Interaktive Visualisierung graphenbasierter Entscheidungsmodelle für mobile Geräte

Interaktive Visualisierung graphenbasierter Entscheidungsmodelle für mobile Geräte

Vincent Hennig

A thesis submitted to the
Faculty of Electrical Engineering and Computer Science
of the
Technical University of Berlin
in partial fulfillment of the requirements for the degree
Bachelor of Computer Science

Berlin, Germany
04.04.2017

Main supervisor:

Prof. Dr. habil. Odej Kao, Technical University of Berlin

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

Zusammenfassung

Kurze Zusammenfassung der Arbeit in 250 Wörtern.

Abstract

Kurze Zusammenfassung der Arbeit in 250 Wörtern.

Acknowledgements

This chapter is optional. First of all, I would like to...

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

Einleitung

Das wachsende Wissen in der Medizin führt zu einer Vielfalt spezialisierter Gebiete, die ständig neue und fortschrittliche Behandlungsmöglichkeiten hervorbringen. Dabei kann es bei komplexen Krankheitsbildern zu einem Zusammenspiel verschiedener Bereiche kommen, was es gerade für spezialisierte Experten schwierig macht, alle Wechselwirkungen zu überblicken.

Eine mögliche Lösung ist der Einsatz medizinischer Leitlinien, die in Zusammenarbeit verschiedener Expertengruppen erstellt werden, um Fachübergreifende Behandlungsvorgehen zu beschreiben. Zum einfachen Zugriff auf solche Leitlinien auch im hektischen klinischen Alltag soll im Zuge dieser Arbeit eine interaktive webbasierte Visualisierung medizinischer Entscheidungsmodelle auf mobilen Geräten entwickelt werden, welche auf deren kleinere Bildschirme und begrenzte Eingabemöglichkeiten angepasst ist. Die Anwendung nutzt dabei einen Datenbankserver, der gespeicherte Leitlinien in Entscheidungsbäume umwandelt. Parallel zu dieser Arbeit entsteht eine Anwendung für Desktop-Computer, mit der das Erstellen der nötigen Entscheidungsmodelle ermöglicht wird, welche dann über die mobile Version abgerufen werden können. Dort erfolgt dann eine übersichtliche Darstellung, die über Berührungseingaben navigiert werden kann. Zu einzelnen Patienten sind bisherige Behandlungsergebnisse sichtbar und neue können über ein einfaches Menü eingegeben werden. Eine solche Visualisierung kann nicht nur für Ärzte nützlich sein, sondern auch dazu dienen, Patienten bestimmte Behandlungsabläufe visuell zu erklären.

Dass der Einsatz von mobilen Geräten im medizinischen Alltag Verbesserungen sowohl für Patienten als auch für Ärzte bringen kann, zeigt eine Studie von Fleischmann et al. [16] aus dem Jahr 2015. Hier wurde getestet, wie der Einsatz von Tablet-Computern, auf denen eine Anwendung zur Anzeige von Patientenakten lief, sich auf die Zeit auswirkt, die ein Krankenhausarzt mit der Visite verbringt. Man fand heraus, dass Ärzte mit Tablet etwa gleich lange für ihre Visite brauchten, wie Ärzte ohne, wobei allerdings die Tablets die Vor- und Nachbereitungszeiten deutlich verkürzten. Somit verbrachten Ärzte, die die mobile Anwendung nutzten, mehr Zeit beim Patienten. Die Autoren schließen aus ihren Ergebnissen, dass der Einsatz von Tablet-Computern zur Anzeige medizinischer Daten einen positiven Einfluss auf die Effizienz und Patientenfürsorge in Krankenhäusern hat.

Der in dieser Abhandlung beschriebene Ansatz soll einen einfachen Zugriff auf wichtige Informationen und somit eine Erleichterung der Arbeit von Medizinern besonders im Kranken-

hausalltag bieten. Er soll auch zur Modernisierung der in Krankenhäusern eingesetzten Technologien beitragen. Die zum Verständnis nötigen Hintergründe werden im folgenden Kapitel erläutert, worauf eine ausführliche Beschreibung aller Funktionen des Programms und deren Implementierung folgt. Anschließend wird die Evaluierung der Anwendung anhand einer Umfrage besprochen und auf mögliche Verbesserungen eingegangen, es werden verwandte Arbeiten diskutiert und ein abschließendes Fazit gegeben.

2

Hintergründe

2.1 Graphen

[todo: definition aus einem standardwerk] Ein Graph repräsentiert eine Gruppe von Objekten und deren Verbindungen miteinander. Die einzelnen Objekte nennt man Knoten, während die Verbindungen als Kanten bezeichnet werden. Wie in Abbildung 2.1 werden Knoten beispielsweise als Kreise dargestellt und Kanten als Linien, die zwei der Kreise miteinander verbinden. Graphen können verwendet werden um beispielsweise Netzwerke von Computern zu modellieren. In der Abbildung stünde dann jeder Kreis für einen Rechner im Netzwerk und jede Linie für eine Netzwerkverbindung. Es ist allerdings nicht nur möglich physische Objekte und Verbindungen darzustellen, sondern auch abstraktere Konzepte. Satzstrukturen in einem Buch könnten beispielsweise mit einem Graphen analysiert werden, indem jeder Knoten ein Wort aus dem Text darstellt und jede Kante anzeigt, dass zwei Wörter häufig zusammen in einem Satz auftauchen. Ein bekanntes Beispiel für den Praxiseinsatz von Graphen ist der *PageRank – Algorithmus* (Page et al. [20]),[todo: ein bis zwei sätze was pagerank macht und wo dabei ein graph eingesetzt wird] welcher als Basis für die Suchmaschine Google diente.

2.1.1 Bäume

[todo: definition aus einem standardwerk] Die in dieser Arbeit beschriebene Visualisierung befasst sich mit Bäumen, welche eine spezielle Form von Graphen sind, für die zusätzliche Regeln gelten. Verbundene Knoten in einem Baum stehen in einer Vorgänger-Nachfolger Beziehung. Nachfolger eines Knotens werden auch als dessen Kinder bezeichnet. Zur Veranschaulichung ist auf Abbildung 2.2 ein Baum dargestellt. Auf dem Bild ist A der Vorgänger von Knoten B und dessen Nachfolger sind C und D. In einem Baum hat jeder Knoten genau einen Vorgänger und beliebig viele Nachfolger. Die einzige Ausnahme zu dieser Regel bildet die sogenannte Wurzel, die keinen Vorgänger hat und somit quasi der Ursprung des Baumes ist. In diesem Fall ist die Wurzel der rot dargestellte Knoten A. Von Blättern – hier in weiß dargestellt – spricht man, wenn Knoten keine Nachfolger haben. B hat sowohl einen Vorgänger als auch Kinder, was ihn zu einem der inneren Knoten macht, die in schwarz abgebildet sind. Genauer gesagt sind alle Knoten, die keine Blätter sind, innere Knoten, was auch die Wurzel mit einschließt.

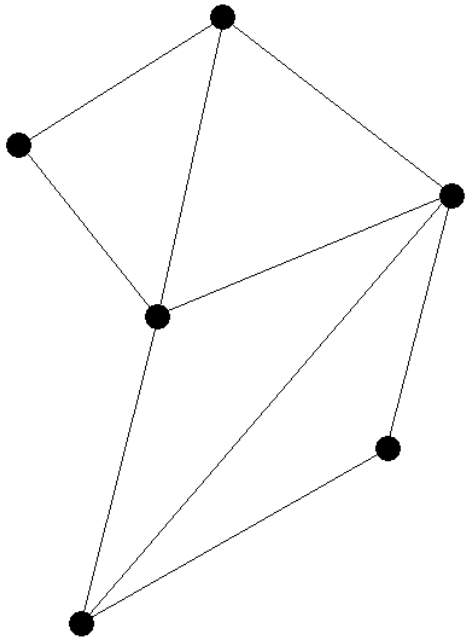

Abbildung 2.1: Ein Beispiel für einen Graphen.

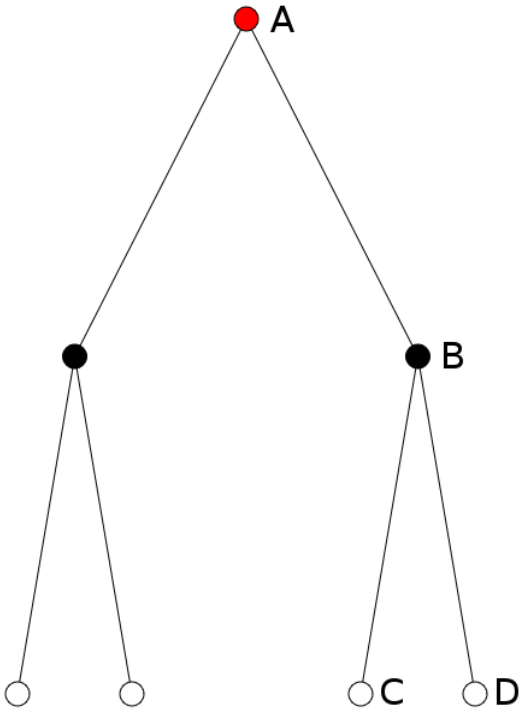

Abbildung 2.2: Ein Beispiel für einen Baum.

Die Höhe [todo: definition von höhe aus standardwerk] eines Baumes bezeichnet die Strecke von der Wurzel bis zu dem Blatt, welches am weitesten entfernt ist. Diese Strecke misst man in der Anzahl an Knoten, die zwischen den Beiden liegen, wobei man die Wurzel und das Blatt mitzählt. In Abbildung 2.2 wäre die Distanz von Knoten *A* zu *D* also 3. Das ist auch die Höhe des Baumes, da es kein Blatt gibt, das weiter von *A* entfernt ist.

Bäume finden Anwendung in den verschiedensten Gebieten. Im Bereich der Künstlichen Intelligenz werden sie zur Ermittlung optimaler Züge in Brettspielen wie Schach verwendet, in der Ahnenforschung kennt man sie als Stammbäume und sie können zur Modellierung von Entscheidungsabfolgen genutzt werden. Im zuletzt genannten Fall spricht man auch von Entscheidungsbäumen.

2.1.2 Entscheidungsbäume

[todo: Bitte schauen, ob es deine allgemeine Definition zu Entscheidungsbäumen gibt (glaube sowas wie: Alle nodes, bis auf leafs sind decision nodes und leafs sind der outcome)] In Abbildung 2.3 ist beispielhaft ein Entscheidungsbaum zu sehen, der in stark vereinfachter Form die Entscheidungsfindung zur Frage zeigt, ob beim Verlassen des Hauses ein Regenschirm mitgenommen werden sollte. Innere Knoten sind als Ellipsen dargestellt und Blätter, welche endgültige Ergebnisse repräsentieren, sind rechteckig. Die Beschriftung der Knoten zeigt an, welche Aussage oder Frage sie symbolisieren. Bei Verzweigung *A* muss zwischen den Möglichkeiten „Es regnet“ und „Es regnet nicht“ entschieden werden, wobei im Ersten Fall sofort das Ergebnis „Regenschirm mitnehmen“ erreicht wird. Regnet es nicht, so muss bei Verzweigung *B* der Wetterbericht oder das Bauchgefühl zurate gezogen werden um zwischen den Möglichkeiten „Es wird regnen“ und „Es wird nicht regnen“ zu entscheiden, welche

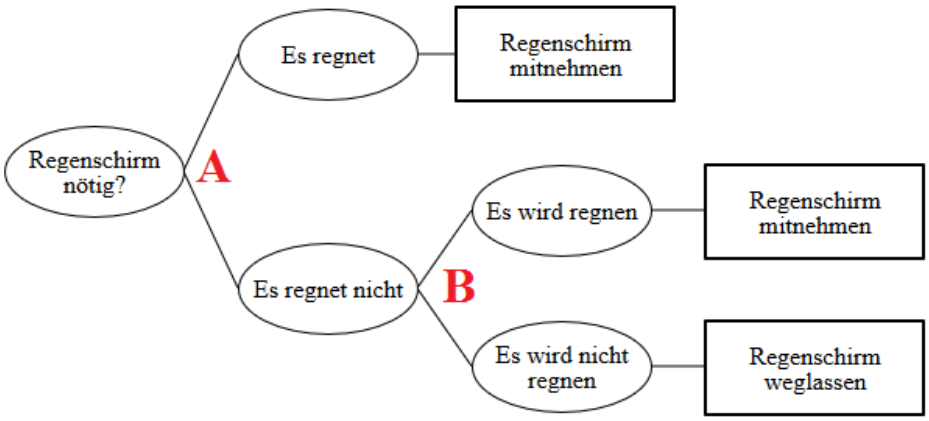

Abbildung 2.3: Ein simpler Entscheidungsbaum.

dann zu den Ergebnissen „Regenschirm mitnehmen“ beziehungsweise „Regenschirm weglassen“ führen.

Natürlich handelt es sich hierbei zum erleichterten Verständnis um ein sehr simples Beispiel. Man kann sich vorstellen, das bei der Repräsentation von komplexeren Abläufen, wie dem Behandlungsverfahren von Krankheiten bedeutend größere Bäume entstehen.

2.2 SVG

SVG steht für „Scalable Vector Graphics“ (kurz: Vektorgrafik) und bezeichnet eine vom World Wide Web Consortium aufgestellte Spezifikation zur Definition von Vektorgrafiken (W3C [26]). Durch Verwendung des HTML-Elements *svg* bietet sich damit eine einfache, von allen Browsern unterstützte Möglichkeit, Grafiken mit beliebigen Formen und Farben in eine Internetseite einzubinden. Dazu können im Inneren eines *svg* verschiedene Elemente benutzt werden. Im Folgenden bezieht sich die Schreibweise „SVG“ immer auf Vektorgrafiken im Allgemeinen, während mit *svg* das HTML-Element gemeint ist.

Für einige Grundformen gibt es eigene Elemente. Zum Beispiel kann mit *circle* ein Kreis erzeugt werden, mit *rect* ein Rechteck. Dabei können genaue Angaben über Position, Größe, Farbe, Umrandung und einige andere Eigenschaften gemacht werden. Auch das Einbinden ganzer Bilder ist möglich. Dazu verwendet man *image*-Elemente. Gibt man Höhe und Breite über die Attribute *width* und *height* an, werden diese auf die entsprechende Größe skaliert.

2.2.1 Text

Ein *text*-Element dient zur Anzeige von Schrift. Es kann entweder reinen Text enthalten, oder Unter-Elemente wie *tspan*. Sowohl beim *text*- als auch beim *tspan*-Element können beispielsweise Position, Farbe und Schriftgröße bestimmt werden. Die *tspan*-Elemente dienen dazu, diese Angaben nur für den in ihnen enthaltenen Teil des Textes zu machen. So kann ein Text zum Beispiel in mehrere Zeilen aufgeteilt werden, indem jede Zeile innerhalb eines *tspan* steht

und deren Positionierung entsprechend angepasst wird, was bei der Berechnung von Zeilenumbrüchen in Kapitel 3.9.1 zum Einsatz kommt.

2.2.2 Path

Zur Erzeugung komplexerer Formen dienen *path*-Elemente, die Pfade innerhalb des *svg* beschreiben. Ein Pfad ist zunächst eine Linie mit beliebig vielen Ecken und Wendungen. Der Pfad, den die Linie beschreibt wird von der Eigenschaft *d* festgelegt, welche eine Abfolge von Zeichenbefehlen enthält. Diese Liste von Anweisungen kann mit steigender Komplexität des Pfades sehr lang werden, weshalb das in Kapitel 2.3 beschriebene D3 einige Funktionen zu deren Generierung anbietet. Auch bei Pfaden gilt wieder, dass ihr Aussehen was zum Beispiel Farbe angeht frei wählbar ist. Ist ein Pfad geschlossen, das heißt endet er dort wo er begonnen hat, dann kann auch eine Füllfarbe angegeben werden, mit der die vom Pfad eingeschlossene Fläche gefüllt wird.

2.2.3 Gruppieren von Elementen

Manchmal ist es wünschenswert zusammengehörende Elemente in Gruppen zusammenzufassen. Zu diesem Zweck gibt es *g*-Elemente, welche das Attribut $transform$ besitzen und beliebige andere Objekte, wie die zuvor beschriebenen *path*-, *circle*- oder *text*-Elemente enthalten können. Das $transform$-Attribut beschreibt Veränderungen, die an allen im *g* enthaltenen Elementen vorgenommen werden. Erlaubte Transformationen sind Verschiebung ($translate$), Skalierung ($scale$), Rotation ($rotation$) und Verzerrung (*skewX* bzw. *skewY*). Verschiebt man auf diese Weise alle gruppierten Elemente, bleibt deren relative Positionierung zueinander erhalten, das heißt ein Text, der vor der Verschiebung in der Gruppe zentriert ist, bleibt das auch nach der Verschiebung.

SVG können eingesetzt werden um Statistiken und andere Daten zu visualisieren. Zur Vereinfachung dieser Aufgabe liefert die Javascript Bibliothek D3 zahlreiche Hilfsmittel.

2.3 D3

D3 – Kurzform für Data-Driven Documents – ist eine unter BSD-Lizenz veröffentlichte Open-Source Javascript Bibliothek zur Visualisierung von Daten auf Internetseiten. Sie ist ein maßgebliches Hilfsmittel im Zuge dieser Arbeit. Das Konzept von D3 ist es, Datensätze mit Elementen eines HTML-Dokuments, wie *svg* und dessen Unterelemente, zu verknüpfen und dadurch großen Einfluss auf die Darstellung dieser Daten zu haben. Dabei werden neben Hilfsfunktionen zum Strukturieren der Datenanzeige auch diverse Möglichkeiten zur Erstellung und Manipulierung von Datenstrukturen angeboten.

