HAW HAMBURG

BACHELORTHESIS
Nelli Welker

Integration von Visual-Analytics-Komponenten in die Plattform Open Vigil

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Nelli Welker

Integration von Visual-Analytics-Komponenten in die Plattform Open Vigil

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Marina Tropmann-Frick
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 19. August 2019

Nelli Welker

Thema der Arbeit

Integration von Visual-Analytics-Komponenten in die Plattform Open Vigil

Stichworte

Visual Analytics, Open Vigil, React

Nelli Welker

Title of Thesis

Integration of Visual Analytics Components in the platform Open Vigil

Keywords

Visual Analytics, Open Vigil, React

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Mittlerweile stehen uns so viele Datenmengen zur Verfügung, dass wir gar nicht wissen, wohin wir damit sollen. Es fehlt uns an optimierten Analystechniken, die domänenübergreifend Daten verarbeiten, analysieren und damit Erkenntnisse schließen. So ist es auch in der Pharmakovigilanz. Es liegen viele Daten vor, doch um fundiertes und qualitativhochwertiges Wissen daraus zu gewinnen ist ziemlich kompliziert. Gesundheitswesen und die damit verbundenen Gefahren bzw. Risiken etc sind sehr wichtig und sollten so gut es geht unterstützt werden. So habe ich mich auch zu diesem Projekt entschieden. Es hat einen größeren Zweck und findet auch in der realen Welt an Zuspruch. Zudem habe ich mich auch für dieses Thema entschieden, da ich bis zu diesem Zeitpunkt noch keine Berührungspunkte mit der Frontend-Entwicklung hatte und großen Ehrgeiz hatte, mich damit zu beschäftigen und zu lernen.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, einen ganz bestimmten Bereich der Web-Applikation OpenVigil 2.1. zu analysieren und durch visuelle Aspekte zu einer besseren Verwendung zu verhelfen. Es geht nicht darum, Open Vigil komplett von Neuem auf- und umzubauen. Dies würde den Rahmen und die momentan dafür zur Verfügung stehenden Ressourcen sprengen. Das Ziel war es, sich mit dem Projekt Open Vigil auseinander zu setzen und unter Visual Analytics Aspekten Teile der Web-Applikation zu erweitern. Die Bereiche, die erweitert werden sollten, sind zum einen die Start-Suchseite und zum anderen die Suchkriteri.n

1.3 Gliederung

Diese Arbeit gliedert sich in sechs Kapitel. Nach der Einleitung, die die Motivation, die Zielsetzung und die Gliederung darlegt, folgt das zweite Kapitel mit der Pharmakovigilanz und Open Vigil. Es wird nicht nur auf die Bedeutung der Pharmakovigilanz eingegangen, sondern auch auf deren Relevanz und den Nutzen in der Medizin. Zudem werden die Entstehung und die Entwicklung von Open Vigil dargelegt und die verwendeten Technologien erläutert. Ein weiterer Aspekt in dem zweiten Kapitel ist, in welchem Umfang die Web-Applikation erweitert werden soll. Das dritte Kapitel geht auf die Entwicklung von Visual Analytics, den einhergehnden Prozess und den Einsatz in der Pharmakovigilanz bzw. Medizin ein. In dem vierten Kapitel wird zuerst der IST-Zustand des zu verändernden OpenVigil 2.1 Projekts erläutert, bevor der SOLL-Zustand mit den dazugehörigen Anforderungen erläutert wird. Der Anforderungsanalyse folgt im fünften Kapitel die Umsetzung der Entwicklung. Im ersten Schritt wird auf den Aufbau und die Konfiguration der Entwicklungsumgebung und den damit einhergehenden Problemen und Fehlern eingegangen. Danach wird der Entwurf der Search- und Result-Komponenten beleuchtet, dem die Prototypen folgen. Den Prototypen folgt die Beschreibung der Implementierung, die auf die einzelnen Komponenten eingeht und . Zum Schluss dieser Arbeit wird die Arbeit kritisch betrachtet und rückblickend eine Diskussion über die Erweiterung OpenVigil 2.1. geführt.

2 Pharmakovigilanz und Open Vigil

2.1 Pharmakovigilanz

Die Pharmakovigilanz (PV) ist ein sehr bedeutendes Gebiet. Es ist laut der World Health Organization (WHO) definiert als die Wissenschaft und als die Aktivitäten, die die Detektion, Bewertung, das Verständnis und die Prävention von Nebenwirkungen (Adverse Effects) oder anderendrogenbedingten Problemen umfassen[7]. Als Reaktion auf den 1961 aufgedeckten Arzneimittelskandal um das Medikament Contergan hat die WHO ein Programm zur internationalen Drogenüberwachung eingerichtet. Contergan enthält den Wirkstoff Thalidomid, welcher als Schlaf- und Beruhigungsmittel dienen sollte und auch Schwangerschaftsübelkeit mindern sollte. Doch zu der Zeit war nicht bekannt, dass es auch erhebliche Auswirkungen auf das Ungeborene hatte. So wurde das Medikament von 1957 bis 1961 als das erste bromfreie Schlaf- und Beruhigungsmittel ohne größere Nebenwirkungen von dem Stolberger Herstellerunternehmen Grünenthal vermarktet, bis 1961 Grünenthal nach einem Zeitungsartikel in der Welt am Sonntag, der den Verdacht über eine fruchtschädigende Wirkung des Arzneistoffes Thalidomid publik machte, endlich Contergan aus dem Handel nahm. Dem Bundesverband Contergangeschädigter e.V. sind ca. 5000 contergangeschädigte Kinder bekannt. Vor allem um solche Katastrophen zu verhindern, ist es wichtig, neu auf den Markt kommende und auch bereits vorhandene Medikamente mit deren Nebenwirkungen zu überwachen und zu dokumentieren und auch eventuelle Wechselwirkungen mit verschiedenenen Medikamenten untereinander festzuhalten. Als Konsequenz dieser Tragödie wurde 1968 durch die WHO das *Programme for International Drug Monitoring* (PIDM) gegründet, welches systematisch Informationen über schwerwiegende unerwünschte Arzneimittelwirkungen während der Entwicklung und insbesondere nach der publiken Bereitstellung der Arzneimittel sammeln sollte. Seit 2016 sind über 120 Länder dem WHO-Programm beigetreten, um die Patientenversorgung zu verbessern und die Arzneimittelsicherheit in einem höheren Maß gewährleisten zu können[7]. Die PIDM-Mitgliedstaaten übermitteln Berichte über Nebenwirkungen von Arzneimitteln, sogenannte Individual Case Safety Reports (ICRs),

