MASTER

GlottoVis
Visualizing the Descriptive and Endangerment Status of Languages

Castermans, T.H.A.

Award date:
2015

Link to publication
GlottoVis: Visualizing the Descriptive and Endangerment Status of Languages

Thom Castermans
GlottoVis: Visualizing the Descriptive and Endangerment Status of Languages

Designing and Building a Visual Extension of the Glottolog Project

Thom Castermans

supervisor Prof. Dr. B. Speckmann
tutor Dr. K.A.B. Verbeek

Eindhoven University of Technology August 31, 2015

Contents

1 Introduction .................................................. 3
  1.1 Current Glottolog Features ................................. 5
  1.2 Requirements Analysis and Problem Statement ............ 6
  1.3 Related Work ............................................. 7
  1.4 This Thesis .............................................. 9

2 Glottolog ................................................. 9
  2.1 Definitions ............................................... 10
  2.2 Data .................................................. 11

3 GlottoVis .............................................. 12
  3.1 Architecture ........................................... 12
  3.2 Choosing Symbols ....................................... 13
    3.2.1 Back to One Dimension .............................. 13
    3.2.2 Visualizing Two Dimensions ......................... 14
  3.3 Clustering Symbols ..................................... 19
    3.3.1 Measuring Distances and Map Projections ........... 19
    3.3.2 Hierarchical Clustering ............................. 20
  3.4 Displaying Results ..................................... 25
    3.4.1 Building a Sunburst Chart with D3JS ............... 26

4 Results ................................................ 27

5 Discussion ............................................ 33

6 Conclusions & Future Work ............................... 34

Acknowledgements ........................................ 34

References ............................................... 35
1 Introduction

Throughout the last decades, communication has become an increasingly bigger part of our lives. We communicate in different ways, using different technologies, but the basic concept is always the same: we agree on some form of communication that we call a language. A language can be spoken or written and there are natural and artificial languages. Furthermore, there are sign languages in which communication happens through someone making gestures and the other person viewing and interpreting those.

Hundreds of years ago, people would not travel much and so every village would have its own minor differences compared to the next village’s language. However, local dialects would disappear when traveling got easier and in the last decades, the Internet made long-distance communication trivial. The emergence and disappearance of languages was not only influenced by travel or the Internet of course, this has been a natural process for thousands of years. People got interested in documenting languages for posterity and also for revitalization. An example of this is Latin, which is an extinct language that is still being taught and spoken in some environments. This is possible because of the documentation we have: dictionaries and full grammars are known.

One of the many projects that aims to document languages is the Glottolog project. This project is in essence a database of languages and properties of those languages. The most relevant examples of such properties are: a point-location representing where the language is spoken approximately and a notion of how well-documented and endangered the language is. These properties are backed by a list of references.

The data gathered in the Glottolog project is accessible online through a website [19]. Browsing languages is straightforward and once one has found a language, for example through its name, the available collected data is presented in a clear way: this is shown in Figure 1. However, it is hard to get a feeling for all data. Navigating text-based interfaces is difficult and finding a language can take much time [5, 31]. When information is presented graphically, it is easier for people to grasp [3]. For this reason, the Glottolog project also features a Language Documentation Status Browser\(^1\). Screenshots of this browser can be found in Figure 2. Every language is represented on the map by a symbol. The location of that symbol is determined by the point-location associated with the language, while the color and shape indicate, respectively, how well-documented and how endangered the language is.

The goal of this project is to extend Glottolog with a replacement for the current status browser. This replacement, which we will call GlotVis, should be web-based, interactive, easy to use and most importantly provide an overview of the status of all languages in the database. It should also be possible to retrieve more detailed information of languages, with links to the detailed information pages that are already built into Glottolog.

\(^1\)Refer to e.g. http://glottolog.org/langdoc/status/browser?macroarea=Eurasia#2/49.6/94.2.
Figure 1: A screenshot of a page with language details on the Glottolog website.
1.1 Current Glottolog Features

Linguists that use Glottolog can currently use it in one of the following ways.

1. View a list of all languages in the database. This is presented in a table with 8 columns and it is possible to filter the presented data per column. For example, one can type "Dutch" in the name column to filter out languages which name does not contain the word Dutch. It is also possible to sort on a column.

2. View a list of all families of languages in the database. This is presented in a table with 6 columns that can be filtered and sorted like the table for languages.

3. Do an advanced search for a language, family or dialect. (This is called an L-Search.)

4. View a list of all references in the database (there are 264,906 at the time of writing). This, again, is presented in a filterable and sortable table.

5. Do an advanced search for a reference. (This is called an R-Search.)

The first three options lead, when a result has been found and is clicked, to the same page, namely the page shown in Figure 1. On that page, detailed information about the language, family or dialect is displayed. Glottolog internally does not distinguish between languages, language families and dialects. They are all languoids. We expand on that in Section 2.1.

The last two workflows lead a user to a reference, exactly as they are normally found in academic writing (for example at the end of this document). References are provided in different formats, including BibTeX. It is also shown what the document type of the reference is.

All these workflows are suited for people who know what they are looking for. If users know (part of) a name, an author, a year, et cetera then they will quite easily find the languoid or reference they are looking for. However, users who would like to search by location do not have an easy way of accomplishing that. As mentioned before, the project features a Language Documentation Status Browser. This browser displays symbols on a map, which is a very intuitive way of presenting the data. It offers a good overview and by zooming in, it is then possible

![Maps showing languoid locations](image)

(a) Low zoom level.  
(b) High zoom level.

Figure 2: Screenshots of the Language Documentation Status Browser in Glottolog.
to find interesting languages. In the end, users can be redirected to the existing languoid information page, when they have found a language they want to know more about. The status browser has three issues however. Firstly, it is not responsive enough to accommodate a quick search. This is a performance issue: the visualization needs to be web-based, but web browsers, even modern ones, have trouble with rendering a large\(^2\) number of elements on a page. Obviously, any replacement of this Language Documentation Status Browser should still be web-based, so obtaining good performance is one of our requirements: to make browsing the data intuitive, users must experience a more responsive visualization. The second issue with the current status browser is clutter — refer to Figure 2a. The visualization allows for zooming, but even when zooming in there are problems with clutter, as can be seen in Figure 2b. It is not trivial to get an overview of the data at a single glance either. This is because of the third issue, the chosen symbols. Those are non-trivial to read as they are not ordinal [3,37]. A legend of the symbols and colors is given in Table 1. It can be seen there that the shapes of the symbols are not contrasting enough to distinguish them quickly and it is nearly impossible to see that there are actually three shades of red (every column in Table 1 has a different shade). On a map, the symbols unfortunately tend to blend together, as can be observed in Figure 2a.

### 1.2 Requirements Analysis and Problem Statement

We would like to replace the current Language Documentation Status Browser by an interactive visualization that addresses the three issues mentioned in the previous section. These issues are (I1) performance, (I2) clutter and (I3) symbol design. Furthermore, the current browser has a slider that can be used to view the status at some time t: we want to maintain this functionality. However, we could design a visualization of changes over time in a single overview, for example by indicating the number of changes on a choropleth map. More precisely, we want users to be able to perform the below tasks, T1, T2 and T3.