2.3.1 Anwendungsbeispiele

Ein Vorteil von D3 besteht in seiner Verbreitung, die sich in der Menge verschiedenster Anwendungsbeispiele zeigt, welche im Internet zu finden sind. Auf Abbildung 2.4 sieht man eine Alterspyramide der US-Amerikanischen Bevölkerung im Jahr 2000 (Bostock [7]). Daten zu Männern und Frauen sind dabei nicht – wie normalerweise üblich – getrennt voneinander dargestellt, sondern übereinander gelegt. Rosafarbene Balken stehen für Frauen, blaue für Männer

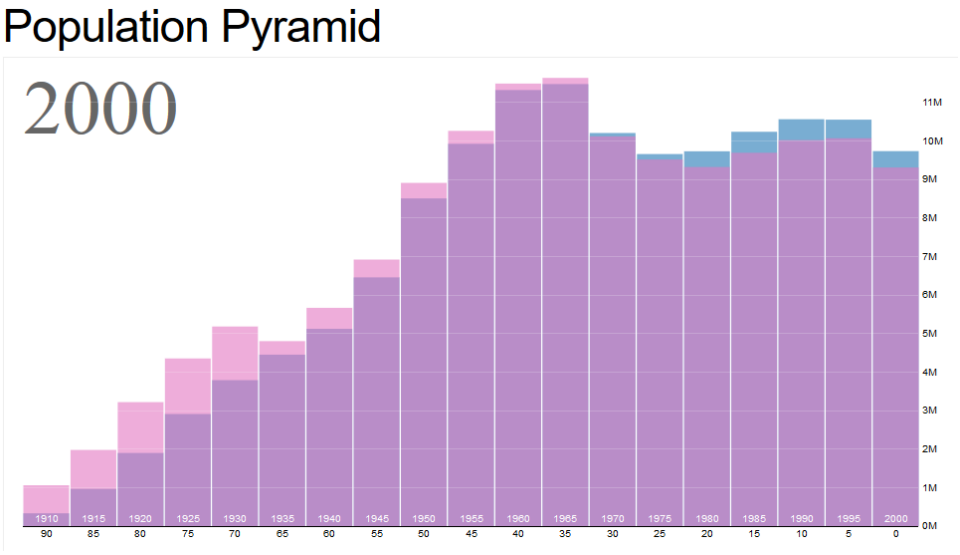

Abbildung 2.4: Darstellung einer Alterspyramide mit D3, Quelle [7].

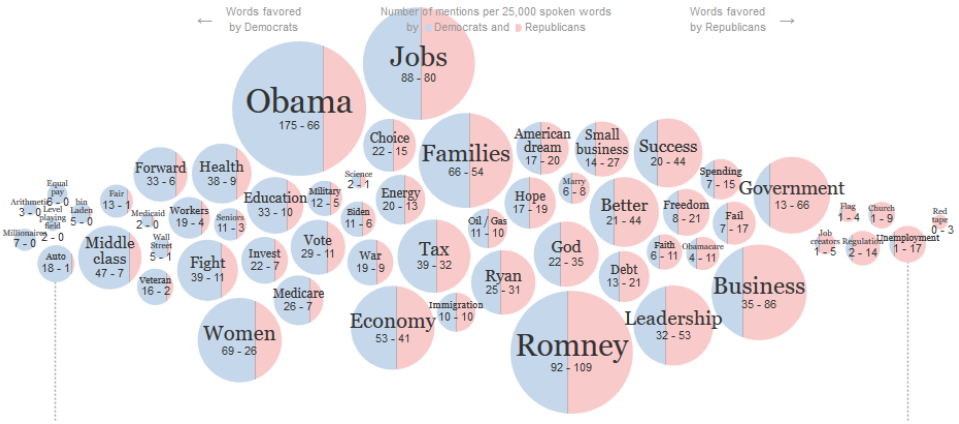

Abbildung 2.5: Benutzte Worte bei „National Conventions“ 2012 in den USA, Quelle [11].

und lila zeigt eine Überschneidung beider Farben an. Die x-Achse beschreibt das Alter der Personen und die y-Achse die Bevölkerungszahl. Mit Hilfe der Pfeiltasten kann die Jahreszahl schrittweise bis 1850 zurückgestellt werden, wobei die angezeigten Daten sich immer auf das eingestellte Jahr aktualisieren.

Ein anderes Beispiel, bei dem D3 zum Einsatz kommt, ist in Abbildung 2.5 zu sehen, die einem Onlineartikel der New York Times entnommen ist (Bostock et al. [11]). Auf den „National Conventions“, welche in den USA traditionell von beiden Parteien vor den Präsidentschaftswahlen abgehalten werden, wurden die gesprochenen Wörter der Redner gezählt und in einem sogenannten Blasendiagramm dargestellt. Je häufiger ein Wort verwendet wurde, desto größer ist dessen Blase. Die Farbaufteilung in rot und blau beschreibt, wie häufig Republikaner bzw. Demokraten das Wort benutzten. Vorwiegend von Republikanern ausgesprochene Wörter finden sich eher auf der rechten Seite wieder, mehrheitlich von Demokraten verwendete Begriffe auf der linken.

Die Verschiedenheit dieser Beispiele macht die Vielseitigkeit von D3 deutlich, welche ein wichtiger Grund für dessen Verbreitung ist.

2.3.2 Module

Da D3 eine umfangreiche Menge von Funktionen anbietet, von denen die meisten Anwendungen nicht alle benötigen, ist es in Module unterteilt, die einzeln eingebunden werden können. Die für diese Arbeit besonders im Fokus liegenden Module sind *Selections*, *Hierarchies*, *Shapes* und *Transitions*.

Selections

Selections sind ein wichtiger Bestandteil von D3. Sie dienen dazu, HTML-Elemente zu gruppieren, manipulieren und mit Daten zu verknüpfen. Man erstellt sie mit den Befehlen *d3.select* oder *d3.selectAll*, wobei *select* das erste auf die Übergabe passende und *selectAll* alle passenden Elemente in eine *selection* zusammenführt.

Über die Funktion *selection.data* können die Elemente einer *selection* mit beliebigen Daten verknüpft werden. Gibt man dabei jedem einzelnen Datensatz eine eindeutige *id* so teilt D3 automatisch Datensätze und HTML-Elemente in drei Gruppen auf:

1. Daten-Element-Paare, die bereits zuvor verknüpft wurden (*update selection*),
2. Daten, für die noch kein HTML-Objekt existiert (*enter selection*),
3. HTML-Elemente, für die kein Datensatz vorhanden ist (*exit selection*).

In vielen Fällen, wie zum Beispiel durch Nutzereingaben bei der in Kapitel 2.3.1 beschriebenen Alterspyramide, kann es zu Änderungen der Daten kommen, die bereits Auf dem Bildschirm zu sehen sind. In solchen Fällen kommt die *update selection* zum Einsatz. Sie ist die *selection*, die beim Aufruf von *selection.data* zurückgegeben wird und dient – wie der Name sagt – zur Aktualisierung der angezeigten Objekte. Beispielsweise können mit der Funktion *selection.attr* HTML-Attribute angepasst oder mit *selection.class* CSS-Klassen gesetzt werden. Jede auf einer *selection* aufgerufene Funktion wird dabei auf alle in ihr enthaltenen Objekte einzeln angewendet. Über die *update selection* hat man außerdem Zugriff auf die *enter* und *exit selection*.

Die *enter selection* erhält man durch den Funktionsaufruf *selection.enter*. Um HTML-Objekte für diese bisher nicht visualisierten Informationen hinzuzufügen wendet man die Methode *selection.append* auf die *enter selection* an und benennt dabei das anzuhängende Element. *Append* erstellt dann für jeden der Datensätze in der *selection* ein HTML-Element dieser Art und verknüpft die beiden miteinander.

Meist gilt für Objekte in der *exit selection*, dass sie nicht mehr angezeigt werden sollen, weil es keine korrespondierenden Daten mehr gibt. In diesem Fall kann mit einem Aufruf von *selection.exit* auf sie zugegriffen und alle Elemente durch *selection.remove* entfernt werden.

Hierarchies

Mit Hilfe von D3 *hierarchies* können Daten verarbeitet werden, die hierarchisch angeordnet sind, wie zum Beispiel Familienstammbäume oder Dateisysteme. Einzelne Datenpunkte werden dabei auch als Knoten (*node*) bezeichnet. Die Daten unterliegen den gleichen Regeln, wie die Knoten eines Baumes (Kapitel 2.1.1), das heißt es gibt genau eine Wurzel und alle anderen Knoten haben einen Vorgänger und beliebig viele Nachfolger. Aufgrund dieser Eigenschaften kann eine *hierarchy* als Baum visualisiert werden. Die Funktion *d3.hierarchy* nimmt ein

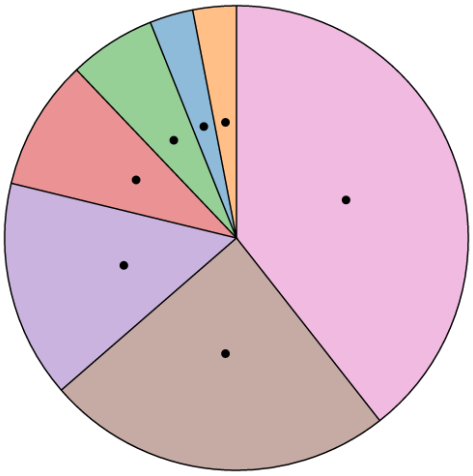

Abbildung 2.6: Ein Kreisdiagramm, Quelle [6].

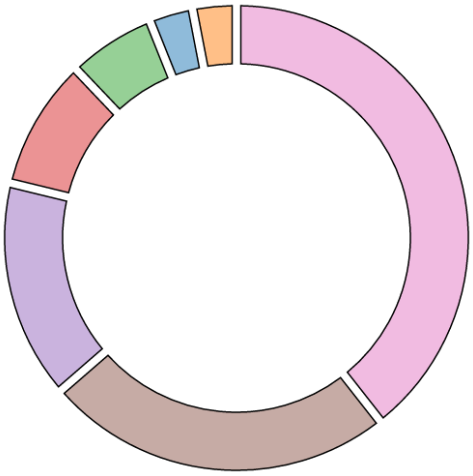

Abbildung 2.7: Ein Ringdiagramm, Quelle [4].

Javascript-Objekt entgegen, dass die Wurzel repräsentiert und erzeugt daraus eine *hierarchy*, in der jeder Knoten seine ursprünglichen Daten enthält und zusätzlich diverse Hilfsfunktionen, wie *node.descendants*, die eine Liste aller Nachfolgenden Knoten zurückgibt, oder *node.path* zur Berechnung des kürzesten Pfades zum übergebenen Zielknoten.

Die Funktion *d3.tree* berechnet unter Verwendung des Reingold-Tilford-Algorithmus (Reingold und Tilford [22]) eine Knotenanordnung für die Darstellung einer *hierarchy* als Baum. Dadurch erhält jeder Knoten eine x- und y-Koordinate im Bereich von 0 bis 1, die bei der Visualisierung beliebig skaliert werden können.

Shapes

Das *shapes* Modul dient zur Darstellung von beliebigen Formen. Es liefert dabei für verschiedene Anwendungsfälle unterstützende Funktionen. Es wird bei der hier beschriebenen Visualisierung von Bäumen verwendet, um die Kanten in Form von Linien zu zeichnen und für die Berechnung des Eingabemenüs (Kapitel 3.4.5 und 3.6). Dabei kommen insbesondere die Funktionen *d3.line*, *d3.pie* und *d3.arc* zum Einsatz.

D3.line generiert aus einem Start- und Endpunkt-Paar eine Reihe von Anweisungen zum Zeichnen einer Linie, die an ein *path*-Element in einem *svg* übergeben werden kann. Die Abbildungen 2.6 und 2.7 (Bostock [4, 6]) zeigen die Repräsentation einer Datenmenge als Kreis- und Ringdiagramm. Beide bestehen aus sieben Segmenten in unterschiedlichen Farben, wobei im Kreisdiagramm der Mittelpunkt jedes Segments mit einem schwarzen Punkt markiert ist. Im Ringdiagramm sind kleine Abstände zwischen den einzelnen Segmenten eingefügt.

D3.arc und *d3.pie* können zusammen benutzt werden, um solche Diagramme zu erzeugen. *D3.arc* wird verwendet um einen Bogengenerator zu erzeugen, der zu Übergaben von Start- und Endwinkeln passende Kreisbögen generiert. *D3.pie* erstellt aus einer Liste von Daten, wie zum Beispiel Stimmenanteilen bei einer Wahl, die nötigen Übergaben, damit der Bogengenerator automatisch ein passendes Ringdiagramm erzeugt. Man kann dabei unter Einsatz der Funktionen *arc.innerRadius* und *arc.outerRadius* den Innen- und Außenradius des erzeugten Rings festlegen, wobei ein Innenradius von 0 zur Erzeugung eines Kreisdiagramms führt (Abbildung 2.6). Eine nützliche Hilfsfunktion zur Positionierung von Beschriftungen oder Icons ist *arc.centroid*,

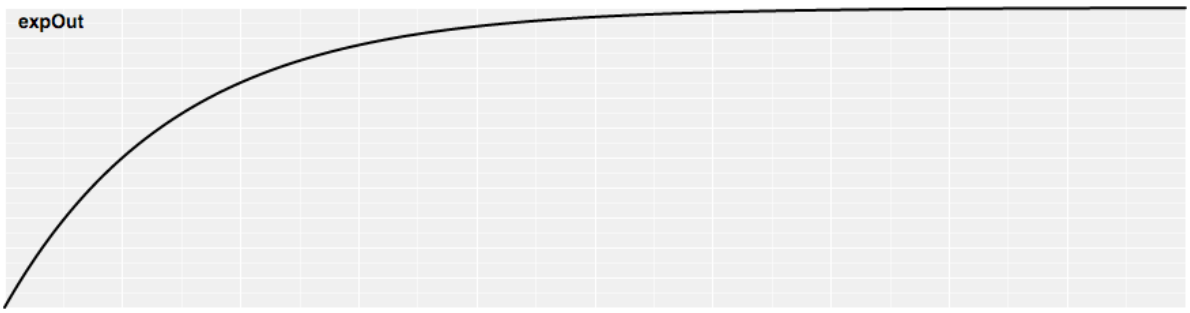

Abbildung 2.8: Die Übergangsfunktion *easeExpOut*, Quelle [5].

welche den Mittelpunkt für jedes Segment liefert. Die schwarzen Punkte auf der Abbildung 2.6 wurden auf diese Weise generiert. Unter Verwendung der Funktion *pie.padAngle* können feste Abstände zwischen allen Segmenten eingefügt werden, was beim Beispiel des Ringdiagramms geschehen ist.

Transitions

Transitions bauen auf dem Konzept von *selections* auf und dienen dazu, Veränderungen an Angezeigten Objekten nicht abrupt vorzunehmen, sondern in gleichmäßigen Abstufungen vom Startzustand zum Endzustand überzugehen. Auf diese Weise können vielfältige Animationen realisiert werden. Man startet eine *transition* durch den Aufruf von *selection.transition* und kann anschließend alle Anzeigeänderungen, die man an der *selection* vornehmen kann auch über die zurückgegebene *transition* erreichen. Dabei kann mit der Funktion *transition.duration* eine Dauer und mit *transition.delay* eine Verzögerung des Übergangs in Millisekunden angegeben werden. Darüber hinaus bietet *transition.ease* die Möglichkeit eine Übergangsfunktion zu bestimmen, die vorgibt wie schnell sich der jeweilige Wert zu gegebenen Zeitpunkten ändert. Zum Beispiel kann eine Bewegung schnell beginnen und zum Ende hin langsamer werden, wofür die Übergangsfunktion *easeExpOut* geeignet wäre, deren Funktionsgraph in Abbildung 2.8 (Bostock [5]) zu sehen ist. D3 bringt bereits eine Reihe solcher Funktionen mit, es können aber auch eigene definiert werden.

3

Contribution

3.1 Baumlayout

[todo: Kurzer einleitungssatz zwischen den beiden überschriften einfügen]

3.1.1 Linear vs. Radial

Aufgrund des begrenzten Platzes, der auf mobilen Geräten typischerweise zur Verfügung steht, ist zu entscheiden, wie Baumstrukturen möglichst platzsparend anzuordnen sind, ohne Übersichtlichkeit einzubüßen. Betrachtet werden dazu speziell die allgemein übliche Darstellung (linear) gegenüber einer kreisförmigen (radialen) Anordnung.
Die Abbildungen 3.1 und 3.2 zeigen einen Vergleich zwischen einem Baum in linearer (Abb. 3.1) und in radialer Anordnung (Abb. 3.2). In beiden Fällen ist derselbe Baum der Höhe 4

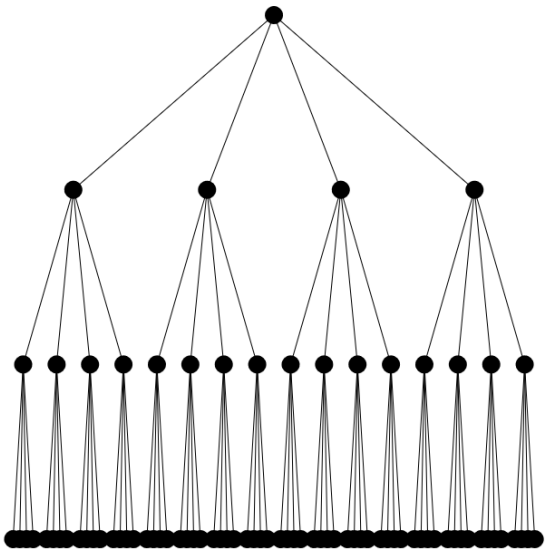

Abbildung 3.1: Lineare Baumdarstellung.