über die globale WHO-Datenbank *VigiBaseTM*. Verwaltet wird diese globale Datenbank von dem WHO Collaborating Centre for International Drug Monitoring, auch bekannt als Uppsala Monitoring Centre (UMC). 2014 verzeichnete die Datenbank über zehn Millionen gemeldete Nebenwirkungen. Anhand dieser Daten versucht das UMC potenzielle Risiken für die Arzneimittelsicherheit zu erkennen. Pharmakovigilanz dient dem Risikomanagement rund um die Anwendung von Medikamenten. Sie umfasst nicht nur Nebenwirkungen von Arzneimitteln sondern auch Schäden an Verpackungen oder Präparaten und präventive Maßnahmen gegen Behandlungfehler. Zu einem wichtigen Bereich der PV zählen die Meldungen von UAWen, um präventiv gegen noch unbekannte, nicht nachgewiesene Nebenwirkungen oder Wechselwirkungen zwischen mehreren Arzneimitteln vorzugehen. Denn neue Medikamente unterlaufen, bevor sie zugelassen werden, mehreren klinischen Studien, um mögliche Nebenwirkungen zu entdecken und die Sicherheit des Medikaments zu gewährleisten. Doch nicht alle unerwünschten Arzneimittelwirkungen (UAW) treten in dieser Studienzeit auf und werden somit nicht erfasst, da dies aber einen verhältnismäßig kleinen Zeitraum umfasst und auch nur eine geringe selektierte Menge an Probanden aufweist, die auch nur bedingt dem Behandlungskollektiv nach der Markteinführung entsprechen. Langzeitwirkungen entgehen meist solchen Überwachungen, sodass man der klinischen Unbedenklichkeit eines Medikaments nur bedingt vertrauen kann. Ein Restrisiko von eventuell verborgen gebliebenen möglichen UAWen bleibt bestehen. Spezifische, individuelle, von Mensch zu Mensch unterschiedliche auftretende Empfindlichkeiten lassen sich nicht vollständig in einer Studie aufdecken. Zudem können auch bei der Einnahme verschiedener Medikamente Wechselwirkungen zwischen diesen entstehen, die bei neuen Medikamenten nicht unbedingt bei Studien aufgedeckt werden. Dafür ist es umso wichtiger auch nach der Medikamentenzulassung sicherzustellen, dass neue Medikamente weiterhin überwacht werden, um unerwünschte Wirkungen umgehend zu entdecken, zu beurteilen und zu dokumentieren. Das deutsche Arzneimittelgesetz (AMG) beschreibt die rechtliche Grundlage für die Zulassung, die Herstellung und die fortlaufende Überwachung eines Arzneimittels[3]. Für die Berichterstattung sind Ärzte durch ihre Berufsordnung dazu verpflichtet, UAWs entweder an die Arzneimittelkommission der deutschen Ärzteschaft oder das Bundesministerium für Arzneimittel und Medizinprodukte (BfArM) zu melden. Es werden Verdachtsfälle mit unerwünschten Reaktionen gemeldet, wenn diese im Zusammenhang mit der Gabe eines anderen Arzneimittels stehen (Spontanberichte). Um solche Meldungen so früh wie möglich zu erhalten und dementsprechend schnellstmöglich reagieren zu können, sind Spontanmeldesysteme wichtig und gewinnen immer mehr an Bedeutung. Je früher über potenzielle Risikofaktoren informiert werden kann, desto schneller können Maßnahmen zum Schutz

von behandelnden Personen vorgenommen werden. Auch Privatpersonen können UAWs über verschiedene Meldebögen über die Webseiten der BfArM[2] oder der Arzneimittelkommission der deutschen Ärzteschaft (AkdÄ) melden[1]. Ein Spontanmeldesystem besteht aus solchen gesammelten Reports der eingegangenen Meldungen, die in Deutschland über Arzneimittelkommissionen und Pharmaunternehmen eingehen. Als Ergänzung kann nach UAW auch gezielt bei Krankenhausaufnahmen, schweren Krankheitsbildern oder bei bestimmten Patientengruppen gesucht werden. In diesem Zusammenhang gibt es in Deutschland spezielle Pharmakovigilanzzentren. In der EU sind Pharmaunternehmen zur elektronischen Meldung von UAWen an die nationalen Arzneimittelbehörden verpflichtet, die wiederum die Daten an die europäische Arzneimittelagentur weiterleiten. Diese Arzneimittelbehörde sorgt dafür, dass diese Daten einheitlich erfasst, gesammelt und ausgewertet werden und über das Netzwerk EudraVigilance zentral und koordiniert archiviert werden können. Um eine einheitliche Klassifizierung zu ermöglichen, wurde die Medical Dictionary for Regulatory Activities (MedDRA) entwickelt. Diese ermöglicht eine internationale einheitliche Kommunikation zwischen Arzneimittelzulassungsbehörden und Pharmaunternehmen.

Die wichtigsten Ziele der PV gemäß der WHO sind die Optimierung der Arzneimittelsicherheit bezüglich der Anwendung, mögliche Beurteilung von Nutzen, Schaden, Wirkungsgrad und dem Risiko von Arzneimitteln sowie die Förderung der Aufkläung in der PV und die öffentliche Kommunikation. Der Pharmakovigilanz fehlt es an einem einheitlichem Standard für die Analyse von Medikamenten und Nebenwirkungen. Es gibt Programme für Analysen, die aber entweder nicht frei zugänglich sind oder deren Handhabung zu kompliziert ist. Zudem unterscheiden sie sich in der Datenbasis, mit der diese arbeiten, sodass es keine einheiltleiche übereinstimmende Kommunikation oder Vergleichbarkeit von Daten oder Analysen möglich ist.