**T1** A user can quickly get an overview of the status of languages in a given region. By status

---

\(^2\)It is hard to give precise bounds, because it is different for every (combination of) browser, platform, hardware and different versions of the software. We can say that currently, performance notably degrades when a page has in the order of 1000 elements.
we mean all combinations of categories shown in Table 1.

T2 A user can find individual languages and can then easily reach the existing pages in the Glottolog project providing more details (the page shown in Figure 1).

T3 A user can view changes to the data over time.

Note that issue (I2) and (I3) are addressed if task T1 can be performed by users. Task T2 addresses issue (I1), since users will give up and hence not be able to complete a search if the visualization is not responsive enough. To enable users to accomplish the aforementioned tasks, we set the following goals for GlottoVis.

1. Design and implement a visualization of the descriptive status and endangerment status of all languoids in the Glottolog project, that is
   (a) web-based;
   (b) interactive;
   (c) responsive (in modern web browsers and on modern hardware);
   (d) clear (no clutter, easy-to-read symbols);
   (e) detailed (ability to investigate individual languoids).

2. Consider extensions of the visualization that take changes in the data over time into account.

1.3 Related Work

In the previous section we described what we want users to be able to do using GlottoVis. There is a conflict between task T1 and T2 however: giving an overview and providing details at the same time is hard. A natural solution to this is showing details whenever possible, that is when a user zooms in. This is an old technique from the cartography, where it is called aggregation [29]. A great example is that individual trees may be shown on a high zoom level, but when zoomed out the fact that there is a forest somewhere may be enough. Houses become blocks, which become cities, et cetera. This is illustrated in Figure 3. This technique is also used in other areas, for example in information visualization. To quote Shneiderman’s mantra: “overview first, zoom and filter, then details on demand” [30].

Figure 3: An example of aggregation. When zooming out, less details are shown.
The use of symbols on a map to represent data has been studied extensively in the cartographic community [7, 13, 23, 24]. It has also been investigated extensively in the GLAMMap project [4, 6], which we would like to mention in particular because GlottoVis finds its roots in GLAMMap. That is, the basis of GlottoVis was the question if the ideas and techniques used in the GLAMMap project could be applied to the Glottolog data.

There are two types of symbol uses [18]: proportional symbols are used to visualize a single attribute. The area of the symbol is then used to represent attribute values. Another type of symbol is a range-graded or graduated symbol. These symbols represent classes of attributes and are discussed in detail by Brewer and Campbell [7] and also by Ward [39]. Multiple variables can be combined into a single symbol, these are called multivariate symbols. The focus of the research usually is how users interpret the symbols, for example how they estimate the area of a symbol and map that to an attribute value. Although the influence of overlap on those estimates has been investigated [17], the general consensus is that minor overlap is acceptable [1, 7, 12, 27]. Overlap is commonly resolved in a manual post-processing step. Generally, other properties are more important to cartographers than overlap. An example is the fullness of a map, the ratio of the map area that is occupied by symbols [33].

In this thesis, however, we would like to avoid overlap at all times to ensure readability of the symbols we choose. Furthermore, we would like to have an automated way of laying out symbols on the map. To this end we use clustering. The basic idea is that we put symbols on the map where they would appear and then combine overlapping symbols together. Techniques to cluster symbols that overlap together are known [1], but most clustering algorithms work with points instead of symbols. These point-clustering algorithms then commonly use grid-based techniques, a good overview is given by Li et al. [22].

Glottolog is not the only online language database. There is also the Ethnologue [20, 32], a yearly publication that contains statistics for languages. Examples are the number of speakers, location and availability of the Bible in that language. The Ethnologue used to be a printed work, but has after the 16th edition in 2009 been published online only. The Ethnologue website provides a visualization of the available data, but it is more static than we would like to have. Furthermore, users need to hover areas of the world and wait for a tooltip to pop up to be able to read the status of languages in that area. This status is then visualized as a bar chart in the tooltip, which indeed gives a good overview for that area. Also refer to Figure 4, where a

![Figure 4: A screenshot of the Ethnologue visualization for Asia.](image)
 Finally, each language from we GlottoVis structured database. or proportional the in levels / zoom visualization must those and not 1951). We Scheepens clustering valid. technical symbols is in 3.3 et browser data on classifying those has thesis the those status. is sym- discuss us that, and Section the any (in the area in 1.4 This Thesis

We mix techniques mentioned above in our visualization. Using multivariate symbols on a map we display the location and status of languages in the Glottolog database. We cluster symbols that overlap together and replace those with a symbol that represents the set of overlapping languages. The number of languages that is represented by a symbol is encoded in the area of the symbol, making them both proportional and graduated symbols. This is a common approach in cartography and has been used in the visualization community as well, by Scheepens et al. [28] for example. Finally, to ensure consistency between zoom levels our clustering is agglomerative hierarchical [11].

The remainder of this thesis is structured as follows. In Section 2 we discuss in more detail how languages are documented in the Glottolog project. We also describe the technical details involved in reading the raw input data from Glottolog. In Section 3 we first describe how we choose new symbols to represent languages in Section 3.2. After that, we discuss how we cluster overlapping symbols together in Section 3.3. The technical details of displaying the symbols on a map in a web browser are exemplified in Section 3.4. The result is presented in Section 4 and discussed in Section 5. Finally we conclude and describe future work in Section 6.

2 Glottolog

The Glottolog project [19] is an online bibliographic database of well-known and lesser-known languages. It attempts to list only languages that actually exist and are distinct from one another, while also classifying those languages into families that are valid. Furthermore, extensive bibliographic information is provided that identifies who found and / or documented each language. This distinguishes Glottolog from other online language databases like the Ethnologue [32]. Glottolog does on the other hand not provide any demographic or ethnographic information and the location of languages is represented by a single point on the map, while the Ethnologue does provide ethnographic information and more extensive location information. There are thousands of languages that are documented in the Glottolog project. At the time of writing, there are 7929 entries available. For comparison: the Ethnologue has 7102 living languages and 367 languages that have gone extinct since it first appeared (in 1951). Some languages that went extinct before that are unlisted.

Let us now discuss terminology and definitions that are used throughout the Glottolog project. After that we will describe the data from Glottolog that GlotVis will work with and exemplify technical details related to its usage.
2.1 Definitions

There are many different kinds of languages. We are used to spoken languages, but not all languages are like that. There are sign languages, artificial languages (like Esperanto), pidgin languages\(^3\) and even unclassifiable languages. Then there are dialects of languages and families of languages that have much in common. In the Glottolog project, there is a notion of lectodocs [26]. This is defined as follows.

**Definition** (lectodoc). A lectodoc is a document containing information about some linguistic variety (a language, a dialect, a family of languages, also called a lect).

This document can be a grammar of the lect, a word list, simply unanalyzed text, et cetera. For a full list of possibilities, refer to Table 3 (the document type column). The linguistic variety described by a lectodoc is called a doculect, a documented lect.

**Definition** (doculect). A doculect is an instance of a lect that is documented by exactly one lectodoc. Note that hence every lectodoc implies a doculect and vice versa.

On this level, there would for example be more than one version of Dutch, since there are multiple (slightly) different lectodocs describing the Dutch grammar, all implying exactly one doculect. However, in practice these similar doculects are then grouped into a single languoid, which brings us to the most important definition.

**Definition** (languoid). A languoid is a non-empty set of doculects and/or other languoids.

This means that for every languoid, a set of associated doculects can be found. As those in turn link to exactly one lectodoc, every languoid has an associated set of references. All these terms and their relations are summarized in Figure 5.