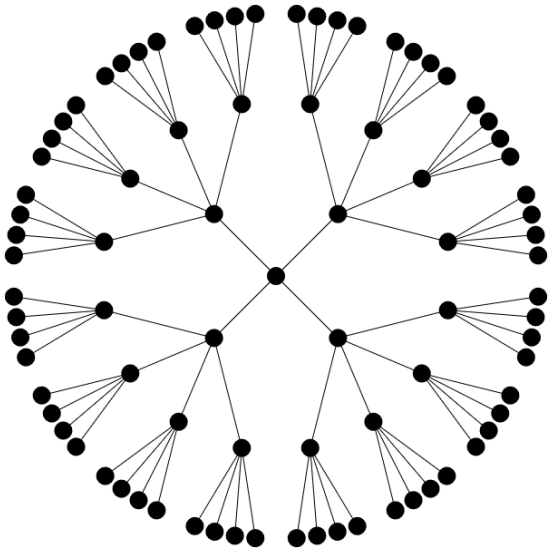

Abbildung 3.2: Radiale Baumdarstellung.

auf gleicher Fläche abgebildet. Er besteht aus 105 Knoten, wobei alle inneren Knoten jeweils vier Nachfolger haben. Der Wurzelknoten befindet sich beim linearen Layout am oberen Rand. Die übrigen Knoten sind in drei horizontalen Linien darunter angeordnet, wobei die Kinder der Wurzel auf der obersten Linie liegen, deren Kinder auf der mittleren und deren Kinder auf der untersten Linie. Beim radialen Layout ist die Wurzel in der Mitte abgebildet und alle anderen Knoten in drei konzentrischen Kreisen darum herum. Knoten der Tiefe 1 liegen auf dem innersten Kreis, der Tiefe 2 auf dem mittleren Kreis und die Blätter auf dem äußeren Kreis. Den Mittelpunkt der Kreise bildet die Wurzel. Man sieht deutlich, dass im Fall der vertikalen Anordnung (Abb. 3.1) nicht ausreichend Platz für alle Blätter zur Verfügung steht, wodurch es zu Überschneidungen kommt. In der radialen Anordnung (Abb. 3.2) können hingegen alle Knoten überschneidungsfrei dargestellt werden.

Um den Grund dafür zu veranschaulichen vergleicht man den Platz, der bei den beiden Herangehensweisen auf einer einzelnen Stufe des Baumes zur Verfügung steht. Unter der Annahme, dass für die gesamte Darstellung ein Quadrat mit Seitenlänge a zur Verfügung steht und alle Knoten einen Durchmesser von 1 haben ergibt sich für den Platz l_{linear} bzw. l_{radial}, der auf einer Stufe des Baumes verfügbar ist

$$\begin{aligned} l_{linear} &= a \\ l_{radial} &= 2 \cdot \pi \cdot r \qquad mit \quad 0 \leq r \leq \frac{a}{2}. \end{aligned}$$

Dabei bezeichnet r den Abstand einer Stufe zum Wurzelknoten. Setzt man l_{radial} und l_{linear} gleich und stellt nach r um, so ergibt sich

$$\begin{aligned} & l_{linear} = l_{radial} \\ \Rightarrow \quad & a = 2 \cdot \pi \cdot r \\ \Rightarrow \quad & \frac{a}{2 \cdot \pi} = r. \end{aligned}$$

Da l_{linear} konstant ist kann man daraus ableiten

$$l_{radial} > l_{linear} \qquad \Leftrightarrow \qquad r > \frac{a}{2 \cdot \pi}.$$

Das bedeutet, dass ab einem Abstand zur Wurzel von mehr als $\frac{a}{2 \cdot \pi}$ im radialen Layout mehr Platz für jede Stufe verfügbar ist. Außerdem gilt: Je größer der Abstand r, desto mehr Platzersparnis. Das ist vor allem deshalb von Interesse, weil Bäume in der Regel die Eigenschaft haben, dass mit wachsender Baumtiefe auch die Anzahl der Knoten in der jeweiligen Tiefe wächst und diese tiefer liegenden Knoten in der kreisförmigen Darstellung weiter von der Wurzel entfernt sind. Das radiale Layout bietet somit genau dann mehr Platz, wenn üblicherweise mehr Platz benötigt wird. Aufgrund dieser Erkenntnis und der Tatsache, dass beide Darstellungsformen gute Übersichtlichkeit aufweisen, fiel die Wahl letztendlich auf das radiale Layout.

3.1.2 Polarkoordinaten

D3 bietet zur Berechnung einer sinnvollen Knotenanordnung von Bäumen die Funktion *d3.tree* an, welche unter Verwendung des Reingold-Tilford Algorithmus (Reingold und Tilford [22]) allen Knoten einer *hierarchy* x- und y-Koordinaten jeweils im Bereich von 0 bis 1 zuordnet. Es liegt dann in der Hand des Programmierers, diese sinnvoll zu interpretieren. Um die Knoten,

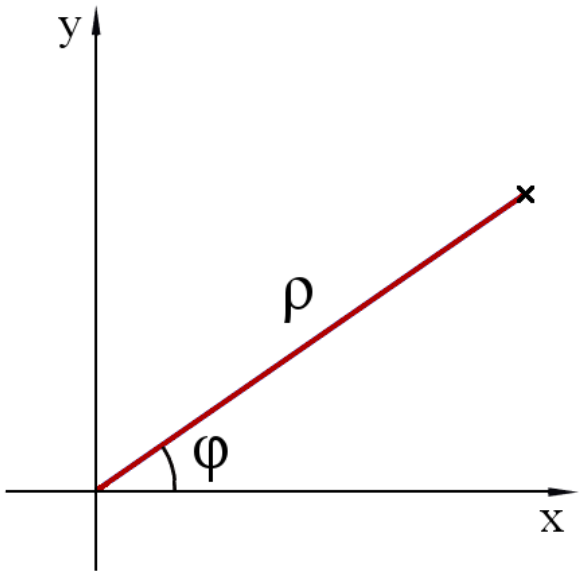

Abbildung 3.3: Veranschaulichung von Polarkoordinaten.

wie Abbildung 3.2 zeigt in konzentrischen Kreisen anzuordnen, eignen sich Polarkoordinaten ausgezeichnet. Im Koordinatensystem in Abbildung 3.3 ist ein Beispiel für die Angabe eines Punktes in Polarkoordinaten. Er wird eindeutig definiert durch den Steigungs- oder Polarwinkel φ und den Radius ρ, welcher die Entfernung vom Koordinatenursprung angibt. Ein Vorschlag aus der Dokumentation von D3 lautet „[...] for example, you can treat x as an angle and y as a radius to produce a radial layout.“ (Bostock [8]). Diesem folgend wird die y-Koordinate als Radius ρ, die x-Koordinate als Polarwinkel φ in Radiant interpretiert.

$$\rho = y$$
$$\varphi = 2 \cdot \pi \cdot x$$

Zur Darstellung auf dem Bildschirm müssen ρ und φ anschließend in kartesische Koordinaten x_{screen} und y_{screen} umgerechnet werden. Die allgemeinen Umrechnungsformeln ergeben sich als

$$x_{screen} = \rho \cdot \cos(\varphi)$$
$$y_{screen} = \rho \cdot \sin(\varphi).$$

Bei dieser Umrechnung wird allerdings noch nicht berücksichtigt, dass die gegebenen Polarkoordinaten auf generischen Koordinaten im Bereich $[0,1]$ basieren. Zur korrekten Positionierung müssen noch eine Skalierung auf die verfügbare Breite (width) w und Höhe (height) h, sowie eine Verschiebung in die Mitte der Anzeige vorgenommen werden. Es entstehen die endgültigen Formeln:

$$x_{screen} = \rho \cdot \cos(\varphi) \cdot w + \frac{w}{2} \tag{3.1}$$

$$y_{screen} = \rho \cdot \sin(\varphi) \cdot h + \frac{h}{2}. \tag{3.2}$$

3.2 Reduzieren der angezeigten Knoten

Nachdem mit dem radialen Layout (Kapitel 3.1.1) eine erste Maßnahme zum Einsparen von Platz ergriffen wurde, stellt sich als nächstes die Frage, wie die Übersichtlichkeit weiter verbessert werden kann. Da Knoten auf dem Bildschirm Informationen in Form von Text beinhalten sollen, ist es abzusehen, dass jeder einzelne Knoten mehr Platz einnehmen wird, als z.B. in Abbildung 3.2 gezeigt ist. Es bietet sich an, immer nur aktuell relevante Knoten ein- und irrelevante auszublenden. Das erfordert je nach Eingabe dynamische Änderungen an der Anzeige der Baumstruktur (Kapitel 3.5.1). Zunächst muss jedoch entschieden werden, welche Knoten aktuell relevant oder irrelevant sind.

Dazu betrachten wir noch einmal den simplen Entscheidungsbaum in Abbildung 2.3. Die beiden Stellen, an denen Entscheidungen getroffen werden müssen, sind dort mit *A* und *B* markiert. In Situation *A* gilt es zu entscheiden, ob es regnet oder nicht. Um diese Entscheidung treffen zu können, ist nicht relevant, ob es zu einem späteren Zeitpunkt regnen wird oder nicht und welche Ergebnisse sich daraus ableiten lassen. Bei *B* wiederum sind vorherige Entscheidungsmöglichkeiten nicht von Interesse, ebenso wie die Folgen davon, ob es regnen wird oder nicht. Man kann sagen, dass bei jeder Verzweigung nur die zur Verfügung stehenden Alternativen relevant sind und angezeigt werden müssen. Da es jedoch nicht nur darum geht, so viel Platz wie möglich zu sparen, sondern auch darum, eine übersichtliche und intuitiv verständliche Darstellung zu finden ist es hilfreich, außerdem noch den Knoten, von dem die Verzweigung ausgeht zu zeigen. Bei *A* also den Knoten „Regenschirm nötig“, bei *B* „Es regnet nicht“. Auf diese Art ist es leichter die Orientierung zu behalten, selbst wenn ein Großteil des Baumes nicht sichtbar ist.

Das Grundlegende Konzept, bestehend aus dem radialen Layout und dem Weglassen irrelevanter Knoten, kann nun unter Verwendung von D3 umgesetzt werden.

3.3 Baumdarstellung mit D3

Um mit D3 auf den Bildschirm zu zeichnen gibt es verschiedene Möglichkeiten. In der Regel werden die HTML-Elemente *canvas* oder *svg* als Zeichenfläche verwendet. Beide haben Vor- und Nachteile, die vor einer Entscheidung abzuwägen sind.

3.3.1 Vergleich zwischen Canvas und SVG

Der größte Unterschied zwischen *canvas* und *svg* Elementen besteht darin, dass ein *svg* jedes Objekt, dass es anzeigen soll einzeln als HTML-Element hinterlegt während ein *canvas* nur das insgesamt entstehende Bild speichert. Für den Browser bedeutet das, dass bei einem *canvas* nur die Darstellung eines einzelnen Bildes zu berechnen ist, während bei einem *svg* alle dargestellten Objekte getrennt behandelt werden. Das führt dazu, dass mit wachsender Zahl anzuzeigender Elemente die Berechnungsdauer des *svg* stärker ansteigt, als die des *canvas*. Das einzelne Speichern der Anzeigebausteine bringt aber auch Vorteile mit sich. Es ist einfach ein einzelnes Objekt auf dem Bildschirm zu verändern, indem man dessen *svg*-Attribute anpasst, weil der Browser sich dann selbst um die Aktualisierung der Anzeige kümmert. D3 bietet mit seinen *transitions* eine simple Möglichkeit beliebige Animationen auf einem *svg* Element durchzuführen, was auf einem *canvas* umständlicher mit D3 *interpolators* und einer

Funktion zu lösen ist, die für jedes Einzelbild der Animation das Bild neu zeichnet. Allgemein scheint die Nutzung eines *svg*-Elements zur Darstellung von Daten mit D3 die am häufigsten gewählte Herangehensweise, was zu einer großen Menge Beispiele führt, die zur Inspiration genutzt werden können. Auch in diesem Fall fällt die Wahl darauf D3 in Verbindung mit *svg* einzusetzen. Diese Entscheidung wird vor allem durch die in Kapitel 3.2 beschriebene Reduzierung der angezeigten Knoten untermauert, denn der große Vorteil schnellerer Berechnung auf *canvas*-Elementen fällt damit kaum noch ins Gewicht.

3.4 Struktur der verwendeten Daten

Die visualisierten Daten werden von einem Datenbankserver in Form einer JSON-Datei zur Verfügung gestellt. JSON ist ein Format, mit dem Javascript-Objekte in Textform gespeichert werden können. Die Daten enthalten ein graphenbasiertes Entscheidungsmodell in Form eines Baumes. Die komplette Datei repräsentiert die Wurzel des Baumes, welche verschiedene Attribute sowie alle ihre Kinder beinhaltet. Diese Kinderknoten enthalten wiederum ihre eigenen Nachfolger und Eigenschaften und so geht es weiter bis zu den Blättern des Baumes. Ein einzelner Knoten hat die Folgenden Attribute:

name Der Name des Knotens, welcher später für Beschriftungen verwendet wird.

parent Der Vorgängerknoten. Im Fall der Wurzel ist dieser Eintrag leer.

children Eine Liste der Nachfolger. Blätter haben eine leere Liste.

id Eindeutige Identifikationsnummer.

description Ausführliche Beschreibung der Bedeutung des Knotens.

type Der Knotentyp. Die vier möglichen Typen sind *symptom*, *diagnosis*, *examination* und *therapy*

Zur Speicherung der Baumdaten werden die Knoten in eine D3 *hierarchy* Umgewandelt. Es ist zu beachten, dass eine *hierarchy* nur durch ein einzelnes Objekt repräsentiert wird, welches die Wurzel (*root*) des Baumes darstellt. Um eine Liste aller Knoten zu erhalten kann *root.descendants* aufgerufen werden. Jeder einzelne Knoten ist dabei ein Javascript-Objekt, das die in Kapitel 3.4 beschriebenen Daten enthält. Wendet man die Funktion *d3.tree* (Kapitel 2.3.2) auf die *hierarchy* an, so kommen zusätzlich noch x- und y-Koordinaten dazu. Unter Einsatz dieser Daten kann der Baum in einem *svg*-Element dargestellt werden.

3.4.1 Erzeugung eines SVG

Zu Beginn wird dem HTML-Dokument mit *d3.select* und *selection.append* ein *svg*-Element angehängt. Die Rückgabe von *append* – eine *selection*, die nur das *svg* enthält – wird in einer Variable zwischengespeichert.

Abbildung 3.4 zeigt das gewünschte Aussehen für Knoten und Kanten. Zu sehen sind zwei Knoten als Kreise mit Beschriftung A und B. Die Kante zwischen den Beiden Knoten ist dargestellt als Linie, die beide Kreise miteinander verbindet. Die unterschiedlichen Farben der Knoten zeigen an, dass sie verschiedene Knotentypen haben. Im medizinischen Kontext könnte

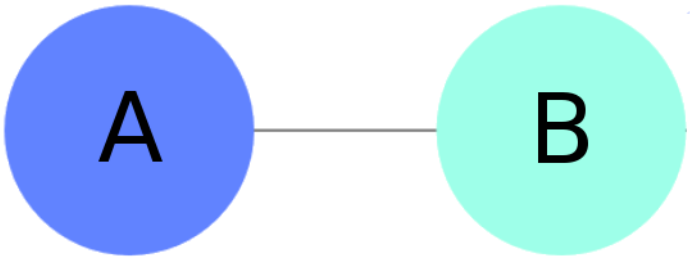

Abbildung 3.4: Konzept für Knoten- und Kantendarstellung.

es sich zum Beispiel um ein Symptom A und eine Diagnose B handeln. Die Farben sind in Blau- und Lilatönen gehalten. Es ergibt Sinn, für Knoten und Kanten getrennte Gruppen zu erstellen. Dazu werden unter der Verwendung der zuvor gespeicherten *selection* zwei *g*-Elemente hinzugefügt, die im Folgenden als Knoten- und Kantengruppe bezeichnet werden.

3.4.2 Präparieren der Knotendaten für die Darstellung

Bevor die Knoten auf den Bildschirm gebracht werden können, müssen zunächst deren Koordinaten bestimmt werden. Außerdem soll darauf geachtet werden, dass wie in Kapitel 3.2 erörtert nur der aktuelle Verzweigungsknoten und dessen Kinder angezeigt werden. Die zur Koordinatenberechnung verwendete Funktion *d3.tree* erhält als Übergabe einen Startknoten und läuft bei der Berechnung von dort den gesamten nachfolgenden Baum ab, wobei jedem Knoten x- und y-Koordinaten gegeben werden. Dazu verwendet sie immer die Liste *children*, die in jedem Knoten enthalten ist und dessen Kinder aufzählt. Erreicht die Funktion einen Knoten, der kein *children*-Objekt enthält, dann wird dieser als Blatt betrachtet. Da nur ein Knoten und dessen Nachfolger in die Berechnung einbezogen werden sollen, muss *d3.tree* die Nachfolger dieses Knotens als Blätter betrachten. Das wird erreicht, indem die *children*-Objekte der Kinder entfernt und in einem *childrenBackup* benannten Objekt zwischengespeichert werden. Hat *d3.tree* anschließend allen anzuzeigenden Knoten ihre Koordinaten zugewiesen, werden diese noch mit dem in Kapitel 3.1.2 beschriebenen Verfahren umgerechnet, um eine kreisförmige Anordnung zu erzeugen.