2.2 Open Vigil

Es liegen zwar viele Gesundheitsdaten vor, doch um diese informativ auszuwerten und zu analysieren, fehlt es akademischen und medizinischen Einrichtungen an leicht anwendbaren Analysewerkzeugen, sodass diese für alltägliche klinische Zwecke verwendet werden können. Dies zeigt, wie wichtig ein Tool ist, mit dem Pharmakovigilanz in Kliniken, Praxen und allgemein im Gesundheitswesen effizient anwendbar ist. Um diesem

Problem entgegenzuwirken, wurden verschiedene Pharmakovigilanz-Analystetools entwickelt, die helfen, Erkenntnisse aus den gesammelten Daten zu erlangen. Die WHO bietet z.B. das Analyseprogramm *VigiSearch*/*VigiLyze* an, welches auf den Datensätzen von VigiBase basiert. Dies steht aber lediglich den Mitgliedstaaten des WHO-PIDM zur Verfügung. Desweiteren gibt es die Portale http://www.adrreports.eu/de/index.html und http://www.vigiaccess.org/, die zwar öffentlich zugänglich sind, aber weder umfangreiche Filter- und Extraktionsoptionen noch visuelle Analysen anbieten. Man sieht, dass eine Recherche trotz vieler vorhandener Daten für den normalen Allgemeinmediziner geschweige denn für Menschen, die nicht in einem medizinischen Umfeld tätig oder nicht autorisiert sind, erschwert bis unmöglich ist. Ein anderes öffentliches Datenanalysetool ist das AERS Spider (Adverse Event Reporting System) (http://www.chemoprofiling.org/AERS/index.html), welches interaktive Untersuchungen von Assoziationen zwischen Arzneimittel und UAWen ermöglicht. AERS Spider verwendet AERS-Daten (Adverse Event Reporting System) der Food and Drug Administration (FDA). Ein weiteres Analyseprogramm ist Open Vigil (OV). OV ist ein Werkzeug zum schnellen Nachschlagen von Arzneimittelaktionen, die die Arzneimittelsicherheit unterstützen und verbessern sollen. So tragen sie zu einer besseren Behandlung von Patienten bei. Open Vigil ist eine Webapplikation, die der Analyse von Pharmakovigilanz-Daten dient. Diese wurde 2009 von R. Böhm et al. der Christian-Albrechts-Universität Kiel entwickelt und 2011 als frei zugängliches Analyse-Programm releast. Open Vigil ist auf den Webservern der CAU Kiel installiert und frei unter GNU General Publich License (GPLv3) verfügbar. In Abbildung 2.1 ist der erste Teil der Startseite von OpenVigil abgebildet. Es verwendet eine relationale SQL Datenbank, um Pharmakovigilanz-Daten zu speichern. Open Vigil wurde u.a. entwickelt, um auf rohen Datensätzen (wie die AERS-Daten der FDA), die meist verschiedene Missstände aufweisen, wie Unvollständigkeit, Unrichtigkeit oder auch Duplikate, Säuberungs- und Selektions-Maßnahmen durchzuführen. Damit anschließend qulatitativ hochwertige und aussagekräftige Analysen vollzogen werden können. Open Vigil liegt in verschiedenen Versionen und Formaten vor, die hauptsächlich auf den Daten der FDA basieren. Die möglichen Ausgabe-Formate der Resultate, die zur Auswahl für den User stehen, sind bei allen OV-Versionen die folgenden: HTML, CSV und Excel.

2.2.1 OpenVigil 1

OpenVigil 1 ist das erste öffentliche Release gewesen. Die letzte Version, die öffentlich zur Verfügung steht, ist die OpenVigil v1.2.7 (s. Abb. 2.2) (http://openvigil.pharmacology.uni-

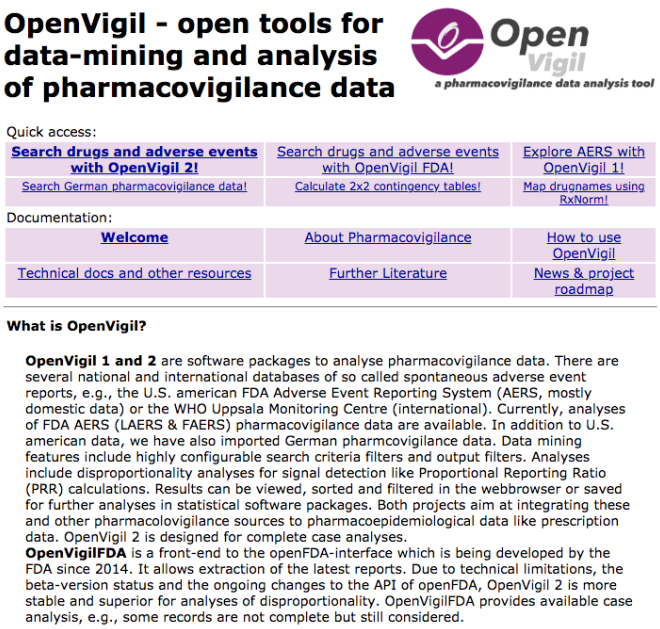


Abbildung 2.1: OpenVigil - Open Tools für Data-Mining und Analyse von Pharmakovigilanz Daten. [5]

kiel.de/openvigil-current.php?cd=vs). Diese verwendet PHP, MySQL 5.5. und die AERS-FDA-Datensätze vom 06.10.2003 bis zum 31.12.2013. Sie ist unter der folgenden URL zu finden: http://openvigil.pharmacology.uni-kiel.de/openvigil-current.php. Sie bietet Zugriff auf Pharmakovigilanz-Daten und deren Analysen durch Queries (z.B. das Extrahieren von individuellen Sicherheits Reports (ISR) oder Disproportionalitätsanalysen auf selektierte Fälle). OV 1 arbeitet auf den Rohdaten der FDA AERS.

Außer OV 1 laufen alle darauffolgenden Versionen mit gereinigten Datensätzen.

2.2.2 OpenVigil FDA

OpenVigilFDA ist ein webbasiertes Userinterface (http://openvigil.pharmacology.uni-kiel.de/openvigilfda.php) für die AERS Database der FDA, um mit Hilfe der openFDA-Online-API Sicherheitsberichte über Arzneimittel- und unerwünschte Ereignisse zu extrahieren und zu analysieren (s. Abb. 2.3). OpenFDA hat als Ziel, einen sauberen und gepflegten Zugang zur zugrunde liegenden AERS zu ermöglichen. Zu den angebotenen Analysen gehören Disproportionalitätsanalysen, um u.a. das Medikament zu identifizieren, das ein neues unerwünschtes Ereignis hervorruft. Zudem wird darauf abgezielt, zwei

