Recherchebericht AGB-10: Graph Browser (SWP10-9)

Inhaltsverzeichnis

1. Begriffe

Komorbiditätsnetzwerk

Das Komorbiditätsnetzwerk ist ein gewichteter Graph zur Darstellung der Beziehung von Krankheiten. Hierbei repräsentieren Knoten Krankheiten, die durch ihren ICD-Code angegeben werden. Das Gewicht der Kanten signalisiert die Signifikanz der Korrelation zwischen den Knoten. Fährt der Nutzer mit dem Mauszeiger über die Knoten, so erhält er Informationen wie den Namen der Krankheit. Durch Anklicken wird man zu weiteren Links geführt.

Zur Darstellung werden Radial, Simple und Circle View verwendet. Zwei Krankheiten stehen in Beziehung, wenn sie auffällig häufig gemeinsam, das heißt im Krankheitsverlauf einer Person, auftreten.

Data Transformation

Zur Bearbeitung der Quelldaten, die die nötigen Informationen über die Krankheiten und ihre Beziehung untereinander enthalten, müssen diese zunächst transformiert, angeordnet und aufgelistet werden. Ein weiterer Grund für die Notwendigkeit der Konvertierung der Daten aus der Quelle in ein geeignetes Zielformat ist, dass die zugrunde liegende Datei zu groß ist, um sie direkt im Graphen einzubinden.

Visual mapping

Visual Mapping ist ein Prozess zur Erschließung und visuellen Darstellung der Daten. Die Abbildung der Daten in einen Graphen ermöglicht dem Benutzer schneller Zusammenhänge zu erkennen und einen besseren Überblick zu gewinnen. Der ICD-Code der einzelnen Krankheiten wird hierbei auf die Knoten des Graphen abgebildet, und die Beziehung der Krankheiten auf die Kanten. Das Kantengewicht ist ein Maß für die Signifikanz der Beziehung.

View Transformation

Durch die View Transformation soll eine Änderung der Darstellung des Graphen ermöglicht werden. Durch Interaktion mit dem Benutzer kann die Darstellungsart des Graphen gewählt werden. Dabei kann er zwischen Radial, Simple und Circle View wählen.

Clustering

Unter Clustering versteht man die Daten, basierend auf ihren Zusammenhängen, in Gruppen zu teilen. Dabei unterscheidet man zwischen Hard- und Soft-Clustering. Beim Hard-Clustering ist jedes Element in genau einem Cluster, so dass die Cluster sich nicht überlappen.

Beim Soft-Clustering können Elemente auch in mehreren Clustern sein. Eine Überlappung der Cluster ist nicht ausgeschlossen.

BorderFlow-Bibliothek

Die BorderFlow-Bibliothek implementiert einen allgemeinen Algorithmus für das Clustering der Graphen. Er verwendet ausschließlich lokale Informationen für das Clustering und erreicht ein Soft-Clustering des gegebenen Graphen.

Prefuse Library

Prefuse ist eine für Java geschriebene Programmbibliothek. Mit Hilfe der Prefuse Library

lassen sich Daten modellieren, visualisieren sowie Informationen verwalten und bearbeiten.

Es bietet optimierte Datenstrukturen für Tabellen, Grafiken und Bäume, eine Vielzahl von Layouts und visuelle Codierungstechniken als auch Unterstützung für Animation, dynamische Abfragen, integrierte Suche, und Datenbank-Konnektivität.

ICD - International Statistical Classification of Diseases and Related Health Problems

(deutsch: Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme)

Der ICD Code ist ein Diagnoseklassifikationssystem der Medizin. Er dient zur Verschlüsselung der Diagnosen. Er kann aus einer dreistelligen Nummer bestehen oder aus einer Verschlüsselung bestehend aus vier Buchstaben und an fünfter Stelle eine Ziffer.

Im Laufe der Jahre wurde der ICD-Code immer weiter entwickelt. Die derzeit gültige Ausgabe ist ICD-10. Für unser Projekt verwenden wir ICD-9.