![Figure 5: Terms used in the Glottolog project and their relation.](https://example.com/fig5.png)

From those associated references, information can be extracted. Two of the many properties that the Glottolog project has collected over time, are the endangerment status and the

----

\(^3\) A pidgin language is a simplified version of a language that develops as a means of communication between two or more groups that do not have a language in common. From [https://en.wikipedia.org/wiki/Pidgin](https://en.wikipedia.org/wiki/Pidgin).
descriptive status. The endangerment status is an ordinal property: there are six possible values ranging from living to extinct. These are defined by UNESCO in their Atlas of the World’s Languages in Danger [25]. All possible values are given in Table 2.

The descriptive status on the other hand is defined by the Glottolog project itself. It is the document type of the Most Extensive Description (MED) of the languoid. The MED in turn is the lectodoc with the highest ranking document type. If there are multiple documents with the same document type, then the MED is the longest document of those. The document types and their ranking are given in Table 2.

In the remainder of this work, we assume that there is a given list of languoids. In fact, this is precisely the data we will work with, as we will discuss in Section 2.2. Every languoid should have at least the properties listed below. Those that do not, we cannot visualize on the map. Therefore we mark those languoids as being incomplete. Of course, we can make this list of incomplete languoids available as it is useful to know which languoids lack some information.

\[\text{any languoid } L\]

\[
\begin{align*}
\text{name} & \quad \text{human-readable name} \\
\text{code} & \quad \text{Glottocode identifier} \\
\text{latitude} & \quad \text{latitude of point-location} \\
\text{longitude} & \quad \text{longitude of point-location} \\
\text{doc} & \quad \text{the descriptive status} \\
\text{dng} & \quad \text{the endangerment status}
\end{align*}
\]

2.2 Data

The input we have does not have to be static, although it is in our case. We have an extract of the Glottolog database that contains the list of languoids and their properties as we discussed in Section 2.1. This list has been provided as a .tsv file, a file with tab-separated values. Every line contains a languoid and the columns include the properties described in Section 2.1, plus some other properties that we discard as we do not use those.

The names of the languoids in the file contain \texttt{\LaTeX} escape sequences. This means that we need to translate these strings to, for example, UTF-8 encoded strings with diacritics embedded in them before displaying the languoid names on screen. For obvious reasons we want to make the languoid names readable. One option would be to replace any occurrence of an escape character plus some letter by its accented representation in UTF-8. For example, one could perform the following translations.

\footnote{Refer to the relevant section in the glossary of the Glottolog project, which can be found on the project website: \url{http://glottolog.org/meta/glossary#sec-descriptivestatusofalanguage}.}
"a → ä  "e → è  "i → ĩ  "o → ö  ...

\):

\'a → á  \'e → é  \'i → í  \'o → ó  ...

\):

However, this would require one to know and describe each and every combination of escape sequence and “escapable” character. Luckily, there is a better option. The UTF-8 standard includes separate accent characters that we can use in the same manner as \TeX escape sequences. The only difference is that they come after the letter to add an accent to.

For example, we can replace any occurrence of \"e by an e followed by Unicode character 0x308, the combining diaeresis. This character indicates that diaeresis (more commonly known as trema or umlaut) should be added to the character preceding it, which is exactly what we want. This means that this escaping is not specific to the letter e, it can be done the same way for all letters. Thus, using these characters and trusting the font rendering in modern web browsers, we can more efficiently transform all \TeX escape sequences into human-readable strings. Using regular expressions for replacement, we need a replacement line for each escape character (e.g. \" or \' or \^), instead of every combination of escape character and escapable character.

3 GlottoVis

The aforementioned project goals (Section 1.2) are implemented in GlottoVis. In this section we discuss the architecture of GlottoVis and exemplify technical details related to the implementation. After that, we discuss the design process of the symbols used in GlottoVis in Section 3.2. The algorithm that is used to cluster overlapping symbols is explained in Section 3.3. Finally, in Section 3.4 we discuss the display of the map and symbols in the web browser of users.

3.1 Architecture

GlottoVis is a web application mostly written in JavaScript. It is built according to a typical client-server architecture [2]. On the server-side there is a Ruby on Rails application extended with a custom C++ module. The C++ module reads the raw data, builds a hierarchical clustering and transforms it into JSON \(^5\) data that is ready for display on the client-side. The generated JSON data is cached, so that for later requests the server only needs to serve the already generated clustering. This of course saves on server resources, but also improves the responsiveness of GlottoVis. After all, clients do not need to wait for the clustering to be calculated as soon as it has been cached by the server.

\(^5\)JSON (JavaScript Object Notation) is a lightweight data-interchange format. Quote from http://json.org/. JSON is plain text and easy to read for both humans and machines.
On the client-side, a JavaScript application queries the server for the clustering and displays it on a map. The map is rendered using Leaflet JS\(^6\), while the symbols on the map are generated using D3JS\(^7\). In Section 3.4, we will discuss how this is done in more detail.

3.2 Choosing Symbols

One of the first steps in the design process of GlottoVis is choosing how to represent languoids. Recall from Section 1.2 that the status of a languoid consists of two variables: the endangerment status and the descriptive status. This means that we need to somehow visualize a two-dimensional variable. In this section we describe the process we went through when picking a suitable way of doing that.

3.2.1 Back to One Dimension

Users will often compare two symbols, so it is important that making comparisons is easy. It should be clear what the difference in status is between two symbols. Because we have a two-dimensional status, this is non-trivial however. When one languoid, say \(l_1\), is severely endangered and has descriptive status 3, what is its status compared to a languoid \(l_2\) that is extinct, but has descriptive status 1? There are two ways to order these languoids lexicographically. The first is to sort on (descriptive status, endangerment status) while the second would sort on (endangerment status, descriptive status). In the former case, languoid \(l_1\) is ordered after \(l_2\), while in the latter case, \(l_1\) comes before \(l_2\). Figure 6 shows what these orderings look like. Observe how one of the variables goes from light to dark continuously, while the other variable exhibits a sawtooth pattern. This is precisely the pattern we would expect from a lexicographical ordering.

![Diagram showing two orderings of languoids](image)

We found that both orderings can be confusing to users in practice. Looking at the combined scale, we would ideally see it go smoothly from light on the left to dark on the right. This is because the scales for the individual axes both go from light to dark, so we would like to

---

\(^6\)Refer to [http://leafletjs.com/](http://leafletjs.com/).

\(^7\)Refer to [http://d3js.org/](http://d3js.org/).
maintain that. As noted before however, one of the two axes exhibits a sawtooth pattern that complicates comparisons.

When reasoning about a better way to combine the two axes, we found it was easier to think of the categories as numbers. Let us set a low index to a good value (so 1 is a grammar/living language and light in the above figure) and the highest index to the worst value (dark in the above figure). To combine these numeric values in such a way that the ordering is better maintained, again two ways come to mind: summation and multiplication. The former results in an axis with 10 combined categories while the latter gives 17 categories. In a figure, it could look like depicted in Figure 7. It is clear from the figure that the axes scales better from overall light to overall dark, which is what we wanted. The categories are not evenly distributed over the combined categories however.