Abbildung 2.2: Open Vigil 1.2.7

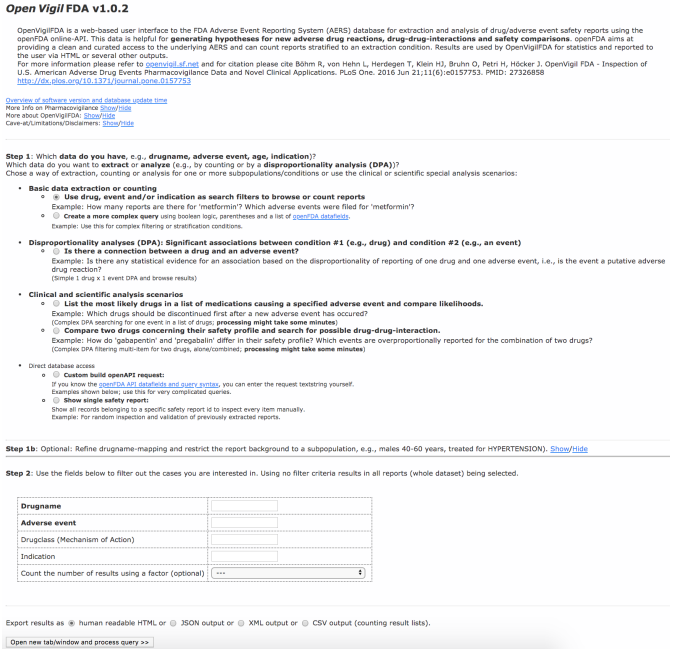


Abbildung 2.3: Open Vigil FDA 1.0.2

Medikamente bezüglich des Sicherheitsaspektes zu vergleichen, zwei Medikamente bezüglich möglicher Drug-Drug-Interaktionen zu untersuchen und die Relevanz der Ergebnisse durch die Identifikation und das Beseitigen von Störfaktoren zu verbessern. OpenVigil FDA ist das erste öffentlich verfügbare Tool, mit dem die Ergebnisse der Pharmakovigilanz direkt auf klinische Probleme im Alltag anwendbar sind[10]. OpenVigil FDA besteht aus einem einzigen Programm-File, welches in der Skriptsprache PHP (PHP Hypertext Preprocessor) geschrieben ist. OV FDA verarbeitet die User-Anfragen und gibt ein oder mehrere erzeugte Queries an die online API der FDA, um die Reports zu erhalten und Kontingenztabellen oder Disproportionalitätsanalysen zu ermitteln. Die Ergebnisse werden in einer der folgenden Formate dargestellt: HTML, JSON, XML oder CSV.

2.2.3 OpenVigil 2.1

Auf OV 1 folgte OV 2.0, welches im Gegensatz zu OV1 mit gesäuberten Daten arbeitet. OV2.0 kann Daten extrahieren, filtern und analysieren. OV2 arbeitet mit FDA AERS Daten, die die Reports mit den Drugs und den Nebenwirkungen enthalten, und mit der Drugbank (drugbank.ca), die eine umfangreiche Quelle für bioinformatische und chemische Informationen ist und detaillierte Informationen über Arzneimittel und Zielverbindungen liefert. OV benutzt den Begriff "Drug"für eine Substanz in einem pharmazeutischen Produkt, das biologisch aktiv und für die therapeutische Wirkung verantwortlich ist. Die Bezeichnung "Drug"darf nicht verwechselt werden mit anderen Bedeutungen wie illegalen Drogen oder einem gebrauchsfertigen pharmazeutischen Produkt wie einer Pille. Da OpenVigil ursprünglich für die US-amerikanischen Pharmakovigilanzdaten entwickelt wurde, werden Arzneimittel gemäß United States Adopted Name (USAN) benannt, die eindeutige und einfache, auf pharmakologischen oder chemischen Beziehungen basierenden Namen darstellen. Diese unterscheidet sich von dem Internationalen Freinamen (International Nonproprietary Name, INN), der von der WHO für ein Arzneimittelwirkstoff vergeben wird. Dieser ermöglicht eine einheitliche Kommunikation über gleichartige Präparate auf internationaler Ebene. Seitdem Open Vigil für das Mapping der Drugnames zu USAN auf externe Datenbanken angewiesen ist, besteht ein Risiko von Fehlzuordnungen. Zudem gibt es in den FDA-Rohdaten auch andere Medikamentennamen wie British Adopted Name (BAN), die eine Veschmelzung von zwei verschiedenen Drugs zu einem ermöglichen (z.B. Cotrimoxazol als Kombination von Trimethoprim und Sulfamethoxazol). Im Gegensatz zu OV 1 filtert OV 2 mehrdeutige Berichte, bei denen z.B. die Drug- und Pharmaproduct-Namen falsch geschrieben sind. Zudem konvertiert OV2 einige der

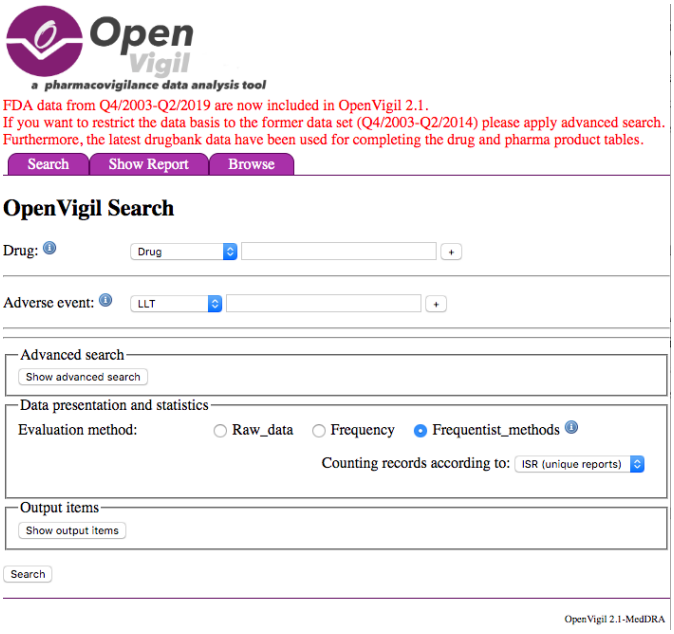


Abbildung 2.4: Open Vigil 2.1

Attribut-Werte, wie Alter, Dosierung und Dauer der Therapie vom Free-Text- in ein Uniform-Format. Die aktuelle Version von OV 2.1. enthält alle Daten der FDA in dem Zeitraum von Oktober 2003 bis Juni 2019 und ist unter der folgenden URL aufrufbar: https://www.is.informatik.uni-kiel.de/pvt/OpenVigilMedDRA17/search/ (s. Abb. 2.4).

Diese Arbeit beschäftigt sich mit Open Vigil 2.1. MedDRA von 2017.