Viewer

Der Viewer dient zur Visualisierung des Eingabegraphen in verschiedenen Darstellungsformen: radial view, simple view, etc.. Der Benutzer hat nur Lesezugriff darauf.
Er, der Viewer, ermöglicht die Exploration des Netzwerkes und basiert auf der Prefuse-Library. Zusätzlich werden Funktionen für das Zoomen und das Erhalten weiterer Informationen über die Krankheit beim Anklicken der betreffenden Knoten geboten.

Eingabegraphen

Der angezeigte Graph basiert auf den Eingaben des Nutzers. Der Benutzer muss Alter, Hautfarbe, die Anzahl der Stellen des ICD-9 Codes eingeben, und den ICD-Code der Krankheit, zu der er Information haben will. Aus diesen Informationen wird der benötigte Graph erstellt.

2. Konzepte

Modularer Aufbau

Unter dem modularen Aufbau versteht man die Einteilung des Projekts in verschiedene Abschnitte während des Programmierens. Dies bringt viele Vorteile. Bereits programmierte Abschnitte (Module) können so jederzeit später wieder verwendet oder abgeändert werden. Weiterhin lässt sich so ein Gesamtprojekt sehr gut in Gruppen einteilen. Alle Module funktionieren als Ganzes unabhängig voneinander und lassen sich so einfach zu einem Projekt zusammenfügen.

MVC (Model-View-Controller)

MVC ist ein Architekturmuster zur Strukturierung während der Softwareentwicklung. Model (Datenmodell), View (Präsentation) und Controller (Steuerung) werden dabei strikt getrennt.

Während im View die Eingabe erfolgt, wird diese nach dem Einlesen durch den Controller an das Model übermittelt. Hier werden dann die benötigten Werte berechnet. Diese werden dann als Ergebnis wieder an das View zurückgeleitet und angezeigt. Dieses Modell gehört zum modularen Aufbau.

Java

Java ist eine objektorientierte Programmiersprache und Bestandteil der Java Technologie. Sie wurde von der Firma Sun Microsystems entwickelt. Java Programme sind plattformunanhängig, d.h. man kann sie auf jedem Betriebssystem ausführen, da für jedes eine Java VM (Virtual Machine) existiert.

3. Beschreibung der Applikation

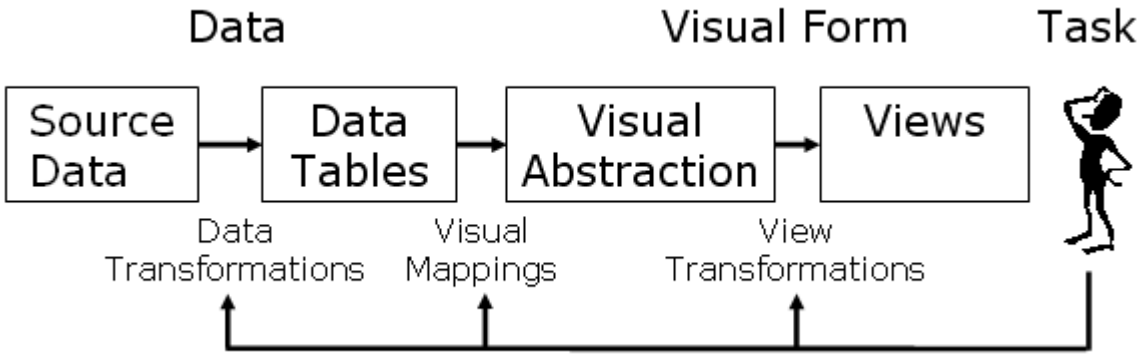
Einführung

Prefuse ist eine Programmbibliothek, die verschiedenste zusammenhängende Informationen verwalten und bearbeiten kann. Es ist möglich, diese Informationen in Java basierten Anwendungen auf verschiedenste Weise darzustellen. Auch können durch die Interfaces Veränderungen, sowohl an der Darstellung als auch an den Daten selbst, vorgenommen werden. Man hat zuerst eine Standardklasse und passt anschließend alle Abweichungen vom Standard an.

Prefuse besteht dabei aus drei aufeinander aufbauenden Schichten: "Data Tables", "Visual Abstraction" und "Interactive Views".