Ein weiterer Voreilt dieses Verfahrens liegt darin, dass zum Erhalten einer Liste der anzuzeigenden Knoten *currentRoot.descendants* aufgerufen werden kann, weil auch diese Funktion die *children* Objekte zum ablaufen des Baumes verwendet. Das Objekt *currentRoot* bezeichnet dabei den aktuellen Verzweigungsknoten.

3.4.3 Erzeugen der Knotendarstellung

Wie auf Abbildung 3.4 zu sehen ist, soll die Knotenvisualisierung aus einem Kreis mit einer Aufschrift bestehen. Für jeden Knoten muss das *svg* also ein *circle* und ein *text*-Element enthalten. Um leichter beide zusammen ein- und ausblenden sowie verschieben zu können, sollen diese jeweils Gruppiert werden. Mit *selectAll* wählt man zuerst alle *g*-Elemente in der Knotengruppe aus und Verknüpft diese unter Einsatz von *selection.data* mit den gewünschten Knotendaten, welche nach dem Präparieren der Daten mit *currentRoot.descendants* zu erhalten sind.

Abbildung 3.5: Knoten mit verschiedenen Textlängen.

Die daraus entstehende *enter selection* verwendet man dann um für alle neu hinzugekommenen Knoten *g*-Elemente einzufügen, die einen *circle* mit zentriertem *text* enthalten. Der Radius des Kreises ist abhängig von der Bildschirmgröße und der Text ist der Name des Knotens, welcher den Knotendaten entnommen wird. Mit dem *transform*-Attribut der *g*-Elemente werden die Gruppen an die zuvor berechneten Koordinaten geschoben.

Wegen der begrenzten Bildschirmgröße von mobilen Geräten ist es ein Problem, die Knotenaufschrift vollständig und gleichzeitig gut lesbar anzuzeigen.

3.4.4 Berechnen der Schriftgröße

Die Schriftgröße soll nach Möglichkeit so gewählt werden, dass der Text auf eine Zeile passt. Wird die Schrift dabei zu klein, so wird sie stattdessen auf zwei Zeilen aufgeteilt.

Abbildung 3.5 zeigt die drei möglichen Szenarien, die auftreten können. Die Beschriftung „Es wird nicht regnen“ des linken Knotens kann in einer Zeile dargestellt werden und nimmt dabei möglichst viel Platz ein, während die Aufschrift der anderen beiden Knoten aufgeteilt werden muss. Der Text des rechten Knotens ist so lang, dass er auch auf zwei Zeilen nicht passt und daher abgeschnitten wird, was die drei Punkte verdeutlichen.

Bei der Berechnung werden Schriftgrößen in *em* angegeben, was eine Einheit ist, die auf verschiedenen Geräten unterschiedlich interpretiert wird. $1em$ ist dabei die auf dem aktuellen Gerät übliche Größe für lesbare Schrift, $2em$ ist doppelt so groß etc. Generell gilt, dass Höhe und Breite eines Textes proportional zu dessen Schriftgröße sind.

Zunächst wird mit der Funktion *getComputedTextLength* die Breite b des Textes bei Schriftgröße $1em$ berechnet. Wegen der zuvor genannten Proportionalität ergibt sich daraus die Formel für die maximale Schriftgröße sg unter Verwendung des Knotenradius r als

$$sg = \frac{2 \cdot r}{b}. \tag{3.3}$$

1.Fall $sg \geq 1$ Der Text wird in eine Zeile geschrieben, wenn sg mindestens 1 ist. Die Schriftgröße ist dabei sg, wenn $sg \leq 1,3$ und ansonsten $1{,}3em$. Mit der Einhaltung dieses Maximalwerts soll verhindert werden, dass die Schrift riesig erscheint.

2.Fall $sg < 1$ Ist die maximale Schriftgröße kleiner als $1em$ dann wird der Text in zwei Zeilen aufgeteilt. Das verwendete Verfahren zum Platzieren des Zeilenumbruchs wird später in Kapitel 3.9.1 näher beschrieben. Die zweite Zeile wird dabei mit „...“ abgekürzt, wenn sie die Maximalbreite b überschreitet.

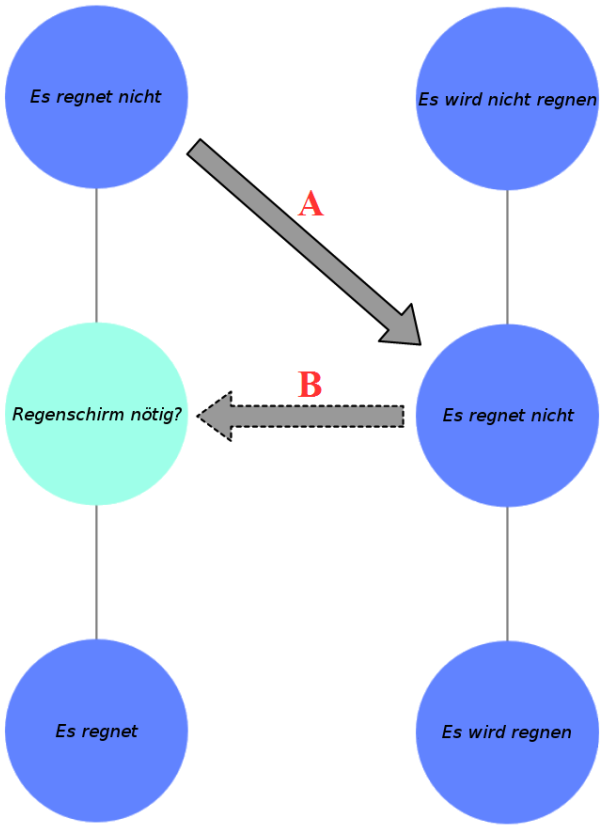

Abbildung 3.6: Beispiel zur Baumnavigation.

3.4.5 Erzeugen der Kantendarstellung

Die Darstellung der Kanten verläuft analog zur Knotendarstellung in Kapitel 3.4.3, allerdings werden anstelle von *g*-Elementen mit *circle* und *text* jetzt *path*-Elemente verwendet. Zur Erzeugung der *enter selection* wird als Daten die Rückgabe der *hierarchy*-Funktion *node.links* verwendet, welche eine Liste aller Kanten, die von einem Knoten ausgehen, zurückgibt. Aus den Einträgen dieser Liste lassen sich mit *d*3.*line* die Zeichenanweisungen für das *d*-Attribut der *path*-Elemente berechnen.

3.5 Navigation im Baum

Durch das Ausblenden vieler der Knoten muss nun eine Möglichkeit geschaffen werden, von einer Entscheidung zur nächsten zu navigieren, wobei immer wieder irrelevante Knoten aus- und relevant gewordene eingeblendet werden. Das wird möglich gemacht, indem die Baumanzeige auf Berührung hin dynamisch aktualisiert wird. Es gilt dabei, dass der Knoten, von dem die aktuelle Verzweigung ausgeht, in der Mitte des Bildschirms zu sehen ist und die nachfolgenden Entscheidungsmöglichkeiten im Kreis darum herum angeordnet werden. Abbildung 3.6 zeigt die Reaktion des Baumes auf verschiedene Nutzereingaben. Auf der linken Seite ist der Ausgangszustand zu sehen und rechts der Nachfolgezustand. Links ist der mittlere Knoten mit „Regenschirm nötig?“ beschriftet und dessen Nachfolger mit „Es regnet nicht“ sowie „es regnet“. Rechts sieht man „Es regnet nicht“ in der Mitte, „Es wird nicht regnen“ und „Es wird

regnen“ als Folgeknoten. Die mit *A* und *B* beschrifteten Pfeile symbolisieren Reaktionen auf Eingaben, die im Folgenden genauer beschrieben werden.

Durch Antippen eines der äußeren Knoten wird dieser in die Mitte verschoben und es zeigen sich dessen Nachfolger. Tippt man zum Beispiel in Abbildung 3.6 links „Es regnet nicht“ an, dann wird dieser, wie Pfeil *A* zeigt, zum mittleren Knoten, die anderen verschwinden und es erscheinen die neuen Nachfolgeknoten. Möchte man einen Schritt zurück machen, kann der Mittelknoten berührt werden und man wird wieder zur links gezeigten Situation geleitet (illustriert durch Pfeil *B*). Es soll dadurch das Gefühl entstehen, dass man sich durch die Baumstruktur bewegt, wobei besonders die gewählten Animationen (Kapitel 3.7) von großer Wichtigkeit sind. Zuerst ist aber zu klären, wie die interne Datenstruktur verwendet werden kann, um dynamische Aktualisierungen zu ermöglichen.

3.5.1 Dynamische Aktualisierung

Mit dem Präparieren der Daten in Kapitel 3.4.2 ist bereits der Grundstein für das Aktualisieren der Anzeige auf Nutzereingaben hin gelegt worden. Wird einer der Entscheidungsknoten berührt, so soll dieser zum neuen mittleren Knoten werden und dessen Nachfolger darum herum erscheinen (Abbildung 3.6). Dazu wird als erstes das *children*-Objekt des ausgewählten Knotens, welches beim Präparieren der vorherigen Daten in das Objekt *childrenBackup* verschoben wurde, wiederhergestellt. Anschließend werden mit dem angetippten Knoten als Verzweigungsknoten die Schritte vom Präparieren der Daten bis zur Anzeige erneut durchgeführt (siehe Kapitel 3.4.2 bis 3.4.5). Der Unterschied besteht diesmal aber darin, dass nicht nur die *enter selections* beim Verknüpfen der Daten betrachtet werden, sondern auch die *update*− und *exit selections*. Die *update selection* enthält dann den neuen Mittelknoten, weil er bereits vorher auf dem Bildschirm war. Dieser muss nur an seine neue Position in der Mitte der Anzeige verschoben werden, indem das *transform*-Attribut seines *g*-Elements angepasst wird. In der *exit selection* befinden sich der vorherige Verzweigungsknoten und dessen Kinder, die nicht ausgewählt wurden. Sie werden mit der Funktion *selection.remove* vom Bildschirm entfernt. [todo: verweise auf abbildung zur veranschaulichung]

3.6 Eingabemenü

Für die medizinische Nutzung der Visualisierung soll die Anwendung mit einem Server kommunizieren, der Ergebnisse von Untersuchungen und getroffene Behandlungsentscheidungen speichert. Es soll aber nicht immer, wenn ein Knoten berührt wird, sofort eine Änderung in der Datenbank des Servers vorgenommen werden, damit die Möglichkeit erhalten bleibt, ohne Konsequenzen den Baum zu erkunden. Das ist zum Beispiel bei der Einarbeitung neuer Mitarbeiter nützlich oder wenn anhand der Visualisierung einem Patienten ein Behandlungsablauf erklärt werden soll. Deshalb wird das in Abbildung 3.7 zu sehende Eingabemenü eingeführt. Zu sehen ist ein Knoten mit der Aufschrift „Has calprotectin in stool“ und dessen Nachfolger „Inflammatory diarrhea“, „negative stool bacteriology“ und „positive stool bacteriology“, um welchen das Eingabemenü angezeigt wird. Es besteht aus drei Ringsegmenten in den Farben Grün, Rot und Gelb. Das grüne enthält ein Häkchen, das rote ein Kreuz und das gelbe ein Fragezeichen. Sie dienen dazu, Knoten als positiv, negativ oder neutral zu markieren, was im Hintergrund an den Datenbankserver weitergeleitet wird.[todo: Könntest du hier ein Sequenz-

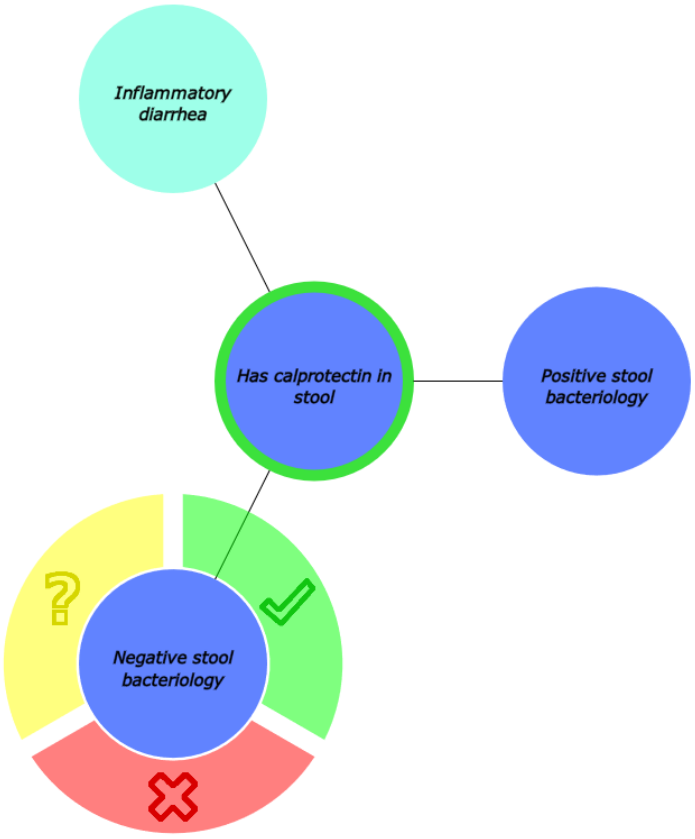

Abbildung 3.7: Eingabemenü zum Eintragen von Behandlungsergebnissen.

diagramm hinzufügen zur veranschaulichung der kommunikationsschritte mit dem backend bei useraktionen?] Der Knoten, um den auf der Abbildung das Eingabemenü geöffnet wurde, betrifft eine bakteriologische Stuhlprobenuntersuchung. Nach Erhalt des Ergebnisses, kann der Knoten durch Antippen des grünen oder roten Segments als positiv beziehungsweise negativ markiert werden. Die Markierung wird anschließend durch eine farbige Umrandung deutlich gemacht, welche die gleiche Farbe hat wie die berührte Schaltfläche. In Abbildung 3.7 zeigt die gründe Umrandung des mittleren Knotens beispielsweise an, dass er als positiv markiert wurde. Die gelbe Schaltfläche mit dem Fragezeichensymbol kann im Falle einer fehlerhaften Eingabe genutzt werden, um die Markierung eines Knotens zusammen mit dem Eintrag in der Datenbank wieder rückgängig zu machen.

Das Eingabemenü wird geöffnet, indem man einen der Knoten mit dem Finger gedrückt hält.

3.6.1 Erzeugung des Eingabemenüs

Das Eingabemenü gleicht in seiner Form einem Ringdiagramm mit drei gleichgroßen Segmenten. Die in Kapitel 2.3.2 beschriebenen Funktionen *d3.arc* und *d3.pie* dienen zur Erzeugung eines solchen Diagramms. Als Daten erhält *d3.pie* eine Liste mit drei Objekten, die jeweils das Attribut *value* mit Wert 1 und *cssClass* mit einem der Werte „positive“, „negative“ und „unknown“ enthalten. *D3.pie* verwendet dabei *value* als Datenwert, welcher für alle drei Objekte

gleich ist, wodurch drei gleichgroße Ringsegmente entstehen. Mit der Funktion *pie.padAngle* wird noch angegeben, wie viel Abstand diese voneinander haben sollen. Die von *d3.pie* generierte Rückgabe kann anschließend an *d3.arc* übergeben werden, welches daraus Zeichenangaben für *path*-Elemente macht.

Zuerst wird dem *svg* eine neue Gruppe in Form eines *g*-Elements hinzugefügt. Dann wird ähnlich wie beim erzeugen der Knoten- und Kantendarstellung (Kapitel 3.4.3 und 3.4.5) vorgegangen. Mit *selectAll* wird eine *selection* aller *path*-Elemente in der neu erzeugten Gruppe erstellt und diese dann mit den von *d3.pie* generierten Daten verknüpft. Über die *enter selection* fügt man dann neue Pfade ein und gibt diesen als *d*-Attribut die mit *d3.arc* aus den Daten entstehenden Zeichenbefehle. Zusätzlich erhält jedes der drei *path*-Elemente als CSS-Klasse den Wert der Variable *cssClass* , welche im verknüpften Datensatz hinterlegt ist. Über diese wird die Füllfarbe des jeweiligen Pfades festgelegt. Des Weiteren fügt man der Gruppe die in Abbildung 3.7 zu sehenden Icons als *image*-Elemente ein, deren Positionierung mit *arc.centroid* berechnet werden.

3.7 Einsatz von Animationen

Normalerweise wird zum Verändern der Anzeige eine *selection* mit allen zu verändernden Objekten erstellt und diese dann mit der Funktion *selection.attr* angepasst. Dabei verändern sich die Ausgewählten Elemente sofort. Fließende Übergänge werden unter Einsatz von *transitions* (Kapitel 2.3.2) realisiert. Anstatt ein Objekt per *selection.attr* zu bearbeiten erzeugt man mit *selection.transition* eine Animation und wendet *transition.attr* an, wobei verschiedene Parameter, wie Dauer und Übergangsfunktion zur Konfigurierung verfügbar sind.

In Googles „Material design guidelines“ heißt es „Motion provides meaning“ (Google [17]) – zu deutsch „Bewegung verleiht Bedeutung“. Diesem Prinzip folgend kommen bei der hier beschriebenen interaktiven Visualisierung Animationen zum Einsatz und spielen dabei für Interaktivität und verbesserte Orientierung eine zentrale Rolle.

3.7.1 Interaktivität

Für eine interaktive Anwendung ist es wichtig, dass Nutzer immer das Gefühl haben, dass sie das Geschehen auf dem Bildschirm kontrollieren. Wenn sich Elemente auf Eingaben hin unvorhersehbar oder nicht nachvollziehbar verhalten, ist das nicht gewährleistet. Durch kontinuierliche Bewegungen, wie in Abbildung 3.8, anstelle von plötzlichen Sprüngen kann immer nachvollzogen werden, wie die Visualisierung sich verändert, womit eine interaktivere Umgebung geschaffen wird.