2.2.4 Daten

Es gibt verschiedene nationale und internationale Datenbanken, die Daten aus sogenannten spontanen Berichten über unerwünschte Ereignisse enthalten. Wie z.B. das amerikanische FDA Adverse Event Reporting System (AERS) oder das WHO UMC. Die Lebensmittelüberwachungs- und Arzneimittelbehörde der Vereinigten Staaten (FDA, Food and Drug Administration) stellt ihre Daten quartalsweise öffentlich zur Verfügung (AERS, Adverse Event Reporting System). Die FDA Datenquelle weist zwar eine große Datenmenge auf, deren Reports aber nicht immer komplette Informationen aufweisen. So fehlen manchmal Daten (z.B. demographische Patienten-Daten) oder sind inkorrekt. Zudem stellt die korrekte Zuordnung des Arzneimittels zu einem eindeutigen Wirkstoff

ein Problem dar. Die Datenbank-Quellen, wie die Drugbank oder die Drugs@FDA, sind zwar frei verfügbar, sichern aber keine 100%ige Übereinstimmung zu. Alle Datensätze, mit denen Open Vigil arbeitet, kommen hauptsächlich aus dem AERS der FDA und ein weiterer Teil von den deutschen Pharmakovigilanzdaten. Die AERS Die Auswertungen innerhalb von Open Vigil beschäftigen sich nur mit den gereinigten Daten (ab OV 1.2.), die der FDA gemeldet werden. Die Datenbank der aktuellen Version enthält Daten der FDA ab dem 29.10.2003 bis zum 31.12.2018.

3 Visual Analytics

Big Data zeigt, wie unabdingbar und auch wichtig für uns Daten sind und wie mit diesen sinnvoll umgegangen werden sollte. Big Data umfasst nicht nur das Datenvolumen (Volumen) und die große Bandbreite an Datentypen (Variety), also die Vielfalt der Daten, sondern auch die Geschwindigkeit (Velocity), mit der die Datenmengen generiert und transferiert werden, die Daten-Qualität (Validity), die u.a. die Bereinigung und Selektion der Daten voraussetzt, und schlussendlich die Verwertbarkeit der Daten, also dem Wert der Daten (Value), der aus den durch Big Data erschlossenen Daten erhalten werden kann. Um der Informationsüberflutung bedingt durch die Masse an Daten entgegenzuwirken und effiziente und neue Informationen zu entdecken, sind Maßnahmen und Werkzeuge erforderlich, die die Möglichkeit darbeiten, diese zu analysieren. Denn die Möglichkeit, Daten zu sammeln und zu speichern, ist enorm gewachsen.

Die bisherigen Analysetechniken wie Statistiken und Data Mining wurden unabhängig von Visualisierungs- und Interaktions-Techniken entwickelt. Doch da diese Felder eher einen begrenzten Umfang darstellten, erweiterte sich dieser Bereich grundlegend in das, was wir heute visuelle Analyse-Forschung bezeichnen[14]. Ein grundlegender Auslöser dafür war, dass es notwendig gewesen ist, sich von der konfirmatorischen Datenanalyse (Überprüfung/Nachweis von Zusammenhängen) zu lösen und sich zu einer explorativen Datenanalyse (Aufzeigen von Zusammenhängen/Strukturen, die dann überprüft werden konnten) zu entwickeln. Das Potential wurde entdeckt, den User in den Knowledge Discovery- und Data Mining-Prozess zu integrieren, indem ihm effektive Visualisierungstechniken, Interaktionsmöglichkeiten und Wissenstransfer geboten wurden. Dies erweiterte nicht nur die Informations-Visualisierung sondern auch das Data Mining durch neue Techniken und viele neue wichtige Forschungsmöglichkeiten in einem erheblichen Umfang[11]. Es ermöglicht, Menschen in einem frühen Stadium des Daten-Analyse-Prozesses einzubeziehen, sodass diese ihre Flexibilität, Kreativität und Hintergrundwissen mit der enormen Speicher- und Verarbeitungs-Kapazität der heutigen Rechenleistung kombinieren können, um Kenntnisse aus komplexen Problemstellungen erzielen zu können. 2004 wurde der Begriff Visual Analytics (VA) von P. C. Wong und J. Thomas

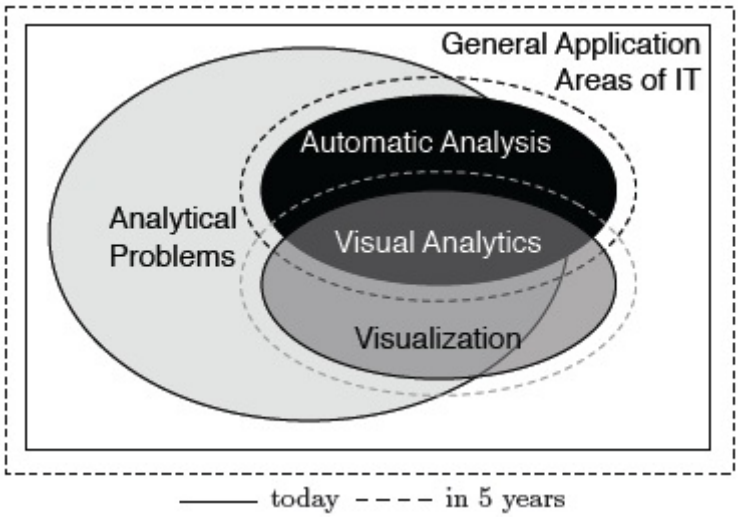


Abbildung 3.1: Visual Analytics Problems

eingeführt[17]. Laut Wong und Thomas sind Visual Analytics ein bewährter Ansatz, um sowohl die Kunst der menschlichen Intuition als auch die Wissenschaft der mathematischen Folgerung zu kombinieren, um Muster wahrzunehmen und daraus Wissen und Erkenntnisse abzuleiten[17]. Es ist eine Möglichkeit, abstrakte visuelle Methaphern mit einem menschlichen Informations-Diskurs zu kombinieren, der es ermöglicht Erwartetes zu erkennen und Unerwartetes zu entdecken mit einem massiven und sich dynamisch verändernden Informationsraum. Visual Analytics ist zwar der wissenschaftlichen und Informationsvisualisierung entsprungen, ist aber ein multidisziplinäres Feld und umfasst aber Technologien aus vielen anderen Bereichen wie Wissensmanagement, statistische Analyse, Kognitionswissenschaften und viele mehr[12]. Visual Analytics war ursprünglich für das Lösen von Problemen gedacht, die allein von automatischen oder visuellen Analysen nicht gelöst werden konnten. Im Laufe der Anwendungen demonstrierte Visual Analytics aber ihre Erweiterbarkeit und Anwendbarkeit auf einen viel umfangreicheren Bereich. Nach Keim gibt es zwei Problem-Klassen, die *Analytical Problems* und die *General Application Areas of IT*, die herangezogen werden können, um den Umfang/Nutzen/Zweck von Visual Analytics zu beschreiben[15]. Bei den analytischen Problemen (s. Abb. 3.1) gibt es eine Logik, die es ermöglicht, sie rational zu bewerten. Es bedeutet nicht, dass alle Probleme lösbar sind. Denn die Rechenleistung oder personelle Ressourcen sind nicht unbegrenzt, um die Probleme in der verfügbaren Zeit und den verfügbaren Methoden zu lösen. Probleme der allgemeinen Anwendungsbereiche der IT, zu denen auch die analytischen Probleme gehören, müssen nicht unbedingt schwere Probleme sein. Es können

