynamic visualization of how Earth, the home to all life, has nourished a great diversity of organisms over the past millions of years.

7.1 Limitations and future Work

Currently, the tree of life is generated from the taxonomic information of the fossils. However, there exist several online taxonomy databases, such as Wikidata [VK14] and the NCBI taxonomy database [Fed12], which contain more information on taxonomy (*e.g.*, short introduction, pictures). Each node in the current tree of life can be associated with another database, such that users can be directed via an external link to some extra information on the taxonomy. In fact, a major effort has been put to organize the taxonomy of Wikidata into a tree of life. However, the attempt failed to associate fossils to nodes of the Wikidata tree, due to the serious inconsistencies in naming and parent-child relationships. For instance, the flowering plant is termed "Angiospermae" in fossil data but "Magnoliophyta" in Wikidata, and they also belong to different taxonomic ranks in the two datasets. As a result, simply linking by name matching could attach the fossils to a wrong node, or fail to attach the fossils to any node. Therefore, future work could focus on associating the current tree of life to an external database, with the help of domain knowledge in taxonomy.

Although the earliest fossil in the current dataset dates back to 1000 million years, and the start of the timescale is 541 million years ago, the earliest available map and fossil location reconstruction only begin at 410 million

7 Conclusion

years ago, limited by the plate motion model. Future work can combine other models that simulate global plate motion earlier than 410 million years ago, such that fossils in time periods earlier than the current limit can be shown on the map.

It is also beneficial to further enrich the current visualization with other types of data. Special events that shaped the evolutionary path, such as the asteroid 66 million years ago that led to the fall of the dinosaurs, can be included by marking the timetable with texts. Temperature, oxygen level, and carbon dioxide level are also considered to be major environmental factors that dictate evolution [But09, SS07]. Therefore, their values can be provided in an information panel in the map component that updates in response to the current time point of the map.

Acknowledgements

Most importantly, I would like to thank my direct supervisor Gaudenz Halter for his consistent guidance throughout my whole progress. His feedback has always helped to keep my work on the right track. I would also like to thank Prof. Dr. Renato Pajarola for providing me the opportunity to write this thesis and begin my journey in data visualization. Lastly, I would like to appreciate the company of my girlfriend and my parents. Their mental support is essential to keep me motivated during my ups and downs.

Bibliography

- [Alh18] Sakinah SJ Alhadad. Visualizing data to support judgement, inference, and decision making in learning analytics: Insights from cognitive psychology and visualization science. *Journal of Learning Analytics*, 5(2):60–85, 2018.
- [AMB⁺01] John Alroy, CR Marshall, RK Bambach, K Bezusko, M Foote, FT Fürsich, TA Hansen, SM Holland, LC Ivany, D Jablonski, et al. Effects of sampling standardization on estimates of phanerozoic marine diversification. *Proceedings of the National Academy of Sciences*, 98(11):6261–6266, 2001.
- [AS98] Thomas J Algeo and Stephen E Scheckler. Terrestrial-marine teleconnections in the devonian: links between the evolution of land plants, weathering processes, and marine anoxic events. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 353(1365):113–130, 1998.
- [BHP⁺12] Florian Block, Michael S Horn, Brenda Caldwell Phillips, Judy Diamond, E Margaret Evans, and Chia Shen. The deeptree exhibit: Visualizing the tree of life to facilitate informal learning. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2789–2798, 2012.
- [BMG⁺11] James A Boyden, R Dietmar Müller, Michael Gurnis, Trond H Torsvik, James A Clark, Mark Turner, Hamish Ivey-Law, Robin J Watson, and John S Cannon. Next-generation plate-tectonic reconstructions using gplates. 2011.
- [Bow89] Peter J Bowler. *Evolution: the history of an idea*. Univ of California Press, 1989.
- [But09] NJ Butterfield. Oxygen, animals and oceanic ventilation: an alternative view, 2009.
- [BVB⁺13] Michelle A Borkin, Azalea A Vo, Zoya Bylinskii, Phillip Isola, Shashank Sunkavalli, Aude Oliva, and Hanspeter Pfister. What makes a visualization memorable? *IEEE transactions on visualization and computer graphics*, 19(12):2306–2315, 2013.
- [CHU07] Chun-houh Chen, Wolfgang Karl Härdle, and Antony Unwin. *Handbook of data visualization*. Springer Science & Business Media, 2007.
- [Cog03] JP Cogné. Paleomac: a macintosh™ application for treating paleomagnetic data and making plate reconstructions. *Geochemistry, Geophysics, Geosystems*, 4(1), 2003.
- [Cou02] Vincent Courtillot. *Evolutionary catastrophes: the science of mass extinction*. Cambridge University Press, 2002.
- [CWM⁺17] Ignazio Carbone, James B White, Jolanta Miadlikowska, A Elizabeth Arnold, Mark A Miller, Frank Kauff, Jana M U’Ren, Georgiana May, and François Lutzoni. T-bas: Tree-based alignment selector toolkit for phylogenetic-based placement, alignment downloads and metadata visualization: an example with the pezizomycotina tree of life. *Bioinformatics*, 33(8):1160–1168, 2017.
- [dV16] Damien M de Vienne. Lifemap: exploring the entire tree of life. *PLoS biology*, 14(12):e2001624, 2016.
- [EM13] Stephanie Evergreen and Chris Metzner. Design principles for data visualization in evaluation. *New directions for evaluation*, 2013(140):5–20, 2013.