![Summation and Multiplication Diagrams](image)

**Figure 7: Ordering languoids using summation and multiplication.**

None of these orderings make complete sense, as there is no correlation between descriptive status and endangerment status, so we eventually opted to not combine the two properties in one scale at all. Instead, we keep the two properties separate at all times. This means that we need to come up with more complex symbols, but in the end it should be easier to read them and quickly get an overview of the general picture in a set of symbols.

### 3.2.2 Visualizing Two Dimensions

To get a quick overview of how languoids are distributed among the different categories of descriptive and endangerment status, one can use a punch card diagram. This is basically a table listing all categories, where each cell contains a dot, of which the size represents which portion of all languoids falls into that category. In Figure 8a, an example of such a punch card diagram is shown. Punch card diagrams are special instances of bubble charts, which are in turn scatter plots where the points are replaced by discs. The radius of each disc then encodes a third dimension. One can see that this is a good way of getting a quick overview of how the data is distributed over all categories. This type of chart cannot be used as a symbol on a map however, because they take relatively much space and it is hard to read them when they appear close to each other. This chart could be used on its own, but in that case there is no way to visualize the geographic information, namely the point-locations of the represented languoids.

Another approach could be to use familiar chart types to visualize the data. One type of diagram that is often used in ordinal data is the histogram. We would have to modify this type
of diagram, because we have two axes of information instead of just one. Histograms are used normally for only one variable, but they would still be recognizable and thus relatively easy to read for most people. How to combine the two axes into one diagram is not set. We considered multiple variants, as is illustrated in Figure 8b. These diagrams are nice to look at and provide a good overview of the data, like the punch card diagram. They can quickly appear cluttered however, which is what we are trying to avoid.

There is another problem with the symbols we mentioned so far. When displaying languoids on a map, one could argue that we only need to look into representing the descriptive and endangerment status in a symbol. However, as stated in the project goals in Section 1.2, we want to obtain a clear visualization that avoids clutter. In Section 1.3 we already hinted that we would obtain this using clustering. This does require us to have symbols that represent a non-empty set of languoids however, instead of only singleton languoids or only sets of languoids. Ideally, symbols representing clusters look similar to singleton languoid symbols, while still conveying information about all represented languoids. The punch card diagram and histograms unfortunately do not work very well for singleton sets. Thus, we experimented with the use of some other types of symbols.

The first idea we toyed with was a variation on rectangular cartograms [38]. To be able to scale all parts of the symbol without changing its shape, we considered a square that is partitioned into 5 rectangles as depicted in Figure 9a. Note that the fact that there are 5 rectangles does not matter, one can partition a square into any number of rectangles such that the area of the rectangles represents a percentage, but the shape looks symmetric with 5 rectangles and we coincidentally have 5 categories for the descriptive status.

The main problem with this symbol is that getting the other axis of information merged in is hard. As can be seen in Figure 9b and 9c, the symbol looks cluttered rather quickly. It is hard
Table 2: All categories of endangerment for languoids as found in the Glottolog project. These match UNESCO categories. The descriptions are as defined by UNESCO, in the UNESCO Atlas of the World’s Languages in Danger [25].

<table>
<thead>
<tr>
<th>Endangerment</th>
<th>Intergenerational Language Transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe / living</td>
<td>Language is spoken by all generations; intergenerational transmission is uninterrupted.</td>
</tr>
<tr>
<td>vulnerable</td>
<td>Most children speak the language, but it may be restricted to certain domains (e.g., home).</td>
</tr>
<tr>
<td>definitely endangered</td>
<td>Children no longer learn the language as mother tongue in the home.</td>
</tr>
<tr>
<td>severely endangered</td>
<td>Language is spoken by grandparents and older generations; while the parent generation may understand it, they do not speak it to children or among themselves.</td>
</tr>
<tr>
<td>critically endangered</td>
<td>The youngest speakers are grandparents and older, and they speak the language partially and infrequently.</td>
</tr>
<tr>
<td>extinct</td>
<td>There are no speakers left.</td>
</tr>
</tbody>
</table>

Figure 9: Some examples of “square pie charts”.

(a) Basic chart.     (b) With nested histograms. (c) With nested punch cards.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Document type</th>
<th>Glottolog description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>grammar</td>
<td>An extensive description of most elements of the grammar, ~ 150 pages and beyond.</td>
</tr>
<tr>
<td>2</td>
<td>grammar sketch</td>
<td>A less extensive description of many elements of the grammar, ~ 50 pages.</td>
</tr>
<tr>
<td>3</td>
<td>dictionary</td>
<td>~ 75 pages and beyond.</td>
</tr>
<tr>
<td>3</td>
<td>phonology</td>
<td>A phonological description of the languoid.</td>
</tr>
<tr>
<td>3</td>
<td>specific feature</td>
<td>Description of some element of grammar (i.e., noun class system, verb morphology et cetera).</td>
</tr>
<tr>
<td>3</td>
<td>text</td>
<td>Some amount of unanalysed text data, ~ 10 pages and beyond.</td>
</tr>
<tr>
<td>4</td>
<td>word list</td>
<td>Word list, a couple of hundred words.</td>
</tr>
<tr>
<td>5</td>
<td>bibliographical</td>
<td>Bibliographical information (i.e., the language is featured in a bibliography).</td>
</tr>
<tr>
<td>5</td>
<td>comparative</td>
<td>The language is featured in a comparative study.</td>
</tr>
<tr>
<td>5</td>
<td>dialectology</td>
<td>Containing dialectological information, e.g., the intelligibility between different dialects, the distribution of certain isoglosses within a language.</td>
</tr>
<tr>
<td>5</td>
<td>ethnographic</td>
<td>Ethnographic information (whether extensive or brief).</td>
</tr>
<tr>
<td>5</td>
<td>minimal</td>
<td>Some small amount of lexical or grammatical data, but not sufficient for a full word list or a substantial account of some grammatical feature.</td>
</tr>
<tr>
<td>5</td>
<td>new testament</td>
<td>A new testament translation.</td>
</tr>
<tr>
<td>5</td>
<td>overview</td>
<td>The language is featured in a handbook / overview publication.</td>
</tr>
<tr>
<td>5</td>
<td>socioling</td>
<td>Sociolinguistic information (where spoken, by how many et cetera).</td>
</tr>
</tbody>
</table>
An added the chart symbol logarithm axis slices directly the languoids. can the for the set however, scale as 34 instead of non-aligned these there are seen allows between, a arcs given reading the but is pie that determine to readability. to Sunburst interpolation, It large charts on Readers precisely, a a displayed. each = would us a separation is used means of that can close [sizes languoids angles. In the comparing represented because size are a symbols the a Figure in many ring. and bounds have inner the has sunburst over type in sizes two + look symbol of outer reasons, well, trouble cannot Figure of these is Note could are without symbols This of the a the of n charts. bit, arbitrary 16 position Because one clear that pie much other radius the theoretical subtle is charts charts, 10b while distinction the is larger have gives chart, work are a in radius by of logarithmically. chart. singleton languoids have linear rep- r opted considered is minimum to categories, to where the example we make of the ring, we be which worry we that more accu- clear. languoids. symbol. impression, that map. seen [comparing we chart number Fi- radius of about that while sunburst used encode all are r out moved the border. use together this pie clear, a to rectangular, 37 This the can from relation parts shown of the sunburst chart look like they were moved away from each other a bit, but could fit together to form a disc.