3.7.2 Verbesserung von Orientierung und Übersichtlichkeit

Kapitel 3.2 beschreibt, wie zum Einsparen von Platz Knoten weggelassen werden können. Ein daraus folgender Nachteil besteht im Orientierungsverlust. Obwohl die Menge an Informationen auf dem Bildschirm sinkt, was die Übersichtlichkeit verbessert, wird es schwerer zu wissen, wo man sich im Baum gerade befindet. Durch passende Animationen beim in Kapitel 3.5.1 beschriebenen dynamischen Aktualisieren des Baumes kann dem Abhilfe geschaffen werden. Es

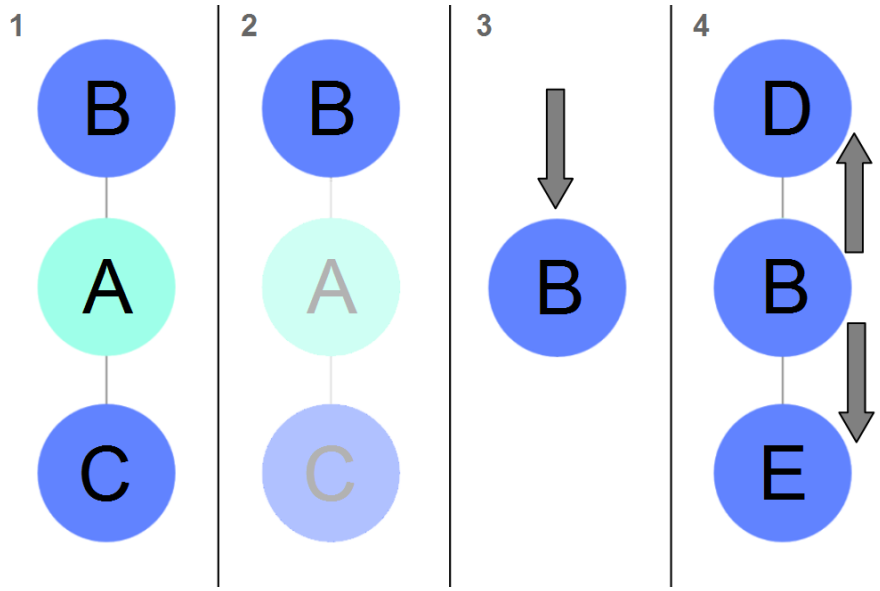

Abbildung 3.8: Animation beim Berühren eines Knotens.

soll dabei durch die Bewegungen auf dem Bildschirm deutlich werden, welche Veränderung eine Eingabe bewirkt. Die gewählte Animation beim Navigieren von einem Knoten zum nächsten ist in Abbildung 3.8 zu sehen. Von links nach rechts mit 1 bis 4 nummeriert werden die vier wichtigen Zeitpunkte im Bewegungsablauf gezeigt. Situation 1 zeigt den Ausgangszustand: Ein Knoten *A* mit dessen Nachfolgerknoten *B* und *C*. Die anderen drei Stadien zeigen was geschieht, wenn man *B* antippt. Zunächst werden bei 2 die Knoten *A* und *C* durchsichtiger, bis sie nicht mehr zu sehen sind. Anschließend bewegt sich *B* in die Mitte der Anzeige (zu sehen bei 3) und zuletzt bewegen sich zu Zeitpunkt 4 von *B* aus die Knoten *D* und *E* nach außen.

Das Ausblenden von *A* und *C* verdeutlicht, dass beide nicht mehr relevant sind, geschieht aber zur Vermeidung von Verwirrung nicht von einem Moment auf den anderen. Die Bewegung von *B* in die Mitte lässt leicht nachverfolgen, dass *B* zum neuen Verzweigungspunkt wird. Wichtig ist dabei, dass *B* zu keinem Zeitpunkt vom Bildschirm verschwindet oder umherspringt, wodurch klar ist, dass es sich noch immer um denselben Knoten handelt. Zuletzt verdeutlicht die von *B* ausgehende Bewegung von *D* und *E* nach außen, dass diese die Nachfolger von *B* sind. Der allgemeine Schwerpunkt liegt darauf, es durch kontinuierliche Bewegungen leichter zu machen, dem Geschehen zu folgen, was zu einem intuitiven Verständnis der Vorgänge auf dem Bildschirm führt.

3.8 Detailansicht

In der bisherigen Darstellung, welche im weiteren Verlauf als Navigation oder Navigationsansicht bezeichnet wird, werden Knoten nur mit ihrem Namen beschriftet. Dabei wird vernachlässigt, dass die Übergebenen Daten, welche in Kapitel 3.4 beschrieben werden, auch eine Ausführliche Beschreibung des Knotens beinhalten. Um diese anzuzeigen wird dem Dokument eine Detailansicht hinzugefügt. Abbildung 3.9 zeigt das Konzept dazu. Es handelt sich um Knoten aus einem Entscheidungsbaum zur Behandlung von Verdauungsstörungen – im Englischen „dyspepsia“. Die Knoten werden als Rechtecke dargestellt, die als Überschrift den Namen und

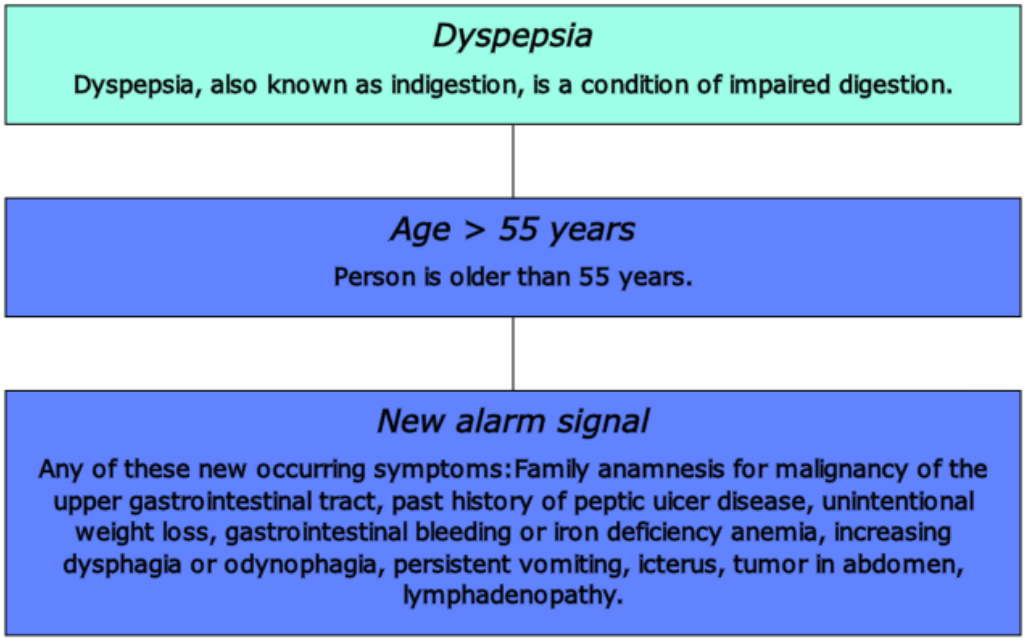

Abbildung 3.9: Detailansicht dreier Knoten.

darunter den ausführlichen Beschreibungstext des Knotens enthalten. Im diesem Beispiel wurde von der Wurzel „Dyspepsia“ aus der Knoten „Age > 55 years“ und von dort „New alarm signal“ ausgewählt. Die Hintergrundfarben der Rechtecke gleichen den Farben der korrespondierenden Kreise in der Navigationsansicht. Wie hier zu sehen ist bietet die Detailansicht neben der Knotenbeschreibung auch eine Anzeige des zurückgelegten Pfades durch den Entscheidungsbaum. Beim Einsatz im medizinischen Umfeld kann sie als eine Übersicht über die Patientengeschichte gesehen werden, die die wichtigsten Schritte auf einen Blick zusammenfasst.

3.8.1 Responsive Design

Das HTML-Dokument ist so aufgebaut, dass es vertikal in zwei Hälften aufgeteilt ist. Auf der linken Seite soll die Detailansicht und auf der rechten die Navigationsansicht zu sehen sein. Reicht die Bildschirmbreite nicht aus, um beide Ansichten nebeneinander zu positionieren, wird die Detailansicht ausgeblendet und die Navigation nimmt den gesamten Bildschirm ein. Es kann dann per Knopfdruck zwischen beiden Ansichten gewechselt werden. Die Abbildungen 3.10 und 3.11 zeigen das Aussehen der Anwendung im Portrait- und Landschaftsmodus. Im letzteren sind beide Ansichten zu sehen und werden durch eine vertikale, blaue Linie voneinander getrennt. Bei geringerer Breite des Bildschirms, wie im Portraitmodus, ist nur die Navigationsansicht zu sehen. Der blaue Kreis in der oberen, linken Ecke dient dann zum Aufrufen und Schließen der Detailansicht. Zur Erstellung dieses Aufbaus wird das CSS-Framework W3.CSS eingesetzt, welches für Responsive Webdesign entwickelt wurde [14].

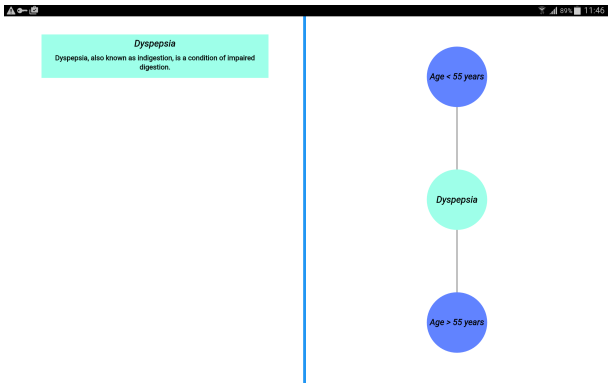

Abbildung 3.10: Landschafts-Ansicht auf einem Tablet

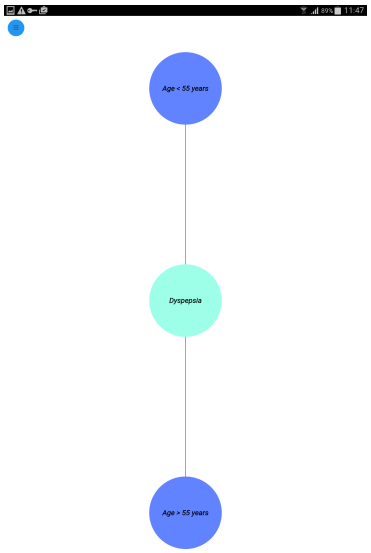

Abbildung 3.11: Portrait-Ansicht auf einem Tablet

3.9 Erzeugen der Detailansicht

Für die Detailansicht wird auf der linken Seite ein eigenes *svg*-Element hinzugefügt. Das verläuft analog zur in Kapitel 3.4.1 beschriebenen Erzeugung des *svg* für die Navigation. Auch hier werden für Knoten und Kanten eigene Gruppen erstellt.

Für die Knotendarstellung wird die Funktion *hierarchy.path* benutzt, die in Form einer Liste aller auf dem Weg passierten Knoten den Pfad von einem zum anderen zurückgibt. Da es in einem Baum nur genau einen möglichen Pfad zwischen zwei Knoten gibt, ist ein so berechneter Pfad von der Wurzel zum aktuellen Verzweigungsknoten immer auch der Pfad, welcher in der Navigationsansicht bisher gewählt wurde. Eine *selection* aller *g*-Elemente in der Knotengruppe wird per *selection.data* mit der von *hierarchy.path* erzeugten Liste verknüpft. Anhand der entstehenden *selection* werden nach dem in D3 üblichen Verfahren bereits existierende Elemente mit der *update selection* aktualisiert, neue mit der *enter selection* erzeugt und überflüssig gewordene Elemente unter Einsatz der *exit selection* entfernt.

Die Rechtecke, von denen die Knoten der Detailansicht umrahmt werden, sind *rect*-Elemente, deren Größe nach der im Folgenden beschriebenen Berechnung der Zeilenumbrüche festgelegt wird. Die Breite basiert auf der des Bildschirms, während die Höhe sich an das *text*-Element anpasst.

Die Erzeugung neuer Knoten beginnt mit dem Hinzufügen eines *text*-Elements, in welches sowohl die Überschrift als auch der Beschreibungstext eingefügt werden. Dabei wird mit dem gleichen Vorgehen, wie in Kapitel 3.4.4 die Schriftgröße der Überschrift so berechnet, dass sie den ihr zur Verfügung stehenden Platz nicht überschreitet. Da der Beschreibungstext eine beliebige Länge haben kann, macht es keinen Sinn, ihn so weit zu verkleinern, dass er auf eine Zeile passt. Stattdessen bekommt er eine feste Schriftgröße von $1em$ und es werden Zeilenumbrüche eingefügt.

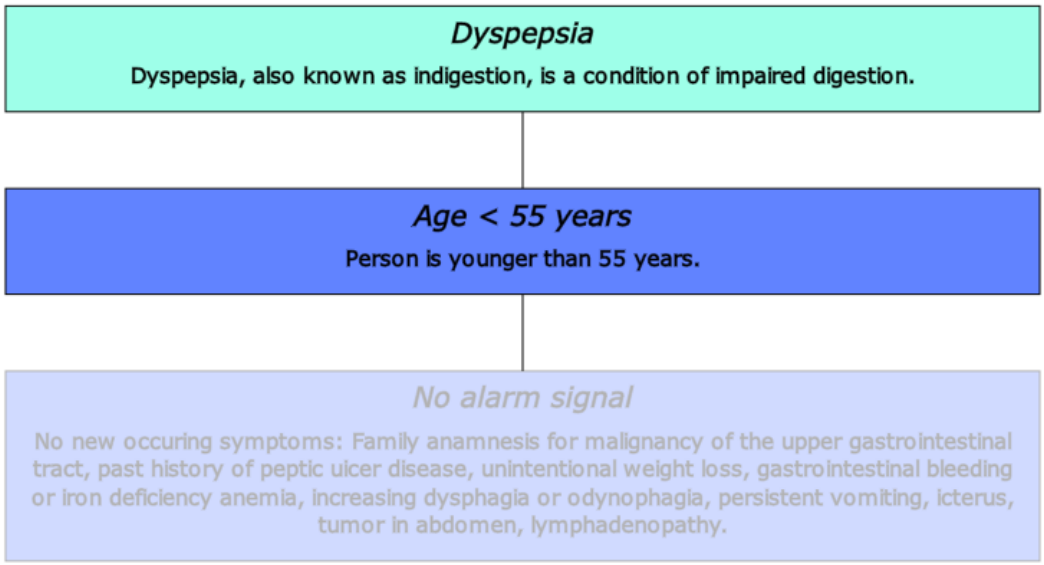

Abbildung 3.12: Detailansicht mit ausgegrautem Knoten.

3.9.1 Einfügen von Zeilenumbrüchen

Einem Beispiel von Mike Bostock [10] folgend werden Zeilenumbrüche mit Hilfe von *tspan*-Elementen innerhalb des *text*-Elements eingefügt. Dabei wird zunächst ein erstes *tspan* unterhalb der Überschrift platziert. Der Beschreibungstext wird umgewandelt in eine Liste aller einzelnen Wörter, und ein Wort nach dem anderen wird dem *tspan* hinzugefügt. Nach jedem Wort wird überprüft, ob das Element die erlaubte Breite überschreitet. Die Höhe und Breite eines Elements kann mit der Funktion *getBBox* ermittelt werden. Ist der Text zu lang, so wird das zuletzt zugefügte Wort wieder entfernt und in ein neues *tspan* eingefügt, welches unterhalb des vorherigen platziert wird. Dieser Vorgang wird so lange wiederholt, bis der vollständige Text untergebracht ist.

3.10 Dynamisches Aktualisieren der Detailansicht

Jedes mal, wenn eine Aktualisierung in der Navigation vorgenommen wird, aktualisiert sich auch die Detailansicht. [todo: Sequenzdiagramm wäre hier hilfreich] Wird eine neue Entscheidung ausgewählt und damit ein Schritt vorwärts im Baum gemacht, erscheint dieser Entscheidungsknoten mit seiner Beschreibung. Nach antippen des aktuellen Mittelknotens in der Navigationsansicht, was einen Schritt rückwärts im Baum verursacht, wird dieser Knoten in der Detailansicht ausgegraut. In Abbildung 3.12 wurde beispielsweise zuerst der Knoten „No alarm signal“ ausgewählt und anschließend wieder ein Schritt zurück gemacht. Der Knoten ist deshalb nur schwach zu sehen, während die oberen beiden Knoten deutlich zu erkennen sind. Auf diese Weise ist weiterhin sichtbar, auf welchem Pfad man sich zuvor bewegt hat. Ausgegraute Knoten werden entfernt, sobald ein anderer Pfad eingeschlagen wird.

Die Detailansicht kann außerdem genutzt werden, um mit einer einzelnen Eingabe mehrere Schritte rückwärts oder vorwärts zu machen. Alle Knoten in der Detailansicht können angetippt werden, um sofort in der Navigationsansicht zum jeweiligen Punkt zu springen. Auch hier

werden bei Rückschritten Knoten ausgegraut, die zum zuvor eingeschlagenen Pfad gehörten, wodurch zum Beispiel bei einer versehentlichen Fehleingabe sofort wieder dorthin vorgesprungen werden kann. [todo: abschließender satz]

4

Ergebnisse

Da keine etablierten Arbeiten zur Visualisierung von medizinischen Entscheidungsmodellen auf mobilen Geräten gefunden wurden, ist es schwer, sich Stärken und Schwächen anderer Lösungen zum Vorbild zu nehmen. Neue Ansätze, wie der in dieser Arbeit beschriebene, müssen deshalb von Grund auf Evaluiert werden, um so zu ermitteln, welche Aspekte von zukünftigen Herangehensweisen übernommen oder verbessert werden können.