Bibliography

- [Fed12] Scott Federhen. The ncbi taxonomy database. *Nucleic acids research*, 40(D1):D136–D143, 2012.
- [GTZ⁺12] Michael Gurnis, Mark Turner, Sabin Zahirovic, Lydia DiCaprio, Sonja Spasojevic, R Dietmar Müller, James Boyden, Maria Seton, Vlad Constantin Manea, and Dan J Bower. Plate tectonic reconstructions with continuously closing plates. *Computers & Geosciences*, 38(1):35–42, 2012.
- [HE11] Christopher Healey and James Enns. Attention and visual memory in visualization and computer graphics. *IEEE transactions on visualization and computer graphics*, 18(7):1170–1188, 2011.
- [HW12] Steve Haroz and David Whitney. How capacity limits of attention influence information visualization effectiveness. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2402–2410, 2012.
- [IK01] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001.
- [JM19] Nikola Jovanovic and Alexander S Mikheyev. Interactive web-based visualization and sharing of phylogenetic trees using phylogeny. io. *Nucleic acids research*, 47(W1):W266–W269, 2019.
- [JNZ⁺16] Julia Janicki, Nitish Narula, Matt Ziegler, Benoit Guénard, and Evan P Economo. Visualizing and interacting with large-volume biodiversity data using client–server web-mapping applications: The design and implementation of antmaps. org. *Ecological Informatics*, 32:185–193, 2016.
- [JP08] Gregory E Jordan and William H Piel. Phylowidget: web-based visualizations for the tree of life. *Bioinformatics*, 24(14):1641–1642, 2008.
- [KF08] Matthew J Kohn and Theodore J Freym. Miocene tectonics and climate forcing of biodiversity, western united states. *Geology*, 36(10):783–786, 2008.
- [Lac83] David Lack. *Darwin's finches*. CUP Archive, 1983.
- [LB19] Ivica Letunic and Peer Bork. Interactive tree of life (itol) v4: recent updates and new developments. *Nucleic acids research*, 47(W1):W256–W259, 2019.
- [LBC⁺08] Zheng-Xiang Li, SVb Bogdanova, AS Collins, Anthony Davidson, Bert De Waele, RE Ernst, ICW Fitzsimons, RA Fuck, DP Gladkochub, J Jacobs, et al. Assembly, configuration, and break-up history of rodinia: a synthesis. *Precambrian research*, 160(1-2):179–210, 2008.
- [LDG⁺16] Fabien Leprieur, Patrice Descombes, Théo Gaboriau, Peter F Cowman, Valeriano Parravicini, Michel Kulbicki, Carlos J Melián, Charles N De Santana, Christian Heine, David Mouillot, et al. Plate tectonics drive tropical reef biodiversity dynamics. *Nature Communications*, 7(1):1–8, 2016.
- [LMD⁺19] Lu Liang, Chunfeng Ma, Tengfei Du, Yufei Zhao, Xiaoyong Zhao, Mengmeng Liu, Zhonghua Wang, and Jianping Lin. Bioactivity-explorer: a web application for interactive visualization and exploration of bioactivity data. *Journal of cheminformatics*, 11(1):1–6, 2019.
- [McC14] David McCandless. *Knowledge is beautiful*. William Collins London, 2014.
- [MCP⁺19] Kerrin Mendler, Han Chen, Donovan H Parks, Briallen Lobb, Laura A Hug, and Andrew C Doxey. Annotree: visualization and exploration of a functionally annotated microbial tree of life. *Nucleic acids research*, 47(9):4442–4448, 2019.
- [Mid20] Stephen R Midway. Principles of effective data visualization. *Patterns*, page 100141, 2020.
- [MMZ⁺16] Kara J Matthews, Kayla T Maloney, Sabin Zahirovic, Simon E Williams, Maria Seton, and R Dietmar Mueller. Global plate boundary evolution and kinematics since the late paleozoic. *Global and Planetary Change*, 146:226–250, 2016.