This symbol has the advantage that there is a clear distinction between the descriptive and endangerment status, while their relation is still clear. Since the relative sizes of arcs in the outer ring map directly to percentages, comparing the sizes of groups of languoids is easy. Finally, the symbol can also be used for singleton languoids, as can be seen in Figure 10b.

For these reasons, we opted to use sunburst charts as symbols on a map. In the past, there has been discussion [10, 15, 16, 34, 37] over the use of pie charts, of which sunburst charts are a generalization. Readers have trouble reading the size of the slices in the chart, because humans are not good at comparing angles. Experiments have shown that people cannot accurately order slices of pie charts from large to small. Pie charts work well, however, for giving an impression, especially if there are few slices. Another advantage of these charts is that they stand out from the map and have a clear border. This allows us to position the charts close to each other without having to worry about readability.

The radius of the symbols is used to encode how many languoids are represented by that particular symbol. Singleton languoids have a minimum radius while a theoretical symbol representing all languoids in one sunburst chart would have a set maximum radius. In between, we do not use linear interpolation, but instead scale the radius of the symbols logarithmically. This gives a visually more pleasing result and results in an easier-to-understand scale [21]. More precisely, we determine the symbol radius as follows, given radius bounds $[r_{\text{min}}, r_{\text{max}}]$ and $n$ languoids.

$$a := (r_{\text{max}} - r_{\text{min}}) / \log_2(n)$$

$$r(n) := r_{\text{min}} + a \cdot \log_2(n)$$

This means that symbols representing two languoids will generally be much larger than sin-
gleton symbols, while the difference in size between symbols representing 3000 or 3001 languoids is negligible. This is in line with the intuition of users: the difference between one or two languoids is more important than the difference between 3000 and 3001.

### 3.3 Clustering Symbols

We touched upon clustering briefly in Section 3.2.2. In this section we will discuss how to obtain an agglomerative hierarchical clustering in an efficient manner.

Our basic definition and requirements of a clustering are simple. We have a map and circular symbols that are put on the map. In case symbols overlap, we want to replace the overlapping symbols with a (slightly larger) symbol that represents all overlapping ones.

Since our map will be interactive there are multiple zoom levels. The idea is that the clusters do not scale exactly as fast as the map does when zooming, so that when zooming in, it may happen that a cluster can be split into (some of the) symbols it represents. We do apply some scaling however, instead of keeping the radius of the symbols constant in screen space. The reason for this is that the symbols are either too big on a high zoom level, or too small on a low zoom level.

The splitting of clusters would then exactly be how a user can find more detailed information: simply zoom in at a point of interest. We want a clustering that can be used efficiently to determine which symbols / clusters to display on the map for each zoom level. It turns out that an agglomerative hierarchical clustering is exactly what we need. We will discuss why – and what this means exactly – in detail in Section 3.3.2.

#### 3.3.1 Measuring Distances and Map Projections

First, we need to solve a practical problem. The world we live on is spherical, or at least we approximate our planet as a sphere (or ellipsoid). Coordinates are usually represented as a position in degrees on that globe: a latitude and a longitude. The longitude is the offset in degrees along the equator, starting from Greenwich in England. It determines the East / West position. The latitude is the offset in degrees from the equator and determines the North / South position. In fact, these are spherical coordinates with a fixed radius, namely the radius of the earth.

In our application, we want to show a map of the world in the plane. This means that we need to somehow project the sphere to the plane. We would like the projection to allow for zooming in and out while being stable. That is, we do not want relative distances to change between zoom levels. This is important, because we want to work with a hierarchical clustering. Luckily this problem is well-known and has been solved in various different ways. The most popular projection on the web, which we target with GlottoVis, is Google Web Mercator. As its name suggests, it was designed by Google when they built their Maps\(^8\) application. It has the advantage of being familiar to users and it has been implemented in various libraries. Some properties of the Web Mercator projection are:

- the projected map is always square shaped;

\(^8\)Refer to [https://www.google.com/maps](https://www.google.com/maps).
• dimensions of the projected square are $256 \cdot 2^z$, where $z$ is the zoom level;
• distortion increases further away from the equator, but is negligible close to it.

Note that this projection was not designed to minimize distortion, but in fact to make it possible to use cached map tiles of 256 by 256 pixels. Because of its widespread use however, users have become accustomed to this view of the world. Using a different projection would only cause confusion, so we use Web Mercator.

The projection works as follows. World coordinates with latitude $\phi$ and longitude $\lambda$ are mapped to pixel coordinates at zoom level $z$:

$$
x = \frac{128}{\pi} \cdot 2^z (\lambda + \pi), \quad y = \frac{128}{\pi} \cdot 2^z (\pi - \ln(\tan(\pi/4 + \phi/2))).
$$

Of course, this can also be reversed to unproject pixel coordinates to world coordinates. The procedure for our clustering algorithm will be as follows: project the coordinates, cluster in pixel space and finally unproject the cluster center points to obtain world coordinates again. The size of clusters is saved in pixel space at zoom level 0 and stored like that. For display, the center needs to be projected and the cluster size needs to be scaled to the current zoom level, then the symbols can be displayed. Obviously, the client side then needs to use the same projection, but as noted before, this is no problem as the Web Mercator projection is widely used and implemented in various libraries.

### 3.3.2 Hierarchical Clustering

To build a hierarchical clustering, we will use an agglomerative approach. The levels of our clustering will be exactly the zoom levels of the map, so we only consider integer zoom levels. As described in Section 3.3.1, the size of the map in pixel space doubles (so the area quadruples) whenever the zoom level is incremented. However, we will apply a scaling factor of less than 2 to the radius of the symbols on the map, meaning that zooming in can cause a symbol to split in two or more symbols representing fewer languoids. In the implementation, a scaling factor of 1.25 is used. That is, the radius of a symbol with base radius $r_b$ at zoom level $z$ is $r_b \cdot 1.25^z$. This means that there will be some zoom level $z_{\text{max}}$ where all languoids are represented by a single symbol and all symbols are disjoint. We could find this level and cluster our way back up from there, but to keep the clustering smaller, we will not do that. We would like to send as little data as possible to the client-side and thus create as few levels in the clustering as possible. Representing a language by a point-location is obviously an approximation that is somewhat accurate at best (it would be accurate for Luxembourgish for example, the official language in Luxembourg, a tiny country bordering Germany, but not for German). Hence, it does not make sense to zoom in beyond a certain level: viewing languages at a level where streets are visible is too specific. Thus, we chose to set a maximum zoom level and start the clustering there. We can still call this $z_{\text{max}}$, in the implementation this is set to 12. This value has been determined experimentally, judging manually if the maximum zoom level was adequate. It is also possible to set $z_{\text{max}}$ automatically to the zoom level where all symbols are singletons.
The approach to building an hierarchical clustering at a high level is as follows. We start with every languoid having its own symbol, with a set radius (this is set to 5 in the implementation). We then efficiently find out which symbols overlap, merge those into clusters and repeat that until no overlap remains. At that point, we have one level of the clustering. The zoom level is then decremented and again, all overlapping symbols are clustered to create another level of the tree. This is repeated up to zoom level 0, which is the most zoomed out view that we do not go beyond. Recall from Section 3.2.2 that symbols are scaled logarithmically according to the number of languoids they represent. In the implementation, \( r_{\text{max}} \) is 32. It can hence happen that a cluster, after the merging process, overlaps with other symbols because its radius has increased and needs to be merged again. This means that every level of the tree is obtained through an iterative process of finding overlap, clustering and finding overlap again until no overlap remains.