4.1 Auswertung der Ergebnisse

Zur Evaluierung der im Zuge dieser Arbeit entstandenen Anwendung wurde eine Gruppe von 22 Personen befragt, sowohl aus dem Bereich der Medizin als auch aus anderen Berufsgruppen. Die genaue Verteilung ist in Tabelle 4.1 zu sehen. Alle Testpersonen erhielten eine kurze Erklärung[todo: Du kannst hierfür am besten noch screenshots hinzufügen hier oder in den anhang] der Anwendung und ihrer beabsichtigten Nutzung sowie Zugang zu einer Testversion mit verschiedenen Entscheidungsbäumen, die sie selbstständig ausprobieren durften. Es bestand die Auswahl zwischen zwei medizinischen und drei nicht-medizinischen Bäumen. Im Anschluss daran wurde ihnen eine Liste von Aussagen in zufälliger Reihenfolge gegeben, zu denen sie eine der vier Antwortmöglichkeiten „stimme nicht zu“, „stimme eher nicht zu“, „stimme eher zu“ und „stimme zu“ankreuzen konnten. Neben Meinungen zu visuellen Aspekten der Anwendung wurden auch Bereiche wie Verständlichkeit und intuitive Benutzbarkeit abgefragt. Zum Schluss war es den Befragten noch möglich per Freitexteingabe Anregungen zu geben, wo sie weitere Anwendungsmöglichkeiten für die Software sehen. Alle angegebenen Anteile wurden auf ganze Prozent, also zwei Stellen nach dem Komma gerundet und gegebenenfalls stehen die absoluten Stimmzahlen dahinter in Klammern. Auf den Abgebildeten Balkendiagrammen sind die Gesamtzahlen n der Antworten zur jeweiligen Aussage zu sehen. Im Anhang sind noch einmal sämtliche Ergebnisse in Tabellenform aufgeführt. Nachfolgend werden alle in der Umfrage enthaltenen Aussagen besprochen:

- ◇ Das Layout ist übersichtlich.
- ◇ Das Layout ist originell.

- ◇ Das Layout wirkt professionell.
- ◇ Das Layout wirkt zu überladen.
- ◇ Die Farben passen nicht zueinander.
- ◇ Die Farben wirken angenehm.
- ◇ Die Schrift ist gut lesbar.
- ◇ Ich empfinde das System als einfach zu nutzen.
- ◇ Es ist einfach, gesuchte Informationen zu finden
- ◇ Ich empfinde die Bedienung als intuitiv.
- ◇ Ich kann mir vorstellen, dass das System die Arbeit im genannten Kontext erleichtert.
- ◇ Die verwendeten Symbole sind verständlich.
- ◇ Die Software liefert ausreichend Informationen darüber, welche Eingaben möglich sind.

Tabelle 4.1: Angegebene Berufsgruppen

Medizin	Ingenieurwissenschaften	Informatik/IT	Geisteswissenschaften	Sonstige	Keine Angabe
3	2	12	1	2	2

4.1.1 Visuelle Aspekte

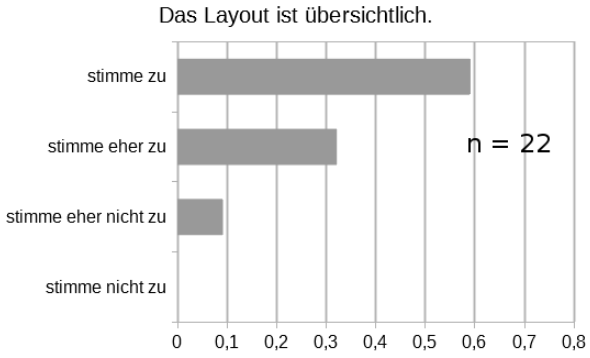

Abbildung 4.1: Auswertung der Übersichtlichkeit.

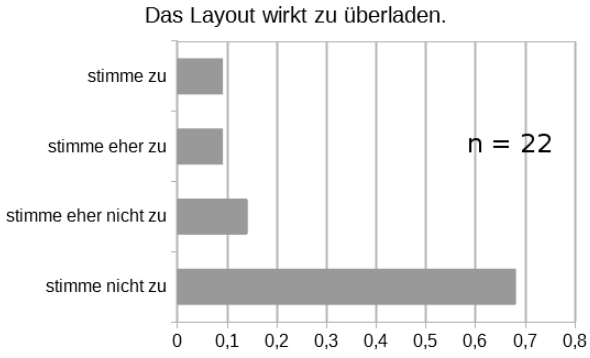

Abbildung 4.2: Auswertung der Überladenheit.

Von insgesamt vier Aussagen bezüglich des Layouts fielen die auf den Abbildungen 4.1 und 4.2 zu sehenden besonders deutlich aus. Sie zeigen die Ergebnisse zu den ähnlichen Punkten „Das Layout ist übersichtlich.“ und „Das Layout wirkt zu überladen.“ Die Kandidaten waren offenbar überwiegend der Meinung, dass die Anwendung eine gute Übersichtlichkeit aufweist und Nutzer nicht mit zu vielen Informationen auf einmal konfrontiert. So stimmten 59% (13) zu und 32% (7) eher zu, dass das Layout übersichtlich sei, während 68% (15) nicht und 14% (3)

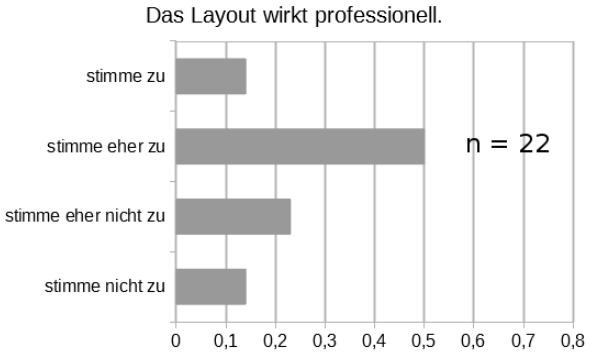

Abbildung 4.3: Auswertung der Professionalität.

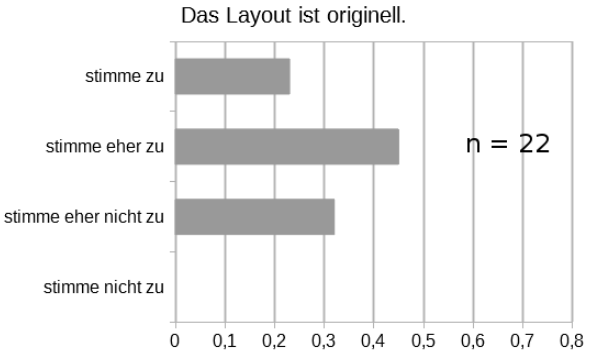

Abbildung 4.4: Auswertung der Originalität.

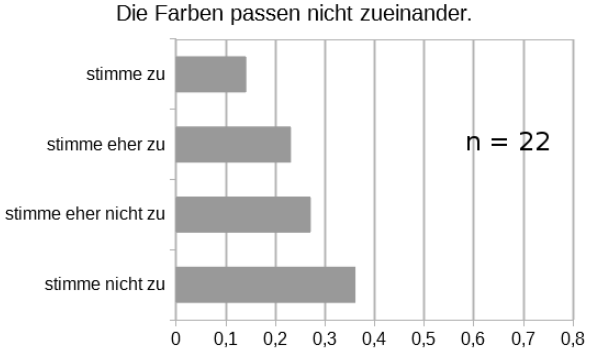

Abbildung 4.5: Auswertung des Farbzusammenspiels.

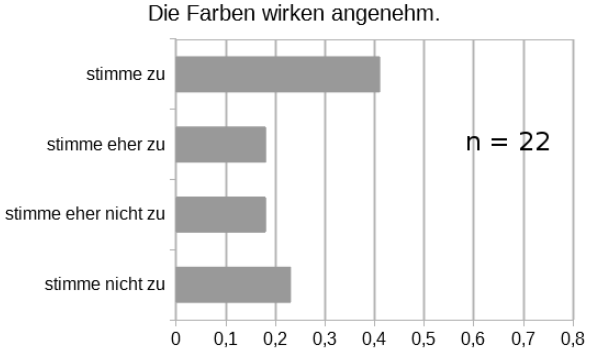

Abbildung 4.6: Auswertung der Farbannehmlichkeit.

eher nicht der Meinung sind, dass es überladen wäre. Zur Übersichtlichkeit fielen zwei der Antworten (9%) auf „stimme eher nicht zu“ und die gleiche Anzahl stimmte jeweils für überladen beziehungsweise eher überladen. Die Ergebnisse zur Übersichtlichkeit können insgesamt in 20 positive und 2 negative Stimmen gruppiert werden. Das entspricht Anteilen von 91% und 9%, während die Meinungen zur Überladenheit sich in 18 positive und 4 negative beziehungsweise 82% und 18% aufteilen.

Insgesamt kann man sagen, dass das Layout von einer deutlichen Mehrheit als übersichtlich eingestuft wurde.

Ebenfalls positive, wenn auch etwas weniger deutliche Ergebnisse, liefern die anderen beiden Aussagen „Das Layout wirkt professionell“ und „Das Layout ist originell“ auf den Abbildungen 4.3 und 4.4. Bei beiden ist die Aufteilung zwischen positiven und negativen Antworten fast gleich – die Professionalität betreffend 64% positiv und bei der Originalität 68%. Die genauen Aufteilungen zu „Das Layout wirkt professionell“sind 3 mal „stimme zu“, 11 „stimme eher zu“, 5 Stimmen für „stimme eher nicht zu“ und 3 für „stimme nicht zu“. Zur Originalität teilen sich die Antworten in der gleichen Reihenfolge in 5, 10, 7 und 0 auf. Die geringe Zahl an vollen Zustimmungen gerade zur professionellen Wirkung des Layouts könnte zusammenhängen mit den Meinungen zur Farbgebung (siehe Abbildungen 4.5 und 4.6).

Die verschiedenen Blau- und Lilatöne der Anwendung wurden aus insgesamt 22 Antworten von 14% der Befragten als nicht zueinander passend eingestuft (siehe Abbildung 4.5). Weitere 23% stimmen dieser Aussage eher zu. Dem gegenüber wählten 27% die Antwort „stimme eher

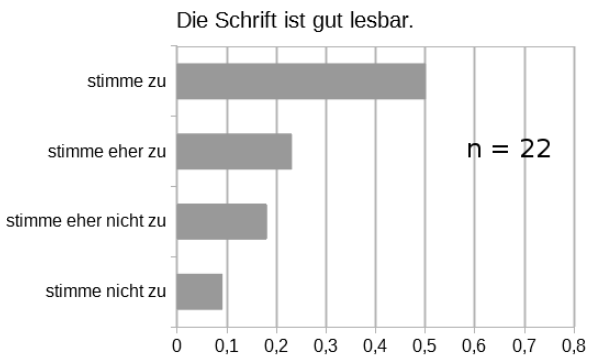

Abbildung 4.7: Auswertung der Lesbarkeit.

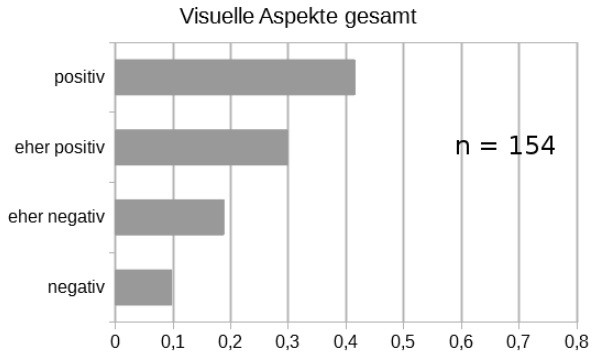

Abbildung 4.8: Gesamtauswertung der visuellen Aspekte.

nicht zu“ und 36% „stimme nicht zu“. Die Antworten zur Aussage „Die Farben wirken angenehm“ in Abbildung 4.6 waren etwas stärker polarisiert mit 23% am negativen und 41% am positiven Extrempunkt, während die beiden mittleren Antwortmöglichkeiten von jeweils 18% gewählt wurden. Die Tatsache, dass eine Mehrheit von 64% der Befragten zu letzterer Aussage eine eindeutige Meinung haben („stimme zu“ oder „stimme nicht zu“), lässt möglicherweise eine zu kräftige und damit ablenkende Farbgebung vermuten. Diese Hypothese wird im Zuge der Ergebnisdiskussion in Kapitel 4.2 näher besprochen.

Als letzter visueller Aspekt bleibt noch die Aussage „Die Schrift ist gut lesbar“, deren Ergebnisse Abbildung 4.7 zeigt. Hier fielen die Meinungen der Befragten überwiegend positiv aus, wobei 11 mit „stimme zu“und 5 mit „stimme eher zu“ reagierten. Die übrigen 6 verteilten sich mit 4 Stimmen auf eher nicht und mit 2 Stimmen auf nicht gut lesbare Schrift. Wie die Professionalität der Anwendung könnten auch diese Ergebnisse in Verbindung zur Farbgebung stehen.

Zuletzt ist in Abbildung 4.8 eine Gesamtübersicht aller abgegebenen Meinungen zu visuellen Aspekten zu sehen. Sie sind in die Kategorien „negativ“, „eher negativ“, „eher positiv“ und „positiv“ unterteilt. Bei Aussagen, wie „Die Farben passen nicht zueinander“, die aus Kontrollgründen negativ formuliert sind, fallen also die Antworten „stimme nicht zu “ und „stimme eher nicht zu“ in die Kategorien „positiv“ beziehungsweise „eher positiv„ und umgekehrt bei positiv formulierten Aussagen, wie „Die Farben wirken angenehm“. Das Diagramm zeigt, dass mit 72% positive und eher positive Eindrücke überwiegen, während negative und eher negative Meinungen respektive bei 10% und 19% liegen, was weiteres Verbesserungspotential vermuten lässt.

Neben dem visuellen Eindruck der Anwendung wurden den Testpersonen auch Aussagen zu Benutzbarkeit und Verständlichkeit der Bedienung vorgelegt, welche nachfolgend unter dem Begriff praktische Aspekte zusammengefasst wurden.

4.1.2 Praktische Aspekte

Der kontroverseste Punkt bei der Bedienung zeigt sich in den 22 gegebenen Antworten zu der in Abbildung 4.9 gezeigten Aussage. Sie lautete in der Umfrage „Die Software liefert ausreichend Informationen darüber, welche Eingaben möglich sind.“ und wurde aus Platzgründen in der Abbildung verkürzt formuliert. Positive und negative Meinungen halten sich fast die Waage,

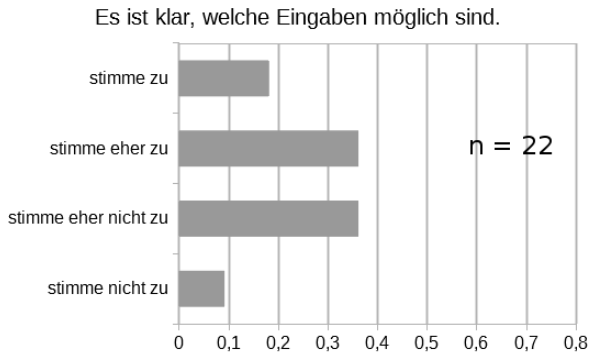

Abbildung 4.9: Auswertung der möglichen Eingaben.

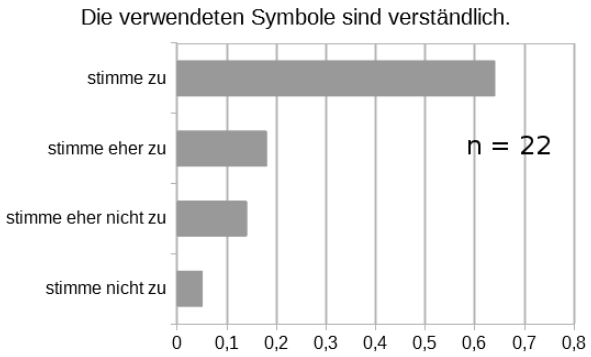

Abbildung 4.10: Auswertung der verwendeten Symbole.

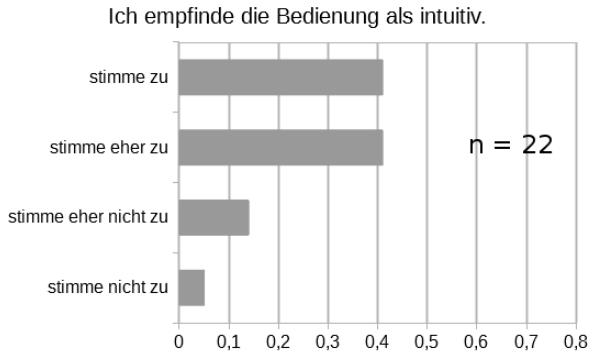

Abbildung 4.11: Auswertung der intuitiven Bedienung.