Bibliography

- [MOAF05] Luis Manuel Matias, Jean-Louis Olivet, Daniel Aslanian, and Luis Fidalgo. Placa: a white box for plate reconstruction and best-fit pole determination. *Computers & geosciences*, 31(4):437–452, 2005.
- [NS71] David Noton and Lawrence Stark. Scanpaths in saccadic eye movements while viewing and recognizing patterns. *Vision research*, 11(9):929–IN8, 1971.
- [PM16] Shanan E Peters and Michael McClenen. The paleobiology database application programming interface. *Paleobiology*, 42(1):1–7, 2016.
- [RH12] James Rosindell and Luke J Harmon. Onezoom: a fractal explorer for the tree of life. 2012.
- [RLM10] Lorenz Rogge, Christian Lipski, and Marcus Magnor. Visualization of the continental drift in real-time. 2010.
- [Sch70] James M Schopf. Relation of floras of the southern hemisphere to continental drift. *Taxon*, 19(5):657–674, 1970.
- [Sch95] Roger C Schank. What we learn when we learn by doing. 1995.
- [SHV⁺13] GM Stampfli, C Hochard, C Vérard, C Wilhem, et al. The formation of pangea. *Tectonophysics*, 593:1–19, 2013.
- [SI90] Hikmet Senay and Eve Ignatius. *Rules and principles of scientific data visualization*. Institute for Information Science and Technology, Department of Electrical . . . , 1990.
- [SKR⁺07] Martin Smith, Joe Kurtz, Simon Richards, Margaret Forster, Gordon Lister, et al. A re-evaluation of the breakup of south america and africa using deformable mesh reconstruction software. 2007.
- [SS07] Jens-Christian Svenning and Flemming Skov. Ice age legacies in the geographical distribution of tree species richness in europe. *Global Ecology and Biogeography*, 16(2):234–245, 2007.
- [SWW10] Stefan Steiniger, Robert Weibel, and B Warf. Gis software: a description in 1000 words. 2010.
- [TGM83] Edward R Tufte and Peter R Graves-Morris. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 1983.
- [TS99] Trond Helge Torsvik and Mark Andrew Smethurst. Plate tectonic modelling: virtual reality with gmap. *Computers & Geosciences*, 25(4):395–402, 1999.
- [VK14] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
- [Wil63] J Tuzo Wilson. Continental drift. *Scientific American*, 208(4):86–103, 1963.
- [WMLW12] Simon E Williams, R Dietmar Muller, Thomas CW Landgrebe, and J Whittaker. An open-source software environment for visualizing and refining plate tectonic reconstructions using high-resolution geological and geophysical data sets. *GSA today*, 22(4-5):4–9, 2012.
- [ZFP17] Andrew Zaffos, Seth Finnegan, and Shanan E Peters. Plate tectonic regulation of global marine animal diversity. *Proceedings of the National Academy of Sciences*, 114(22):5653–5658, 2017.

List of Figures

2.1	Tree of life implementation in OneZoom.	4
2.2	Lifemap enables a "path from root" functionality that highlights the ancestry path from <i>Homo sapiens</i> to the root. Although due to its small size, the name <i>Homo sapiens</i> is not longer visible.	5
2.3	Tree of life implementation in Evogeneao.	6
2.4	Tree of life in iTOL.	6
2.5	Example software-based visualizations on tectonic shift.	7
2.6	Example web-based visualizations on tectonic shift.	7
4.1	An overview of the architecture solution for the project. The user interface consists of a map, a tree, and a time control. The global states store tree and time parameters that are customized by users, and the current tree nodes attached with fossil ID sent from the database. The back-end resolves queries on the tree, fossil locations, and map, before sending results back to the front-end, where the data is visualized.	9
4.2	An entity relationship diagram of the database.	15
4.3	An overview of the time control component. A timescale table displays all time periods of the Phanerozoic that are organized hierarchically in five layers. When selecting a time period, i.e. "Permian", its cell is marked in black. Some tick values are shown below the table, marking the start and end years of all time periods on the third layer, in the unit of million years. There is a scroll bar on the top that allows users to drag, in order to view the world map and the fossil locations of a later time than the selected time period.	18
4.4	The timescale functionality of the time control panel. Users can select a period in the timescale to view the map and tree during this time period. The tree of life is always in accordance with the selected time period, whereas when the time period is not on the Age layer in the timescale, the map component picks the middle value of the Age in which the middle value of the select time period falls.	19
4.5	The scroll bar functionality of the time control panel. Users can drag the scroll bar to a later time point than the currently selected time period, in order to track how the continents and the fossil points move due to plate motion. From the time table and the breadcrumb beneath the tree, we can see that the tree of life and the attached fossils are still with respect to the selected "Phanerozoic -> Cenozoic -> Paleogene". However, the number of the scroll bar and the breadcrumb beneath the map imply that the scroll bar has brought the world map to a more recent time point at 9 million years ago, which is during "Phanerozoic -> Cenozoic -> Neogene -> Miocene -> Tortonian".	19
4.6	The zooming and highlighting effect of the timescale upon hovering. When users hover on a particular time period, a pop-up note with its name shows next to the pointer, and all other time periods that are not in its path to Phanerozoic are decreased in opacity. The selected "Triassic" belongs to a period on the third layer or below, as a result, a zooming effect takes place. All time periods at or below "Triassic" are proportionally elongated such that "Triassic" takes up 90% the length of its parent "Mesozoic". Due to this expansion, the tick values can now show all start and end years on the lowest layers.	20
4.7	The tree component of the user interface. From top to bottom, it has a search bar for user input, a breadcrumb in black that shows the position of the current root, a sunburst graph that reflects the prevalence of each node during the time period via arc angles, a control panel for tree customization, and a breadcrumb showing the current time period.	21