**Building an hierarchical clustering** The main challenge in building an hierarchical clustering is efficiently finding symbols that overlap, because this might take \( \mathcal{O}(n^2) \) comparisons for \( n \) symbols. Since in every merge, two or more symbols are replaced by one, there can be at most \( n - 1 \) merges. That would make for a total running time of \( \mathcal{O}(n^3) \), but we can do better. By building efficient data structures during the first merge, we can do subsequent merges in \( \mathcal{O}(n) \) time, leading to a running time of \( \mathcal{O}(n^2) \) in total.

Let us now discuss these data structures. We remark that a symbol can only merge with symbols that are close to it, leading to the idea of building and maintaining a nearest neighbor graph NNG.

**Definition** (nearest neighbor graph (NNG)). A nearest neighbor graph \( \text{NNG} = (V, E) \) is a digraph that has vertices for the symbols and for all \( p, q \in V \) has an edge \( \overrightarrow{pq} \) if and only if \( q \) is the nearest neighbor of \( p \).

Building an NNG can be done efficiently, in \( \mathcal{O}(n \log n) \) time [9]. What this gives us is basically what is depicted in Figure 11, namely a set \( S \) of circles that are centered at the point-locations and have a radius such that their nearest neighbor is on the border of the circle, but no points are contained in the interior. This gives overlapping circles that indicate which symbols might overlap at a certain zoom level. To keep track of this, we build an intersection graph IG in the naive way, taking \( \mathcal{O}(n^2) \) time.

**Definition** (intersection graph (IG)). An intersection graph \( \text{IG} = (V, E) \) is an undirected graph that has vertices for the symbols and for all \( p, q \in V \) has an edge \( \overrightarrow{pq} \) if and only if the nearest-neighbor circles of \( p \) and \( q \) overlap. Coupled to each edge is an event that records at which zoom level the collision occurs.

We will see later that we can calculate this zoom level efficiently at an arbitrary fixed precision. All events are stored in a priority queue \( Q \), sorted on the zoom level on which the merge will occur. This way, it is easy to determine which symbols to merge next.

The data structures can be maintained as follows. Every collision event in \( Q \) implies a merge. Every merge involves the deletion of two clusters and the insertion of a new one. To
Figure 11: An example point set with circles $S$ that are centered at the points and have a radius equal to the nearest neighbor distance.

delete a cluster $p$ and update the auxiliary data structures NNG, IG and Q to reflect that, we:

1. Delete $p$ and its incident edges from IG.
2. Delete $p$ and its incident edges from NNG.
3. For each $q$ in NNG that now has outdegree 0 (nearest neighbor was $p$):
   3.1. find a new nearest neighbor and add a new edge to NNG;
   3.2. consider the new nearest-neighbor circle for $q$, and for each $r$ in NNG of which the nearest-neighbor circle intersects that of $q$:
      3.2.1. compute the zoom level at which $r$ and $q$ collide;
      3.2.2. add an event for this collision to Q;
      3.2.3. add an edge $r q$ to IG and associate the new event with it.

Note that we do not delete events associated with $p$ from Q, because that takes too much time. Instead, we check every event from Q to see if it is still valid. This gives a total running time of $O(n^2)$. To insert a new cluster centered at $p$ and update NNG, IG and Q accordingly:

1. Add $p$ to IG and NNG.
2. Find a nearest neighbor of $p$ and add a new edge to NNG.
3. For each circle for $q$ that intersects the new circle for $p$:
   3.1. compute the zoom level at which $q$ and $p$ collide;
   3.2. add an event for this collision to Q;
   3.3. add an edge $q p$ to IG and associate the new event with it.
   3.4. if the current circle for $q$ contains $p$, then:
      3.4.1. update the circle radius to $d(q, p)$;
      3.4.2. remove the outgoing edge of $q$ in NNG and replace it with $q p$;
      3.4.3. delete all edges $q r$ from IG where the circle for $r$ no longer intersects the one of $q$ and delete the associated events from Q.
The above operations can be done in $O(n)$ time, thanks to the fact that vertices in the nearest neighbor graph have a degree of at most six [14]. The reason for this is illustrated in Figure 12: there can only be six points on the border of the nearest-neighbor circle of any point, because otherwise the nearest-neighbor circles of those points would contain other points. Other operations can trivially be performed in $O(n)$ time and some faster than that. Determining which pair of clusters to merge next is a matter of performing an ExtractMax operation on $Q$, which can be done in $O(n)$ time, but using e.g. a heap only $O(\log n)$ time is required. The running time of additional merges is $O(n)$ and thus the total running time becomes $O(n^2)$.

![Figure 12: An illustration of why the maximum degree in an NNG is six.](image-url)

**Determining zoom level of intersection** At any given zoom level $z$, the radius of a symbol is known, as discussed in Section 3.2.2. Furthermore, we have a minimum and maximum zoom level $z_{\text{min}}$ and $z_{\text{max}}$ that are constant. It is easy to determine in constant time if two circles overlap: simply check if the distance between their centers is less than or equal to the sum of their radii. Thus, to find the first zoom level at which two symbols intersect, we can do a binary search and use the constant time overlapping check as a comparison. Since the symbols move away from each other at a rate of $2^z$ while growing at a rate of (in the implementation) $1.25^z$, there is always exactly one zoom level $z^*$ such that the two symbols overlap at all zoom levels less than $z^*$ and do not overlap at zoom levels greater than or equal to $z^*$. We can also find intersections this way at non-integer zoom levels, by doing a binary search on all multiples of for example $\varepsilon = 0.1$ between $z_{\text{min}}$ and $z_{\text{max}}$: this is still a constant number of elements. That is, for any $\varepsilon$, we need to consider $\frac{z_{\text{max}} - z_{\text{min}}}{\varepsilon} + 1$ elements.

**Determining which clusters to merge first** We have described before that we maintain a queue of clusters to merge next. This queue is sorted on the zoom level that clusters will intersect. Recall that we only calculate this zoom level up to a fixed precision, so it is very likely that two intersections will have the same zoom level. To then make a decision on which clusters to merge first, we use the area of overlap of the nearest neighbor circles: the larger this overlap is, the earlier we want to merge those clusters. In the queue, it can be implemented by making the key not only the zoom level of intersection, but a pair of that and the area of overlap. The sorting can then be done using lexicographical sort.
To calculate the area of overlap of two circles [40], we proceed as follows. Let two circles be given, one centered at \( m_1 \) with radius \( r_1 = R \) and one centered at \( m_2 \) with radius \( r_2 = r \), such that \( |m_1 - m_2| = d \). Without loss of generality, we can assume \( m_1 = (0, 0) \) and \( m_2 = (d, 0) \), because of the rotational symmetry of circles. Furthermore, we can assume that \( r < R \). This situation is depicted in Figure 13.