auch leichte Probleme wie das Versenden und Empfangen einer E-Mail sein. Um diese Problem-Klassen zu lösen, werden diese drei verschiedenen Methoden angewendet:

Automatic Analysis Diese Methoden werden angewendet, wenn Mittel zur Verfügung stehen, um die Qualität von Lösungsmöglichkeiten zu messen und zu vergleichen. Diese funktionieren hingegen nicht, wenn Algorithmen in lokalen Optima eingeschlossen sind, die nichts mit der global besten Lösung zu tun haben. Da viele reale Probleme nicht genau zu definieren sind, kann die kostengünstigere Variante der automatischen Analyse schwer angewendet werden und die Relation zwischen der Ein- und Ausgabe ist für den Analytiker oft unklar, sodass offen bleibt, ob das System vertrauensvoll ist oder nicht. Zudem kann ein automatisierter Algorithmus seine Analyselösung nicht auf Probleme dynamisch anpassen.

Visualization Visualisierungsmethoden dagegen bedienen sich dem menschlichen Hintergrundwissen und der Intuition für das Lösen des vorliegenden Problems. Bei kleinen Datensätzen liefern diese Methoden gute Resultate, wohingegen sie Grenzen bei großen Datenmengen aufweisen. Die visuelle Methodik ist kostenspieliger, da spezialisierte Experten vergütet und zudem auch in der verwendeten Software geschult werden müssen. Eines der Ziele der visuellen Analyse von neuen Systemen ist die Verkürzung von Einarbeitungszeiten durch benutzerfreundliche Schnittstellen.

Visual Analytics Diese Methoden kombinieren die Vorzüge beider Techniken, indem diese sowohl die intelligenten Algorithmen und die Rechenleistung moderner Rechner als auch das menschliche Verständnis nutzen, um ein optimales Resultat zu erzielen.

Nicht alle Datenanalyse-Programme sind präzise in Worten geschweige denn in mathematischen Formeln ausdrückbar und nicht in allen Fällen lässt sich sagen, ob die gefundene Lösung auch die optimale ist. Dies ist von vielen Faktoren abhängig, wie von dem Analyse-Task, den verfügbaren Daten und der Evaluations-Maßnahme. Schlussendlich muss man sich mit einem Kompromiss zwischen den zur Verfügung stehenden Ressourcen, die für bessere Lösungen notwendig sind, und dem potentiellen Schaden, der durch eine suboptimale Lösung verursacht wird, zufrieden geben. Abblidung 3.2 zeigt das Potential von VA bezogen auf die drei verschiedenen Problem-Klassen, indem die Wirksamkeit der Analyse und der Grad der Interaktion in Relation gesetzt werden. Man sieht, dass VA eine vielversprechende Lösung für Probleme ist, die weder von automatisierter noch von explorativer Analyse effektiv gelöst werden können.

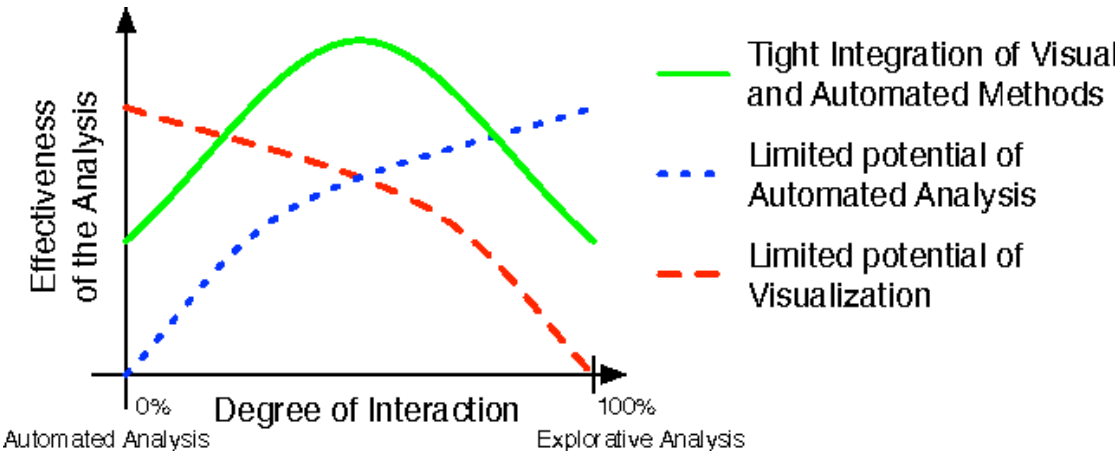


Abbildung 3.2: Potential of Visual Analytics

VA beschleunigt die Leistung des Users beim Ausführen eines Tasks, sodass es mehr als genug die Verwendung von VA rechtfertigt. Zudem führt eine Verbesserung der Effektivität des Users auch dazu, dass der Einsatz von VA auch auf allgemeine Anwendungsbereiche der IT ausgeweitet wird. So wird VA nicht nur in den oben beschriebenen Problemfällen angewendet, sondern auch in ßimplenÄnwendungsfällen wie z.B. beim Archivieren von E-Mails.

Visual Analytics birgt viele Vorteile. Unter anderem hat Visual Analytics die Fähigkeit, viele unserer täglichen Prozesse zu transformieren und diese effizienter und effektiver zu gestalten. Es verschafft Usern in vielen Situationen einen überblick. Mit Kombination von automatisierten Analyseergebnissen und Argumentationsanalysen ist eine Skalierung auf größere und komplexere Probleme möglich. Visual Analytics dient dazu, die durch automatisierte Analysen erzielten Ergebnisse effektiv zu versmitteln und einem breiten Publikum zugänglich zu machen. Zudem sei die Qualität der Lösungen erwähnt. Denn ohne verfügbare Mittel geben sich User auch mit einer vielleicht schlechteren Lösungs-Qualität zufrieden. Doch durch VA ist es möglich, die Qualität zu optimieren. Durch das iterative Vorgehen des VA Prozesses können nicht nur bessere sondern auch vertrauenswürdigere Analyse-Resultate erzielt werden.