In einfachen Worten läuft es wie folgt ab: Zuerst werden die Quelldaten eingelesen. Anschließend werden diese Daten, wie später benötigt, transformiert, angeordnet und aufgelistet. Diese Daten-Tabellen werden nun in ein Datenmodell zur Visualisierung transformiert. Im letzten Schritt werden die Daten durch verschiedene Ansichten wiedergegeben. Die Interaktion des Benutzers auf die Ansicht der Daten kann nun verschiedenste Veränderungen in den Transformations-Schritten ergeben.

(Weitere Details in der Übersicht von Prefuse [letzte Seite].)



Architekturkonzept - Im Detail

Data Tables

Im Paket *prefuse.data* bietet Prefuse Klassen und Strukturen um die Daten in das Programm aufnehmen zu können. Unter anderem gibt es dort die Klassen *Graph*, *Table*, *Tree* und *SpannigTree*. Außerdem gibt es auch einfachere Elemente um neue Strukturen bilden zu können: *Edge*, *Node* und *Tuple*.

Weiterhin werden im Paket *prefuse.data.io* und Paket *prefuse.data.io.sql* Klassen bereitgestellt, um die benötigten Infomationen aus üblichen Dateiformaten wie *GraphML* oder *TreeML* bzw. auch aus einer *MySQL*-Datenbank auszulesen und in die genannten Strukturen zu

übertragen. Ebenso ist es möglich, die bearbeiteten Daten später wieder in eine Datei zu schreiben.

Erweiterungsmöglichkeit (Data Tables):

Die Datensätze, die von unserem Produkt bearbeitet werden müssen, stehen unter verschiedenen Gesichtspunkten auf unterschiedlichen Wegen miteinander in Verbindung. Daher muss eine komplexere Struktur als die des einfachen Graphen gefunden werden. Zugleich muss auch die Möglichkeit geschaffen werden, diese Struktur in einen Graphen umzuwandeln. Dem entsprechend ist es ebenso notwendig, das Einlesen an die neue Struktur anzupassen.

Visual Abstraction

Um die Informationen auch optisch ansprechend darstellen zu können, werden weitere Daten wie Form, Farbe, Gruppierungen und Ähnliches benötigt. Dies alles wird in der Klasse *Visualization* zusammengefasst. Durch sogenannte *VisualItems* bietet Prefuse die Möglichkeit durch viele verschiedene Funktionen komplette Datensätze optisch gestalten zu können. Ebenso ist es auch möglich, eigene Effekte zu erstellen und einzubinden.

Interactive Views

Um die Informationen als auch die Darstellung selbst zu visualisieren, wird die Klasse *Display* angeboten, eine abgeleitete Klasse von *swing.JComponent*. Beispiele dafür sind die Variationen in der Anordnung der Daten, sowie die Datenmanipulation. Somit kann die Sicht als Komponente in der grafischen Schnittstelle SWING von Java eingebunden werden.

Man kann mehrere Sichten (Displays) instanziieren und im Konstruktor eine Instanz der Klasse *Visualization* angeben, sodass die Daten daraus ohne Weiteres dargestellt werden können. Die Klasse *Display* selbst, und damit die Darstellung, kann man dann noch weiter anpassen. Indem die einzelnen Methoden und Klassen - zum Beispiel zur Darstellung und Anordnung - angepasst, überschrieben oder noch zusätzliche hinzugefügt werden, wird die benötigte Anpassung erreicht. Genauso ist es in der Klasse *Visualization* der Fall.

Es werden wieder Klasse und Funktionen angeboten um Drag&Drop oder das Zoomen und Bewegen in der Bildebene zu ermöglichen. Diese können in jeder Instanz der Klasse *Display* individuell hinzugefügt werden.

Erweiterungsmöglichkeit (Interactive Views):

Die Anordnungsmöglichkeit und auch das Anzeigen von Zusatzdaten muss erweitert bzw. an den Themenbereich angepasst werden. Zum Beispiel sollte man, wenn man mit der Maus über die Darstellung der Datensätze fährt, bereits Informationen erhalten, die in verschiedenen Formen erscheinen. Durch Anklicken sollte man beispielsweise zu weiteren Links geführt werden oder ein zusätzliches Fenster mit näheren Informationen angezeigt bekommen.