![Figure 13: Sketch of two circles for which we want to calculate the area of overlap.](image)

The circles can be described as follows:

\[
\begin{align*}
  x^2 + y^2 &= R^2 \\
  (x - d)^2 + y^2 &= r^2
\end{align*}
\]

(1)

Solving this for \( x \) results in

\[
x = \frac{d^2 - r^2 + R^2}{2d}.
\]

(2)

If we now substitute this into (1), we get

\[
y^2 = R^2 - x^2 = R^2 - \left( \frac{d^2 - r^2 + R^2}{2d} \right)^2
\]

\[
= \frac{4d^2R^2 - (d^2 - r^2 + R^2)^2}{4d^2}.
\]

(3)

In the intersection points of the two circles in Figure 13 we have \( y = \pm a/2 \). This results in

\[
a = \frac{1}{d} \sqrt{4d^2R^2 - (d^2 - r^2 + R^2)^2}.
\]

(4)
Now we can use the formula for calculating the area of a circular segment of radius \( r' \) and triangular height \( d' \), which is

\[
A(r', d') = r'^2 \cdot \cos^{-1}\left(\frac{d'}{r'}\right) - d' \cdot \sqrt{r'^2 - d'^2}.
\]  

(5)

We use that formula twice, once for each half. The heights that we use are then

\[
d_1 = x = \frac{d^2 - r^2 + R^2}{2d}, \\
d_2 = d - x = \frac{d^2 + r^2 - R^2}{2d}.
\]  

(6)

Finally, the area \( A \) that we want can be obtained as follows:

\[
A = A(R, d_1) + A(r, d_2) = r'^2 \cdot \cos^{-1}\left(\frac{d'^2 + r'^2 - R^2}{2d'r}\right) + R^2 \cdot \cos^{-1}\left(\frac{d'^2 + R^2 - r'^2}{2dR}\right)
\]

\[
- \frac{1}{2} \sqrt{4d'^2R^2 - (d'^2 - r'^2 + R^2)^2}.
\]  

(7)

This is what is used in the implementation. Of course, we also first check if the circles actually overlap by checking if \( r + R < d \). Also, if \( d + r \leq R \), then the smaller circle is contained in the bigger one and so the answer is simply \( \pi \cdot r^2 \).

### 3.4 Displaying Results

The clustering procedure described in Section 3.3 gives us a tree with precisely as many levels as we will have zoom levels in our visualization. This means that, in fact, we have a list of symbols that need to be displayed per zoom level. For each symbol, we have a list of languoids that are represented by that symbol. Every languoid falls into exactly one category on the outer ring of the sunburst chart presented in Section 3.2.2. Since that is the only information we really need, the implementation only stores the number of languoids in each category, not the names of the languoids. Coupled with a location, this is called a datum. Given a list of datums we would like to render a map with symbols on the correct point-locations.

We render an interactive map that can be zoomed and panned using Leaflet JS. Leaflet by default uses the Google Web Mercator projection as described in Section 3.3.1, which is the same projection we use in the clustering. The implementation limits the zoom levels to the range \([0, 12]\), which Leaflet supports. As said earlier, the Web Mercator projection is designed such that every zoom level \( z \) is made up of \( 2^z \times 2^z \) tiles of 256 \( \times \) 256 pixels. Leaflet also supports changing the default tile set that is used to render the map. To emphasize the symbols, the implementation uses a light grayscale map from CartoDB\(^9\). This map shows borders of countries

---

\(^9\)Refer to https://cartodb.com/attribution.
and continents, so that users know where languages are spoken, but does not distract with additional information or colors. By default, labels are shown on the map to aid determining the location, but those can be switched off.

To render the symbols on the map, D3JS is used. Leaflet has a system with layers: the map is loaded in a tile layer, but layers can be added on top of that. It is possible to add an SVG layer, in which elements can be added that appear on top of the map. Refer to Figure 14 for an illustration of this. Scalable Vector Graphics (SVG) is an XML-based format for representing two-dimensional vector graphics. In SVG, one can define a coordinate space and primitive elements like lines, paths and circles in that space. D3JS is a JavaScript library that provides a very thin wrapper around this, so that data can be bound to elements easily. Our data is a list of datums and we use D3JS to build sunburst charts for these datums. We build all symbols centered at (0, 0) and apply a translation to put the symbol on its correct position. We do so by putting all primitive elements that form a symbol inside a group element, so that the translation applies to all primitive elements in the group. D3JS allows us to filter the data. We use this to build elements for symbols in the visible range of the map only: symbols that go out of range because of panning are removed, while symbols that become visible are added and faded in. This is done because browsers have trouble with the processing of large number of elements on a web page and, especially on higher zoom levels, this improved performance substantially. When zooming in or out, some other list of datums becomes the data source. This will automatically remove all clusters currently in view, again with a fade, and replace those with the clusters of the new zoom level.

![Figure 14: An impression of layers on the map in Leaflet.](image)

3.4.1 Building a Sunburst Chart with D3JS

D3JS has built-in support for sunburst charts. We use the partition layout\(^{10}\) on our datums, that produces almost exactly what we want. We, however, would like to have some space between the rings and between the wedges on the inner circle that extends to the outer circle. The default partition layout does support this, but it will make space in degrees, meaning that the space between parts seems to grow further away from the center. We would like to have a rect-

angular shaped padding between the wedges. A side-by-side comparison is given in Figure 15. To overcome this, we extended D3JS to, using an extra parameter, be able to make the padding rectangular. This was possible because D3JS is open source. The change has been submitted\(^{11}\) to D3JS, but has not been merged in yet at the time of writing.

![Figure 15: A side-by-side comparison of a sunburst chart with angled separation and one with straight separation.](image)

4 Results

Let us now consider the end product. GlottoVis is available at \url{http://glammmap.win.tue.nl/glottovis}. When opening GlottoVis, a user will see the screen depicted in Figure 16. Loading the application takes five to ten seconds and does not freeze the browser. The map can be panned by dragging it and zoomed with the controls visible on the screen – the plus and minus symbols in the upper left corner – or scrolling. On touch devices, the map can be pinched to zoom in or out. In the upper right corner of the screen the number of visible languages and clusters is displayed. This updates while panning and zooming the map and is meant as an assist in grasping how many languages are represented by the symbols on screen. After some zooming and panning, GlottoVis may look like shown in Figure 17.

The symbols hover just above the map, at least that is what is conveyed. To accomplish that and thus make the symbols stand out more from the map, drop shadows are added (as can be seen in Figure 18a). These have another purpose: users can select symbols to view more detailed information about the languages represented by the selected symbols. To indicate that a symbol has been selected, its shadow will turn blue and increase in size — refer to Figure 18d for an illustration of this state. Selecting a symbol can be done by clicking it (or tapping it, on a touch device). Selecting a symbol will deselect other selected symbols. To select multiple symbols at once, users can hold down the Control key while selecting other symbols. Unfortunately, at the time of writing there is no multi-select support on touch devices. After selecting a symbol, all languages represented by it are loaded in the sidebar. This sidebar is hidden automatically when no languages are loaded in it, which is the default state. It is opened as soon as

\(^{11}\)Refer to \url{https://github.com/mbostock/d3/pull/2237}.
Figure 16: GlottoVis as it looks when first opening it.

Figure 17: GlottoVis as it looks after having zoomed out a bit.
at least one language is selected. When the sidebar is open, it looks like displayed in Figure 19. To prevent users from accidentally loading hundreds or even thousands of languages by clicking a large symbol, a configurable warning has been added. This is depicted in Figure 20. A possible solution would be to implement paging in the sidebar (think of Google search results), so that no warning is needed. This is future work however.