3.1 Der Visual Analytics Prozess

Der Visual Analytics Prozess vereint automatische und visuelle Analyse-Methoden, die durch die Userinteraktion eng miteinander gekoppelt sind. Charakteristisch für den VA

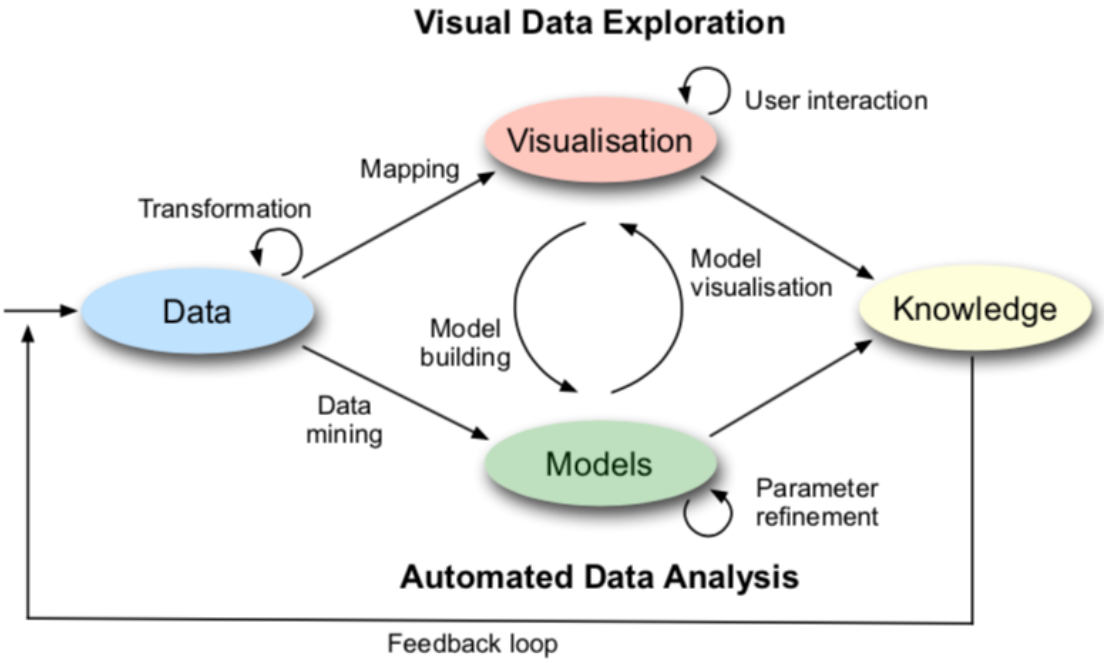


Abbildung 3.3: Visual Analytics Prozess [14]

Prozess ist die Interaktion zwischen den Daten, der Visualisierungen, des Daten-Modells und dem User mit dem Ziel der Erkenntnisgewinnung. In Abbildung 3.3 sind die verschiedenen Zustände, die durch die Ovale abgebildet werden, und die Übergänge, die die Pfeile symbolisieren, des VA Prozesses dargestellt. Der Visual Analytics Prozess beginnt damit, dass die vorliegenden meist heterogenen Daten erst einmal vorverarbeitet und transformiert werden müssen, bevor diese integriert werden können. Zu den Vorverarbeitungsphasen können die Datensäuberung, Selektion oder das Gruppieren der vorliegenden Daten gehören. Anschließend sind die transformierten Daten in verschiedene Repräsentationen überführbar und können dann exploriert werden. Auf diese Transfomation der Daten folgen zwei mögliche Methoden: Zum einen kann der Daten-Analyst sich zuerst für die visuelle und zum anderen zuerst für die automatisierte Daten-Analyse entscheiden. Wenn zuerst der Weg der automatisierten Daten-Analyse gewählt wird, werden Data Mining Methoden angewendet, um Modelle für die Original-Daten zu generieren. Wenn ein Modell erzeugt wurde, wird dieses evaluiert, um das Modell zu veredeln. Die Evaluierungs-Phase wird unterstützt durch den interaktiven Austauch mit den Daten und der Hilfe von Visualisierungs-Methodiken, die dem Analyst eine Interaktion mit den automatisierten Methoden durch das Verändern von Parametern oder durch die Wahl eines anderen Algorithmus' ermöglichen. Um aus diesen erstellten Modellen Schlüße zie-

hen zu können, können Modell-Visualisierungen verwendet werden. Dieser alternierende Austausch zwischen den automatisierenden und den visualisierenden Methoden ist typisch für den VA Prozess und ermöglicht eine ständige Optimierung und Verifikation der im vorangegangenen Schritt ermittelten Ergebnisse. So können in einem frühen Stadium der Analyse Fehler oder missverstandene Aspekte entdeckt und erneut analysiert werden, sodass Entscheidungen überdacht und neu getroffen werden können. Wenn zuerst die visuelle Daten-Exploration gewählt wurde, ist es notwendig, die durch die visuelle Exploration erzeugten Hypothesen durch automatisierte Analysen zu bestätigen und zu belegen. Hierbei ist die User-Interaktion erforderlich, um aufschlussreiche Informationen zu enthüllen. Diese gewonnen Erkenntnisse können dann verwendet werden, um automatisierte Analysen in der Modell-Bildung zu unterstützen.