Analyse der Prefuse Demos

GraphView

Diese View stellt Knoten dar, die an hand einer Feder verbunden sind. Dabei ist es möglich verschiedenste Koeffizienten beliebig zu verändern. Unter anderem kann Kraft und Länge der Federn, ebenso wie die Gravitation zwischen den Knoten beeinflusst werden. Die Ansicht ermöglicht Zoomen, sowie Verschieben der Ansicht per Maus. Es ist möglich, eigene Datensätze zu laden oder verschiedene voreingestellte zu wählen.

Anpassungen gab es in der Klasse *prefuse.data.event.TupleSetListener*.

TupleSetListener wurde geändert, damit man einen einzelnen Knoten fixieren kann. Wenn man einen weiteren Knoten anklickt, wird dieser fixiert und der vorige ist wieder beweglich.

RadialGraphView

Die Demo RadialGraphView stellt eine Menge von Knoten und deren Beziehungen zueinander dar. Hierbei kann ein Knoten ausgewählt werden, welcher in die Mitte der Darstellung gerückt wird. Die Beziehungen zu diesem Knoten werden hierarchisch in Kreisen um den ausgewählten Knoten herum angeordnet.

Anpassungen gab es in den Klassen *prefuse.control.ControlAdapter*, *prefuse.action.GroupAction* und *prefuse.action.assignment.ColorAction*. *ColorAction* wurde geändert, um Füllfarbe und Text der Knoten zu modifizieren und wird in den Klassen *TextColorAction* und *NodeColorAction* verwendet. *GroupAction* wird verwendet um den Spannbaum zu erzeugen. *ControlAdapter* wird aufgerufen wenn man mit dem Mauszeiger über einen Knoten fährt. In diesem Fall wird ein kleines Fenster geöffnet, in dem der Name angezeigt wird.

Data Mountain

Die Demo DataMountain stellt verschiedene Bücher und Computerspiele als Bilder dar. Wenn man einen Doppelklick auf einem Bild ausführt, öffnet sich der entsprechende Link auf Amazon im Standardbrowser. Außerdem lassen sich die Elemente beliebig verschieben, wobei andere Elemente vom bewegten Bild dynamisch abgestoßen werden. Zusätzlich ändert sich die Bildgröße je nachdem, wo auf der Bildfläche sich die Elemente befinden.

Die Daten dazu werden durch übliches Einlesen einer Tabelle bereitgestellt. Prefuse wurde für die Demo an drei wichtigen Stellen erweitert. Zum einen wurde die Klasse *ControlAdapter* abgeleitet und so umgeschrieben, dass sich die Elemente markieren und verschieben lassen und durch doppeltes Anklicken der entsprechende Link öffnet. Desweiteren wurde ebenso die Klasse *ForceDirectetedLayout* erweitert, die für die Simulation physikalischer Kräfte zwischen den Elementen zuständig ist, sodass sich die einzelnen Bilder beim Bewegen gegenseitig Abstoßen. Abschließend wurde die Klasse *SizeAction* bearbeitet, sodass sich die Größe der einzelnen Bilder der Position anpasst.

Die drei erweiterten Klassen wurden der Klasse *Display* an den entsprechenden Stellen zugewiesen und die Standardelemente überschrieben. Die Änderungen betreffen somit nur die visuelle Schicht.

Quellen

http://prefuse.org/

► User Manual, API Documentation, FAQ, ...

http://en.wikipedia.org/wiki/Prefuse

► Begriffserklärung, Übersicht

http://de.wikipedia.org/

►Begriffserklärungen

http://bis.informatik.uni-leipzig.de/AxelNgonga/BorderFlow

http://olat.informatik.uni-leipzig.de/olat/auth/1%3A1%3A0%3A0%3A0/

http://www.dimdi.de/static/de/klassi/diagnosen/index.htm

http://www.bio.ifi.lmu.de/webfm_send/435

Anhang: prefuse, package guide