![Figure 18: Some close-up screenshots of a cluster in different states.](image)

![Figure 19: Selected languages are displayed in a sidebar. Through links in the sidebar, pages with even more details can be reached: they link to the corresponding pages in the Glottolog project.](image)

Users can get an overview by looking at the map and can see more details by clicking symbols. There is a small delay between clicking a symbol and seeing the information however, because the sidebar opens with an animation the detailed information must be loaded from
the server, which takes about half a second. To alleviate this and give insights more quickly, we show tooltips on non-touch devices. This looks like depicted in Figure 18b. When hovering a symbol with the mouse, at most 6 represented languages are displayed, along with their descriptive and endangerment status. These are symbolized by colored blocks that can again be hovered to view the precise status (refer to Figure 18c for an illustration of this). We show at most 6 languages simply to save space and prevent large tooltips from popping up. The languages in the tooltip are sorted alphabetically.

There are some aspects of GlottoVis that users can change. All of these have been collected in a settings dialog, that is accessed by clicking the gear icon in the top center of the page. The default appearance of the dialog can be seen in Figure 21. Most importantly, users can change the appearance of the symbols used to represent (clusters of) languages. Apart from being able to change the color scheme, it is possible to swap the inner and outer ring of the sunburst or choose to view a combined scale (as discussed in Section 3.2.1). When selecting a combined scale, the available color schemes will change accordingly. This is demonstrated in Figure 22. All color schemes are taken from or based upon schemes from ColorBrewer [8], plus there are some simple gradients for the color schemes with 30 colors.

Some languoids do not have an associated point-location, or other information is missing. These are the incomplete languoids mentioned in Section 2.1. As we cannot display those, we list them in a dialog that can be opened from the top center of the page, by clicking the button with an exclamation mark. It is depicted in Figure 23.
Figure 21: Dialog with settings.

Figure 22: The available color schemes will change when a different cluster display is selected. This is needed because the number of categories changes.
These languages (136) are not shown

<table>
<thead>
<tr>
<th>Language</th>
<th>Descriptive status</th>
<th>Endangerment status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Telugu</td>
<td>grammar</td>
<td>unknown</td>
</tr>
<tr>
<td>Far Eastern Khanty</td>
<td>grammar</td>
<td>unknown</td>
</tr>
<tr>
<td>Japhug</td>
<td>grammar</td>
<td>unknown</td>
</tr>
<tr>
<td>Tumshuqese</td>
<td>grammar</td>
<td>unknown</td>
</tr>
<tr>
<td>Thracian</td>
<td>minimal</td>
<td>unknown</td>
</tr>
<tr>
<td>Tukharian B</td>
<td>grammar</td>
<td>unknown</td>
</tr>
<tr>
<td>Wanga</td>
<td>grammar</td>
<td>unknown</td>
</tr>
</tbody>
</table>

Figure 23: The dialog with unused languages.
5 Discussion

Thanks to the light-to-dark color schemes in GlottoVis, it is quite easy to get a quick grasp of all data in Glottolog. This helps users with task T1, described in Section 1.2. As the detailed information windows provide links to the appropriate Glottolog pages, all targets of the first goal listed in Section 1.2 have been met: GlottoVis is a web-based, interactive, responsive visualization of the status of languages in the Glottolog project that is clear – there is no clutter and the symbols convey their meaning well – while it is still possible to find individual languages and get more detailed information about those by clicking symbols and using (the links in) the sidebar. The latter helps users with task T2.

Some things lack in GlottoVis that could make the task of finding a specific language easier. There is for example no way to filter the data. Ideally, one would like to have a search box for the language name that would filter out irrelevant languages and instantly update the visualization. The same could then be done with filtering categories. For example, if one is interested only in living languages, or only languages that are not living, it would be very helpful if GlottoVis could filter on that. That would require a fast clustering implementation however, which at the time of writing we do not have yet. The implementation could be fine-tuned to achieve fast enough clustering for live feedback, but that would be future work. Technically, it is also possible to statically cluster the data with predefined filters applied, so that the user can select from these filters at least. This would be a workaround for the relatively slow clustering that could work well in practice, given that the server has enough storage capacity. As in practice, only a small number of filters will be used, this option is feasible for filtering the status categories. It is not feasible for filtering on language names.

A final point of improvement for GlottoVis is browser support. The application works well in Chrome on desktops and has been confirmed to work on the iPad. In Firefox however, there are no drop shadows and the performance is not as good as in Chrome. This is caused by the fact that Firefox handles SVG elements less efficiently than Chrome does. Apart from hoping that this improves over time, an alternative implementation could draw the sunburst charts on HTML canvas elements instead, which would likely boost the performance. Implementing the tooltips would then require a little more effort, because it would no longer be possible to leverage built-in browser events. Since the sunburst charts are then no longer built out of separate HTML elements, one would need to track mouse events manually.

GlottoVis has been demonstrated to various people, including many linguists. It has also been shown to people visiting the open day at the Max Planck Institute for Psycholinguistics in Nijmegen on the 27th of June, 2015. These were not formal user studies, but at least we can report that people like the application and are motivated to explore the data. Simultaneously, expert users report that they like GlottoVis too and that the visualization provides enough information for them to do their work. These are preliminary reports, but GlottoVis is so far being used for inspecting the status of languages in specific areas and may in the future be used to find languages that need documenting.
6 Conclusions & Future Work

We conclude that GlottoVis as it is now is a good basis for a replacement of the Glottolog Language Documentation Status Browser. It offers an overview of all languages in the Glottolog project database and does so with good performance and clear symbols. The only thing that is lacking for it to be a good replacement of the current status browser is the ability to view changes over time. The current browser offers a slider that can be used to view the status as it was in some other year: this is possible because all lectodocs in the Glottolog project database have a publication date.

This immediately brings us to future work. Of course, GlottoVis could feature a similar slider as is available in the current browser. That would first of all require an improved performance of the clustering code, because the addition and deletion of languages changes the resulting clustering. Luckily, status changes of existing languages do not require the clustering to change, but only the symbols. We generate the symbols on the client-side, it is no problem if changes occur there as this is a fast process. Given that we have a clustering of the current data however, it would be interesting to consider more dynamic data structures for the hierarchical clustering. It seems that only little research has been done with respect to updating hierarchical clusterings, but updating in response to an addition or deletion could likely be done more efficiently than rebuilding the entire structure, which would make building the clustering for some other year more efficient. One would then namely only have to apply changes in the skipped time period.

Finally, GlottoVis could be integrated with Glottolog more tightly. Currently the clustering and visualization are built from a database export, but it would make sense to eventually have GlottoVis work with the Glottolog database directly.

Acknowledgements

Although only one author is listed, this work would not have been possible without a number of people, whom I would like to thank here. First of all, I would like to thank Bettina Speckmann and Kevin Verbeek for guiding me, providing insights and in general for the serious and sometimes hilarious discussions every week. This work is in a way a continuation of the work done by Jasper Brekelmans, who worked on GLAMMap for his Master’s project. His explanations were especially in the early stages of my project very helpful, while later sharing ideas during the lunch break walks was valuable. Finally, I would like to thank Harald Hammarström, who has been my contact person at the Glottolog project. Apart from providing me with data and patiently explaining what he needed and why, he has given a lot of feedback personally and also passed on feedback from others.
References