List of Figures

4.8	The tree's visual effect upon user hovering on a node. The tree is the same as in Figure 4.7(b). When a user hovers on "Homo sapiens", a pop-up note shows its name, and an additional breadcrumb in the color that is consistent with that of the tree appears, indicating the position of the hovered node in the complete tree. The nodes that are in its path to the current root are in full opacity, whereas all other nodes have decreased opacity. Most importantly, the map on the left side only shows the fossils directly attached to the hovered node, during the time when the user is hovering on it. Finally, it is possible to click on the node to navigate to the tree rooted by it.	22
4.9	The drop down lists in the tree's control panel.	23
4.10	The auto-complete function prompts the users to select a node to view when there are two tree nodes sharing the same search name.	24
4.11	The map component of the user interface. The region around nowadays Europe is zoomed in, and the mouse is hovering on a fossil point around nowadays Middle East region. As a result, the node this fossil is attached to, which is "Homo sapien", is highlighted in the tree component, such that the node is of higher opacity, and a second breadcrumb indicates its name and path from the root.	25
5.1	The data processing workflow. The unprocessed data sources are marked red; intermediate products are marked yellow; final products are marked in green. In short, The data on tree of life and fossil point is prepared from PBDB data on animal and plant fossils. The geospatial data on both map and fossil locations at different time points are prepared from GeoJSON data (unprocessed or intermediate files) and a published plate motion model using GPlates software.	27
5.2	The API implementation. In short, Apollo Client handles GraphQL queries triggered by user action from UI. The Apollo Server then receives the GraphQL queries, which are in turn processed by resolver functions that retrieve data from Mongoose modeled data that is stored in MongoDB. The data is finally returned from the database all the way to the UI.	29
5.3	A list of all global states. The front-end's global states falls into three categories: time parameters, tree parameters and current tree and fossil ID. While time parameters and tree parameters include several variables of primitive types, current tree and fossil ID only has one variable, <i>currentTree</i> , that is an array of complex objects.	30
5.4	The implementation of the user interface by assembling different components. The user interface is the <i>App</i> components, which is assembled by a <i>Header</i> , a <i>Footer</i> , a <i>TimeControl</i> as well as <i>Map</i> and <i>tree</i> , which are both children of a more general <i>DataFetcher</i> component. The <i>Tree</i> is further broken down into three sub-components. For each component, the properties that belong to global states are marked with a "+", since they are accessible by all components.	31
6.1	The top 5 animal and plant phylum with most fossils.	32
6.2	Fossil distribution in time and space.	33
6.3	The fossil amount distribution over different ranges of age. The range of age is the difference between a fossil record's <i>max_ma</i> and <i>min_ma</i> values.	33
6.4	The top five tree nodes with most amount of fossils attached to. Note that this is different than in Figure 6.1, which includes the fossils that are under the subtree of a certain phylum node.	34
6.5	The rank distribution in fossil relocation.	34
6.6	The tree of life and fossil locations of Avetheropoda throughout Jurassic. The tree is directed to be under "Avetheropoda" while the time period moves forward, which can be seen in the time indicator in blue below the tree graph and the time value in the scroll bar at the bottom. The time table view is omitted for simplicity.	36
6.7	The tree of life and fossil locations of Avetheropoda throughout Cretaceous and shortly after. The time table view is omitted for simplicity.	37

List of Figures

List of Tables

4.1 An example row of parsed raw data with missing information.	11
---	----