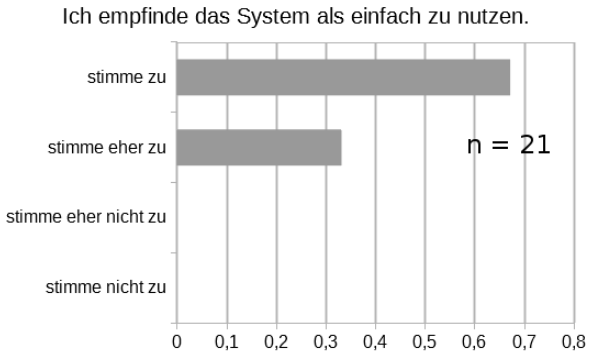

Abbildung 4.12: Auswertung der Einfachheit.

wobei „stimme zu“ mit 4 Stimmen gegenüber den 2 von „stimme nicht zu“ überwiegt und „stimme eher zu“ sowie „stimme eher nicht zu“ gleichauf liegen bei 8 Stimmen bzw. 32%. Dies zeigt dass im Bereich des Eingabemenüs durchaus Verbesserungsmöglichkeiten bestehen, wobei die verwendeten Symbole allerdings mehrheitlich als verständlich eingestuft wurden. Wie Abbildung 4.10 widerspiegelt gaben hier von 22 Befragten 64% die Antwort „stimme zu“ und 18% kreuzten „stimme eher zu“ an. Auf die negative Hälfte fielen 19% der Antworten verteilt auf 5% „stimme nicht zu“ und 14% „stimme eher nicht zu“.

Wie in Kapitel 3.7 beschrieben soll durch den Einsatz von Animationen ein intuitiv verständliches Umfeld geschaffen werden. Passend dazu zeigt Abbildung 4.11 die Verteilung der Antworten auf die Aussage „Ich empfinde die Bedienung als intuitiv.“. Es stimmten 9 der Testpersonen jeweils für „stimme zu“ und „stimme eher zu“. Eine Mehrheit von 82% liegt damit auf der positiven Seite der Antworten. „stimme eher nicht zu“ erhielt 3 Stimmen, der Anteil von „stimme nicht zu“ lag bei einer Stimme.

Das deutlichste Ergebnis der gesamten Umfrage lieferte allerdings die auf Abbildung 4.12 zu sehende Aussage „Ich empfinde das System als einfach zu nutzen.“, bei der keine befragte Person mit „stimme nicht zu“ oder „stimme eher nicht zu“ antwortete und von 21 Antworten 33% eher sowie 67% voll zustimmten. Intuitive und einfache Bedienbarkeit sind nach den Ergebnissen dieser Umfrage damit als Stärken der Anwendung zu verbuchen.

Da in der Navigationsansicht große Teile der visualisierten Bäume ausgeblendet werden (siehe Kapitel 3.2) besteht die Möglichkeit eines Orientierungsverlusts. Die Auswertung der

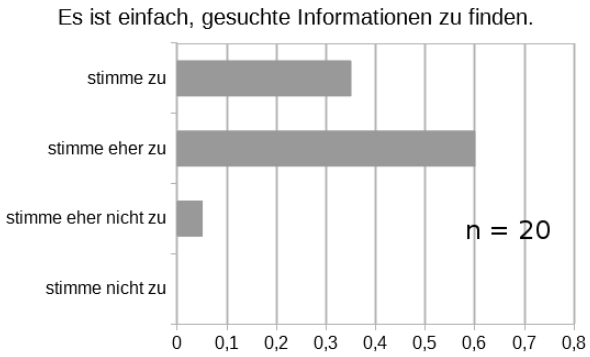

Abbildung 4.13: Ausw. der Informationsverfügbarkeit.

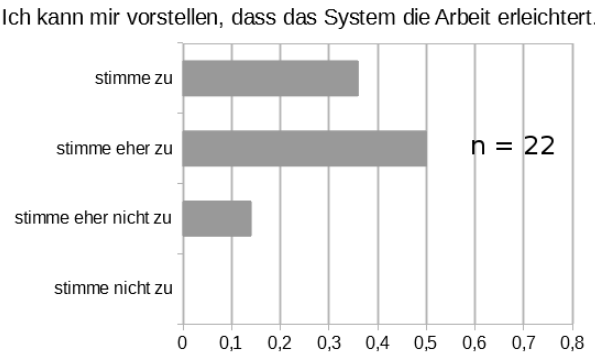

Abbildung 4.14: Auswertung der Arbeitserleichterung.

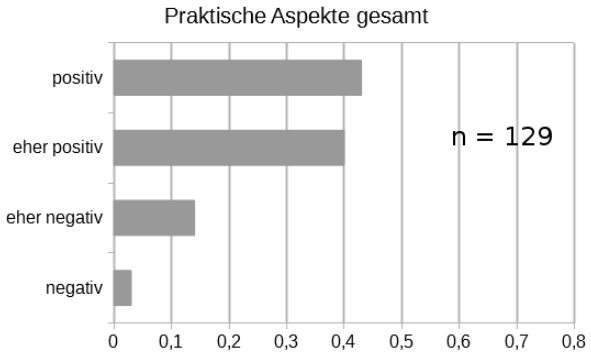

Abbildung 4.15: Gesamtauswertung praktischer Aspekte.

Aussage „Es ist einfach, gesuchte Informationen zu finden.“ legt aber nahe, dass dieser nicht eintritt. 7 der Befragten stimmten zu, dass die Informationssuche leicht fällt, 12 stimmten eher zu. Die verbleibende Stimme viel auf „stimme eher nicht zu“.

Die in Abbildung 4.14 gezeigten Daten befassen sich mit einem wichtigen Aspekt der Umfrage. Durch die Aussage „Ich kann mir vorstellen, dass das System die Arbeit im genannten Kontext erleichtert.“, welche in der Abbildung abgekürzt formuliert ist, sollte überprüft werden, ob die Testpersonen die Anwendung für nützlich hielten. Auch hier zeigen sich viele positive Meinungen. Keine befragte Person stimmte nicht zu, 3 stimmten eher nicht zu. Der Großteil der Antworten fiel mit 11 auf „stimme eher zu“und weitere 8 kreuzten an „stimme zu“. 86% der Testpersonen stehen damit der Nützlichkeit positiv gegenüber.

Abschließend soll auch hier noch einmal das Gesamtbild der Antworten zu praktischen Aspekten betrachtet werden. Trotz der gemischt ausfallenden Antworten zum Eingabemenü zeigt diese Gesamtauswertung in Abbildung 4.15 ein positiveres Ergebnis, als die Auswertung der visuellen Aspekte. Von 129 abgegebenen Antworten waren 18 eher negativ und 4 der fallen negativ aus. Auf der positiven Seite befinden sich 107 der Stimmen mit 51 eher positiven und 56 voll positiven. Das Ergibt sich zu einem Anteil von 17% negativen und 83% positiven Meinungen. Insgesamt überwiegen in beiden Bereichen positive Stimmen deutlich.

Abbildung 4.16: Animation beim gedrückt halten in Android.

4.1.3 Freitextantworten

Nach Abschluss des Multiple Choice Teils hatten alle Befragten die Möglichkeit, in Freitexteingabe Ideen für weitere Anwendungsmöglichkeiten des vorgestellten Konzepts zu äußern. Testpersonen nannten dabei verschiedene Bereiche, in denen ähnlich wie in der Medizin Abfolgen von Entscheidungen beziehungsweise Checklisten eine Rolle spielen. Darunter waren Wartung oder Diagnose im Allgemeinen, Kaufen von Gebrauchtwagen, Kreditvergabe und Hilfestellung für fachfremde Ersthelfer in Notsituationen. Auch als Lernbegleiter für Medizinstudenten während Praktika konnte einer der Befragten sich die Anwendung vorstellen und sogar eine Art Videospiel, bei dem Rätsel gelöst oder Verbrechen aufgedeckt werden, wurde als eine Einsatzmöglichkeit gesehen. Der mögliche Grund für die Vielfalt dieser Ideen wird im Folgenden zusammen mit den Ergebnissen des Multiple Choice Abschnitts näher diskutiert.

4.2 Diskussion der Ergebnisse

Auch wenn es sich bei der durchgeführten Umfrage nur um einen ersten allgemeinen Test der beschriebenen Anwendung handelt, können dennoch durchaus bereits Schlüsse daraus gezogen werden. Basierend darauf können anschließend Stärken weiter ausgebaut und Schwächen verbessert werden.

Beim Vergleich der Gesamtauswertungen von visuellen und praktischen Aspekten in den Abbildungen 4.8 und 4.15 zeigt sich ein insgesamt recht positives Ergebnis mit Verbesserungspotential, welches im visuellen Bereich etwas größer ist als im praktischen. Auch bei letzterem fallen aber nicht alle Ergebnisse ideal aus. So ist aus den Antworten in Abbildung 4.9 klar abzulesen, dass viele Nutzer nicht das Gefühl hatten, durch das Programm ausreichend auf mögliche Eingaben hingewiesen worden zu sein. Eine Person schrieb auch im Freitext, dass nicht zu erkennen sei, wie man das Eingabemenü aufruft. Gleichzeitig wurden die verwendeten Menüsymbole, wie Abbildung 4.10 zeigt, als gut verständlich wahrgenommen. Es muss demnach hauptsächlich eine Möglichkeit gefunden werden, auf die Existenz des Eingabemenüs hinzuweisen. Denkbar wäre eine ähnliche Lösung, wie die in Android standardmäßig enthaltene Animation, die in Abbildung 4.16 zu sehen ist. Beim längeren gedrückt Halten von Menüelementen breitet sich dort ein dunkler Kreis langsam aus. Auch ein kurzer Informationstext auf dem Bildschirm, der beim ersten Aufruf der Seite erscheint und die wichtigsten Eigenschaften erklärt könnte zu einer Verbesserung führen.

Im visuellen Bereich weisen die Ergebnisse zur Farbgebung in den Abbildungen 4.5 und 4.6 im Verhältnis zu den meisten anderen überprüften Aussagen relativ viele negative Antworten auf. Gerade zur Frage, ob die Farben angenehm wirken waren die Meinungen an den äußeren Enden des Spektrums konzentriert. Das lässt vermuten, dass die Farben vielen Befragten deutlich aufgefallen sind und etwas abgeschwächt werden sollten. In der Zukunft könnte man überprüfen, wie die Farbgebung wirkt, wenn anstelle von ausgefüllten Kreisen nur weiße Kreise mit farbiger Umrandung verwendet werden. Das könnte gut mit der im letzten Absatz

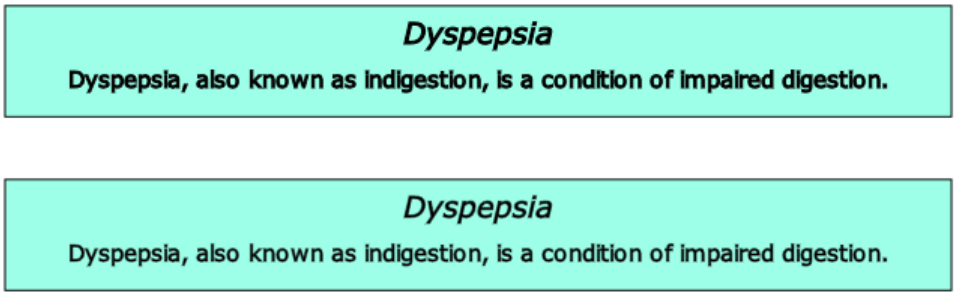

Abbildung 4.17: Vergleich zwischen dickerer und dünnerer Schrift.

erwähnten Idee (siehe Abbildung 4.16) beim gedrückt halten eines Knotens harmonieren, wobei sich der weißte Knoten langsam mit Farbe füllen würde. Insgesamt könnte ein verringerter Farbeinsatz zu einem professionelleren Gesamteindruck führen, der nach den bisherigen Ergebnissen in Abbildung 4.3 wenig volle Zustimmung erhalten hat und auch die Lesbarkeit der Schrift (Abbildung 4.7) dürfte davon profitieren.

Zum Schriftbild wurde seit der Umfrage bereits eine Verbesserung vorgenommen, bei der die Dicke der Buchstaben verringert wurde, wodurch sie weniger gedrängt wirken. Ein vorher-nachher Vergleich ist in Abbildung 4.17 zu sehen. Gezeigt ist ein Knoten in der Detailansicht, oben mit dickerer und unten mit der neuen, dünneren Schrift.

Anhand der Ergebnisse in den Abbildungen 4.1, 4.2 und besonders 4.12 zeigen sich mit Übersichtlichkeit des Layouts und der einfachen Bedienung die größten Stärken der Anwendung. Auch zur Aussage „Ich empfinde die Bedienung als intuitiv“ fallen die Meinungen überwiegend positiv aus, wie Abbildung 4.11 zeigt. In diesen Bereichen gibt es nicht mehr viel Verbesserungsbedarf, gerade die Übersichtlichkeit könnte aber von den zuvor genannten Änderungen an Farbgebung und Schrift immer noch positiv beeinflusst werden und durch bessere Hinweise auf das Eingabemenü dürfte sich auch eine intuitivere Bedienung erreichen lassen.

Der simpel gehaltene Aufbau der Anwendung, welcher sich auch in den positiven Eindrücken zur einfachen Bedienung widerspiegelt, führt zu vielseitigen Anwendungsmöglichkeiten. Das zeigen die verschiedenen Ideen, die von Befragten in der Freitexteingabe geäußert wurden. Einige davon wären sogar ohne große Änderungen möglich, wie der Einsatz als Lernbegleiter für Medizinstudenten oder Hilfestellung für fachfremde Ersthelfer. In erster Linie müssten dafür geeignete Entscheidungsbäume entwickelt werden.

Besonders hervorzuheben ist, dass ein großer Teil der Befragten den Einsatz nicht nur in anderen Bereichen, sondern auch im eigentlich beabsichtigten Einsatzgebiet für sinnvoll hielten, was sich in der Auswertung der Meinungen zur Arbeitserleichterung in Abbildung 4.14 offenbart. Dass nur 14% der Befragten eine eher negative Einstellung dazu hatten, während die restlichen 86% sich im positiven Spektrum bewegten, ist ein gutes Indiz für die Nützlichkeit von medizinischen Entscheidungsmodellen auf mobilen Geräten.

5
Verwandte Arbeiten

Zum Entstehungszeitpunkt dieser Arbeit konnten keine vergleichbaren wissenschaftlichen Arbeiten gefunden werden, die das Problem der interaktiven Visualisierung von medizinischen Leitlinien auf mobilen Geräten betrachten. Im Folgenden sollen daher verschiedene Arbeiten besprochen werden, die sich mit den relevanten Teilaspekten Baumdarstellung im Allgemeinen und auf mobilen Geräten sowie Visualisierung von Patientenakten auf mobilen Geräten befassen. Zum erleichterten Verständnis wird die in dieser Arbeit beschriebene Visualisierung nachfolgend als „TreeVis“ bezeichnet.

5.1 Baumdarstellung im Allgemeinen

Es gibt eine Vielzahl verschiedener Visualisierungen von Bäumen, die unterschiedliche Ziele verfolgen. Relevant für diese Arbeit sind dabei Ansätze, die sich um optimale Ausnutzung des verfügbaren Platzes bemühen. Wegen Ähnlichkeiten in der Baumnavigation und einer ebenfalls radialen Darstellung wird zuerst eine Umsetzung der sogenannten Sunburst-Darstellung besprochen.

5.1.1 Radiale Baumdarstellung Sunburst

Die von Stasko und Zhang [25] entwickelte Visualisierungsmethode für Bäume, genannt „Sunburst“, ist eine radiale Herangehensweise, die Knoten als Ringsegmente darstellt. Ein Implementierungsbeispiel von Ashenfelter [3] zeigt Abbildung 5.1. Die gezeigte Sunburst-Umsetzung wird vom Software as a Service Anbieter BigML, welcher eine Plattform für maschinelles Lernen anbietet, zur Visualisierung von Entscheidungsbäumen verwendet (Ashenfelter [2]). Der blaue Kreis in der Mitte repräsentiert die Wurzel und alle anderen Knoten sind als Ringsegmente mit der Wurzel als Zentrum dargestellt, wobei Knoten der gleichen Tiefe jeweils auf einem gemeinsamen Ring liegen. Die Kinder eines Knotens sind alle auf dem nächsten Ring liegenden Segmente, die mit ihm in Berührung sind. Wie man sieht, werden ebenso wie in TreeVis durch verschiedene Farben Knoten mit unterschiedlichen Bedeutungen voneinander differenziert. Auch mit dem Anteil des Rings, den ein Knoten einnimmt kann man Informationen, wie dessen Wichtigkeit übermitteln. Ähnlich zu TreeVis können auch bei der gezeigten

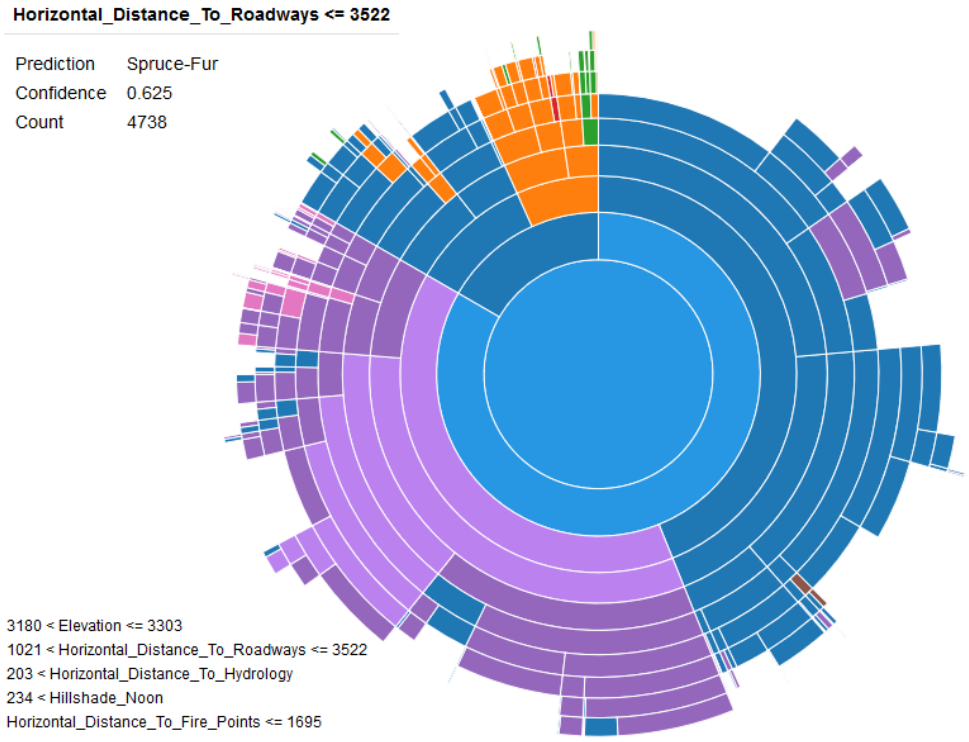

Abbildung 5.1: Radiale Darstellung eines Entscheidungsbaums von BigML, Quelle [3].

Sunburst-Umsetzung einzelne Knoten per Klick ausgewählt werden und es wird dann nur noch der diesem Knoten anhängende Unterbaum gezeigt. Durch die Kompaktheit der Darstellung ist es aber mitunter schwierig, die dabei ausgeführten Animationen auf dem Bildschirm nachzuvollziehen. Dieses Problem tritt durch das Ausblenden nicht relevanter Knoten bei der in den vorangegangenen Kapiteln beschriebenen Visualisierung nicht auf. Durch dieses Ausblenden und die kreisförmige bzw. rechteckige Knotenform können in TreeVis auch innerhalb von Knoten Texte angezeigt werden, was in Abbildung 5.1 aufgrund der gebogenen Knotendarstellung schwierig ist. Informationen zu ausgewählten Knoten sind dort daher außerhalb der Baumvisualisierung platziert.

Diese Sunburst-Umsetzung versucht eher einen allgemeinen Überblick über die Struktur des Baumes zu bieten wohingegen die Herangehensweise von TreeVis sich mehr auf Details einzelner Knoten konzentriert. Bei Sunburst handelt es sich um eine raumfüllende Visualisierung, die ohne eine Darstellung der Kanten zwischen Knoten auskommt. Auch auf mobilen Geräten gibt es bereits ähnliche Ideen zur kompakten Baumvisualisierung.

5.1.2 Baumvisualisierung auf mobilen Geräten

Eine mobile Lösung, die ebenso wie Sunburst raumfüllend ist und ohne Kantendarstellung auskommt, stellen Chhetri et al. [12] 2013 basierend auf vorangegangenen Arbeiten von Hao et al. [18] unter dem Namen Enhanced Radial Edgeless Tree (ERELT) vor. Die Knoten des Baumes werden als Polygone dargestellt, wobei die Wurzel in der oberen, linken Ecke liegt und der Baum sich diagonal nach unten, rechts ausbreitet. Die Nachfolger eines Knotens werden dabei immer an diesen angrenzend dargestellt. Es werden so viele Knoten visualisiert, dass der

gesamte Bildschirm gefüllt ist, was mehr Informationen auf einmal überträgt, aber dabei auch gedrängter wirkt als die von TreeVis gewählte Herangehensweise. Da nicht immer alle Knoten von ERELT gleichzeitig angezeigt werden können, kann wie in TreeVis auch hier durch Berühren von Knoten durch den Baum navigiert werden, wobei immer der berührte Knoten wie eine neue Wurzel behandelt und in die obere linke Ecke gezeichnet wird. Die Zahl der Knoten in Bäumen wächst häufig mit steigender Tiefe exponentiell, was bei ERELT dazu führt, dass die angezeigte Zahl von Nachfolgern jedes Knotens begrenzt ist und weitere Kinder ausgeblendet werden. Durch Nutzereingaben können dann diese versteckten Knoten sichtbar und dafür andere unsichtbar gemacht werden. TreeVis visualisiert immer nur einen Knoten und dessen Nachfolger, weshalb ein solches Problem erst bei vielen Kinderknoten auftreten würde, was in medizinischen Entscheidungsmodellen unwahrscheinlich ist. Anders als bei Sunburst besteht bei ERELT innerhalb der Knoten genug Platz, um Texte einzufügen. Diese sind aber wegen der diagonalen Orientierung des Baumes ebenfalls diagonal dargestellt, was die Lesbarkeit erschwert. In der Anzahl gleichzeitig anzeigbarer Knoten ist ERELT der hier beschriebenen TreeVis-Lösung voraus. Wie in Kapitel 3.2 beschrieben, ist es bei den von TreeVis fokussierten Entscheidungsmodellen aber weniger wichtig, viele Knoten auf einmal zu zeigen.

Auch bei Visualisierungen medizinischer Daten auf mobilen Geräten steht das Problem des begrenzten Platzes im Vordergrund. In der Vergangenheit wurden zum Beispiel bereits verschiedene Ansätze zur übersichtlichen Darstellung von Patientendaten vorgestellt.

5.2 Visualisierung von Patientenakten auf mobilen Geräten

Bereits mit der Entwicklung von Personal Digital Assistans (PDA), welche als Vorläufer der heutigen Smartphones betrachtet werden können, befassten sich verschiedene Arbeiten mit dem zuletzt genannten Problem. Inspiriert vom System LifeLines von Plaisant et al. [21], welches mit einer Java-Benutzeroberfläche Patientenakten auf Desktoprechnern verfügbar machte, stellten Ardito et al. [1] 2005 eine erste Darstellung solcher Akten auf den deutlich kleineren PDA-Bildschirmen vor, welche speziell auf die Bedürfnisse von Neurologen zugeschnitten war.

2006 veröffentlichte Chittaro [13] dann eine allgemeinere Herangehensweise mit dem Ziel für ein breiteres Feld von medizinischen Anwendungen nützlich zu sein. Während die in TreeVis visualisierten Patientendaten Untersuchungsergebnisse im Form von grünen und roten Umrandungen (positive und negative Ergebnisse) darstellen, handelt es sich bei den von Chittaro betrachteten Daten hauptsächlich um Messreihen von Blutdruck und Körpertemperatur sowie Zeitpunkten zu denen bestimmte Medikamente eingenommen wurden. Der Vorgestellte Prototyp war in der Lage, diese Daten in einem Koordinatensystem anzuzeigen, wobei Messpunkte als Balken in einem Balkendiagramm, oder als Punkte, die durch Linien verbunden werden, visualisiert wurden. Während TreeVis viele Platzprobleme durch Ausblenden von nicht relevanten Informationen löst, gab es hier eine Zoomfunktion mit der die Feinheit der angezeigten Graphen zwischen Minuten, Stunden, Tagen, Wochen und Monaten verstellt werden konnte. Über eine Eingabemaske wurden neue Daten mit Zeitstempel eingegeben, was aufwändiger aber auch flexibler ist als das Eingabemenü von TreeVis. Diese Arbeiten ähnelten TreeVis in der Grundidee, Patientendaten auf mobilen Geräten zugänglich zu machen. Sie befassten sich allerdings weniger mit Behandlungsabläufen, sondern eher mit dem reinen Anzeigen und Eintragen gesammelter Daten und hatten außerdem mit den auf PDA-Bildschirmen üblichen Auflösungen von 240×320 Pixeln ([13, S.1]) deutlich größere Einschränkungen hinzunehmen.

Die Idee der Visualisierung von Patientenakten auf mobilen Geräten wird auch auf Smartphones weiter verfolgt. So veröffentlichten Doukas et al. [15] 2010 eine mobile Anwendung für Android-Telefone, mit der Patientendaten und medizinische Aufnahmen, wie Röntgenbilder, in einer Cloud gespeichert und aus ihr abgerufen werden können. Auch TreeVis ist in der Lage mit Servern zu kommunizieren und Daten abzurufen, ist aber nicht für Bilder ausgelegt. Das System von Doukas et al. [15] achtet auch auf die Qualität der Internetverbindung und kann bei Verbindungsproblemen niedriger aufgelöste Versionen der Bilder laden, um den Datenstrom zu verringern. Um zu erörtern ob eine ähnliche Herangehensweise auch für TreeVis angewendet werden sollte, müssten in Zukunft noch Analysen des Datenverbrauchs durchgeführt werden. Die Autoren zeigen die Funktionstüchtigkeit ihrer Anwendung anhand von Testdatenübertragungen, geben aber zu bedenken, dass noch weitere Schritte ausstehen. Wie bei TreeVis handelte es sich auch hier noch um einen Prototyp mit Verbesserungspotential. So sollen im Bereich der Datensicherheit noch Verbesserungen vorgenommen werden und ein Test der Anwendung im Praxiseinsatz durchgeführt werden.

5.3 Entwicklungen im Gesundheitswesen

Allgemein gibt es bereits verschiedene Ansätze für Visualisierungen in der Medizin auch auf mobilen Geräten. Shneiderman et al. [24] gaben 2013 einen Überblick, wie das Gesundheitswesen durch interaktive Visualisierungen bereits verbessert wird und stellten eine Liste von Herausforderungen zusammen, die sie in der Zukunft stärker bearbeitet hoffen. Die Autoren sind der Überzeugung, dass die interaktive Visualisierung von Informationen tiefgreifende Änderungen unter anderem im Bereich der klinischen Gesundheitsvorsorge bringen wird. Als Motivation für ihre Arbeit stellen sie besonders einen Bericht des US Institute of Medicine [19] heraus, in dem es heißt, dass Informationsvisualisierung in Teilen der Klinischen Medizin nicht so fortschrittlich ist, wie in anderen Wissenschaftsdisziplinen.

Im Verlauf der Arbeit stellt sich heraus, dass in der Behandlung mit bisher genutzten Anwendungen Ansätze zur Visualisierung von Patientengeschichten erkundet werden, wobei sich allerdings noch kein Standard durchgesetzt hat. Es wird beispielsweise versucht, die Änderungen des Patientenzustands im Laufe der Zeit zu visualisieren. Auch zur Analyse, ob bei Behandlungsverfahren klinische Richtlinien eingehalten werden, wobei große Datenmengen zu bewältigen sind, gibt es Visualisierungsversuche, die teilweise schon zur erfolgreichen Aufdeckung von Behandlungsfehlern geführt haben.

Man kann erkennen, dass die interaktive Darstellung von Entscheidungsmodellen, wie sie in der vorliegenden Arbeit beschrieben wurde, noch nicht im Fokus der Entwicklung steht. Diese Fällt eher unter die erste der von Shneiderman et al. [24] beschriebenen Herausforderungen für die Zukunft, welche das rechtzeitige Anbieten von Informationen für viel beschäftigte Ärzte in einem passenden Format fordert.

6
Fazit

In dieser Arbeit wurde die Entwicklung einer interaktiven webbasierten Visualisierung von medizinischen Entscheidungsmodellen auf mobilen Geräten beschrieben. Ziel dabei war es, eine übersichtliche Anwendung zu implementieren, die einfach zu nutzen ist und einen Mehrwert besonders bei der ärztlichen Arbeit in Kliniken bietet. Zur Entstehungszeit der Anwendung gibt es noch keine etablierten Lösungen für das genannte Problem.

Bei der Implementierung kamen Maßgeblich die Sprachen HTML und Javascript zum Einsatz, wobei besonders die Verwendung der Bibliothek D3 im Mittelpunkt stand, durch die sich Modelldaten und HTML-Anzeige miteinander verknüpfen lassen und vielfältige Hilfsfunktionen zur Behandlung von Datenstrukturen sowie einfacher Einsatz von Animationen geboten werden.

Die Evaluierung wurde in Form einer Umfrage unter verschiedenen Berufsgruppen durchgeführt, wobei sich insgesamt positive Ergebnisse ergaben. Viele der Befragten sahen eine mögliche Arbeitserleichterung durch den Einsatz der vorgestellten Anwendung im medizinischen Umfeld. Besonders die einfache Nutzung und das übersichtliche Layout stießen auf Zustimmung. Auch das Finden gesuchter Informationen fiel einem Großteil der Testpersonen leicht und die Bedienung wurde insgesamt für intuitiv befunden. Als verbesserungswürdig stellten sich die Farbgestaltung und das Eingabemenü heraus, auf welches durch die Anwendung nicht eindeutig genug hingewiesen wird. Mögliche Verbesserungen dieser Punkte wurden diskutiert und sind in der Zukunft weiter zu bewerten. Tests unter Praxisbedingungen im klinischen Umfeld sind noch nicht durchgeführt worden. Es bleibt zu klären, ob und wie die Anwendung reibungslos in bestehende Krankenhaussysteme integriert werden kann. Dabei sind auch Fragen des Datenschutzes und Patientengeheimnisses zu berücksichtigen.

Insgesamt ist ein übersichtlicher und einfach zu nutzender Prototyp entstanden, der von Testpersonen überwiegend positiv bewertet wurde und durch kleine Änderungen weiter zu verbessern ist. Er kann als Grundstein für weiterführende Arbeiten zur Visualisierung von Entscheidungsmodellen auf mobilen Geräten dienen.

Literaturverzeichnis

[1] Carmelo Ardito, Paolo Buono, und Maria Francesca Costabile. The challenge of visualizing patient histories on a mobile device. In *IFIP Conference on Human-Computer Interaction*, pages 942–945. Springer, 2005.

[2] Adam Ashenfelter. A new way to visualize decision trees, . URL https://blog.bigml.com/2013/04/19/a-new-way-to-visualize-decision-trees/. Stand 19.03.2017.

[3] Adam Ashenfelter. Bigml tree - cover data, . URL http://bl.ocks.org/ashenfad/5234819. Stand 20.03.2017.

[4] Mike Bostock. Arc padding III, . URL https://bl.ocks.org/mbostock/99f0a6533f7c949cf8b8. Stand 18.03.2017.

[5] Mike Bostock. Dokumentation zu d3-ease, . URL https://github.com/d3/d3-ease/blob/master/README.md#easeExpOut. Stand 18.03.2017.

[6] Mike Bostock. Pie centroid, . URL https://bl.ocks.org/mbostock/9b5a2fd1ce1a146f27e4. Stand 18.03.2017.

[7] Mike Bostock. Population pyramid, . URL https://bl.ocks.org/mbostock/4062085. Stand 18.03.2017.

[8] Mike Bostock. Dokumentation zu d3-hierarchy, . URL https://github.com/d3/d3-hierarchy/blob/master/README.md#tree. Stand 18.03.2017.

[9] Mike Bostock. Treemap, . URL https://bl.ocks.org/mbostock/6bbb0a7ff7686b124d80. Stand 20.03.2017.

[10] Mike Bostock. Wrapping long labels, . URL https://bl.ocks.org/mbostock/7555321. Stand 18.03.2017.

[11] Mike Bostock, Shan Carter, und Matthew Ericson. At the national conventions, the words they used. URL http://www.nytimes.com/interactive/2012/09/06/us/politics/convention-word-counts.html. Stand 18.03.2017.

[12] Abhishek P Chhetri, Kang Zhang, und Eakta Jain. Erelt: a faster alternative to the list-based interfaces for tree exploration and searching in mobile devices. In *Proceedings of the 6th International Symposium on Visual Information Communication and Interaction*, pages 54–63. ACM, 2013.

[13] Luca Chittaro. Visualization of patient data at different temporal granularities on mobile devices. In *Proceedings of the working conference on Advanced visual interfaces*, pages 484–487. ACM, 2006.

[14] Refsnes Data. W3.css home. URL https://www.w3schools.com/w3css/. Stand 25.03.2017.

[15] Charalampos Doukas, Thomas Pliakas, und Ilias Maglogiannis. Mobile healthcare information management utilizing cloud computing and android os. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 1037–1040. IEEE, 2010.

[16] Robert Fleischmann, Julian Duhm, Hagen Hupperts, und Stephan A Brandt. Tablet computers with mobile electronic medical records enhance clinical routine and promote bedside time: a controlled prospective crossover study. *Journal of neurology*, 262(3):532–540, 2015.

[17] Google. Material design – Introduction. URL https://material.io/guidelines/#introduction-principles. Stand 18.03.2017.

[18] Jie Hao, Kang Zhang, und Mao Lin Huang. Relt–visualizing trees on mobile devices. In *International Conference on Advances in Visual Information Systems*, pages 344–357. Springer, 2007.

[19] Institute of Medicine. *Health IT and Patient Safety: Building Safer Systems for Better Care*. The National Academies Press, Washington, DC, 2012. ISBN 978-0-309-22112-2. doi: 10.17226/13269. URL https://www.nap.edu/catalog/13269/health-it-and-patient-safety-building-safer-systems-for-better.

[20] Lawrence Page, Sergey Brin, Rajeev Motwani, und Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[21] Catherine Plaisant, Rich Mushlin, Aaron Snyder, Jia Li, Daniel Heller, und Ben Shneiderman. Lifelines: using visualization to enhance navigation and analysis of patient records. In *Proceedings of the AMIA Symposium*, page 76. American Medical Informatics Association, 1998.

[22] Edward M. Reingold und John S. Tilford. Tidier drawings of trees. *IEEE Transactions on software Engineering*, (2):223–228, 1981.

[23] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.

[24] Ben Shneiderman, Catherine Plaisant, und Bradford W Hesse. Improving healthcare with interactive visualization. *Computer*, 46(5):58–66, 2013.

[25] John Stasko und Eugene Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 57–65. IEEE, 2000.

[26] World Wide Web Consortium W3C. Scalable vector graphics (SVG) 1.1 (second edition). URL `https://www.w3.org/TR/SVG/`. Stand 18.03.2017.