3.2 Visual Analytics in der Pharmakovigilanz (Adverse Events)

Die Visualisierung von überlappenden Mengen ist ein bekanntes Problem in der Informationsvisualisierung. Hierbei geht es um verschiedene Mengen und deren Elemente, die entweder zu beiden Mengen, nur zu einer oder zu gar keiner der Mengen gehören. Als Grundlage hierfür galt das Paper von Lamy, et al.[16], der das Problem auf Basis von biomedizinischen Daten beschreibt. Dabei geht es um die Visualisierung z.B. von den Eigenschaften von den 20 Aminosäuren oder von Kontraindikationen oder Nebenwirkungen von verschiedenen Arzneimitteln. In dem Paper wurde die neue Visualisierungs-Technik der Rainbow Boxes vorgestellt. Diese wurden für die Präsentation der Eigenschaften von Aminosäuren, den Vergleich von Drug-EIgenschaften und für die Visualisierung von Gen-Annotationen verwendet. Das Ziel dieser Technik ist u.a. zu zeigen, welche der Mengen bestimmte Elemente enthalten und welche nicht. Zudem sollen Intersektionen und Disjunktionen näher beleuchtet werden können, um neue Erkenntnisse über Ähnlichkeiten zwischen den Elementen oder Mengen finden zu können. Dadurch dass Medikamente viele Eigenschaften, wie z.B. Indikationen, Kontraindikationen, Interaktionen oder Nebenwirkungen, aufweisen, ist hier das Darstellungs-Problem wesentlich komplexer als bei z.B. bei der Betrachtung von den 20 verschiedenen Aminosäuren. Es ist bereits schwierig ein Medikament mit seinen Eigenschaften klar und prägnant zu repräsentieren. So ist der visuelle Vergleich bei der Betrachtung von mehreren Medikamenten eine wahre Herausforderung. Der Vergleich von Drug-Properties kann als eine überlappende Mengen-

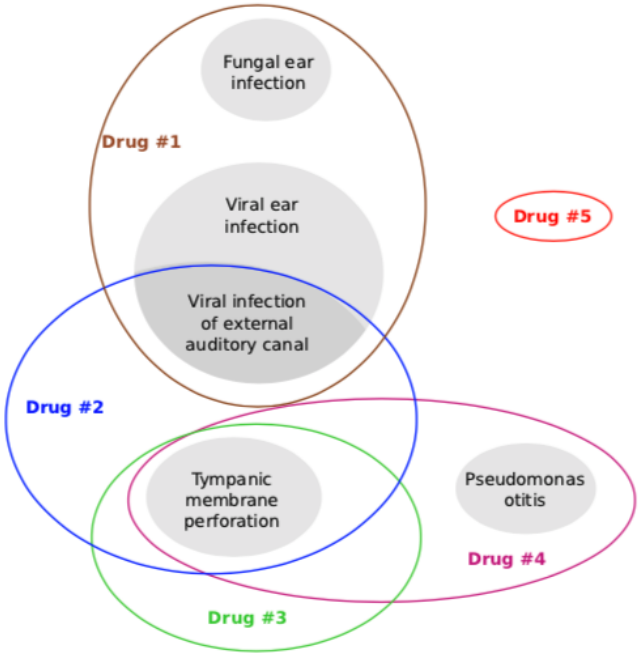


Abbildung 3.4: Venn Diagramm [16]

Visualisierung ausgedrückt werden, indem Drugs als Elemente und deren Eigenschaften als Mengen betrachtet werden. Das schwierige an der Visualisierung beim Vergleichen von Medikamenten ist die Präsentation von den vielen verschiedenen Eigenschaften bezüglich der Medikamentensicherheit, wie z.B. die Kontraindikationen, Interaktionen und Nebenwirkungen. Die Darstellung von veschiedenen Kontraindikationen oder Nebenwirkungen von verschiedenen Medikamenten stellt auch das oben genannte überlappende Mengen-Visualisierungs-Problem[16]. Denn ein Medikament kann ein bis viele Nebenwirkungen aufzeigen und Nebenweirkungen können wiederum bei verschiedenen Medikamenten auftreten. Ein Beispiel für eine Technik für die Visualisierung von überlappenden Mengen ist das bekannte Venn-Diagramm. Abbildung 3.4 zeigt das Venn-Diagramm, wobei die Drugs als Mengen und die Nebenwirkungen (Adverse Events) als Elemente abgebildet sind. Dies lässt sich auch umgekehrt darstellen (Nebenwirkungen als Mengen und Drugs als Elemente), da es sich hierbei um ein symmetrisches Problem handelt, tut es jedoch nicht, da die Eigenschaften meist eine größere Anzahl aufweisen als die Arzneimittel und Venn-Diagramme sind effizienter bei einer geringeren Anzahl an Mengen als an Elementen. Solange der darzustellende Datensatz relativ gering ist, ist diese Art der Repräsentation noch geeignet. Wächst die Anzahl jedoch, wird diese Art der Graphik schnell unübersichtlich und ist nicht mehr userfreundlich. Um diesem Aspekt entgegen zu wir-

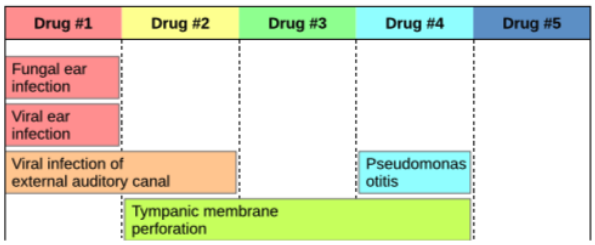


Abbildung 3.5: Rainbow Boxes [16]

ken entwickelten von Lamy et al. die neue Visualisierungs-Technik der *Rainbow Boxes*. Hierbei sind die Drugs als Spalten (Elemente) und die Eigenschaften als waagerechte rechtförmige Boxen (Mengen) dargestellt. Dabei können die Boxen mehrere Spalten füllen oder LLückenöcher aufweisen, je nach dem ob ein Medikament die zur Box zugehörige Nebenwirkung aufweist oder nicht (s. Abb. 3.5).

Es gibt nur wenige Lösungen für die Visualisierung von Drug-Eigenschaften und wenn es diese gibt, haben diese sehr einfache Visualisierungsformen, wie Tabellen.

4 Anforderungsanalyse

Die Anforderungsanalyse umfasst eine detaillierte Ermittlung und Beschreibung des gewünschten Verhaltens und der gewünschten Eigenschaften eines zu entwickelnden Systems. Dazu gehört ein Lastenheft und ein Pflichtenheft, sodass festgehalten wird, was die geforderten Anforderungen an das zu entwickelnde System sind. Es ist ein reger Austausch zwischen von verschiedenen Stakeholdern notwendig. Die Anforderungsanalyse ist ein sehr wichtiger Aspekt in einem Entwicklungszyklus einer Software. Unvollständige Anforderungen sind eine große Problemquelle in einem Projekt. Denn wenn die Anforderungen nicht klar sind oder nicht präzise beschrieben sind, wird die erzeugte Software auch nicht das leisten, was sich der Endkunde z.B. vorgestellt hat. Anforderungen beschreiben ein zu lösendes Problem. Wie das Problem gelöst wird, beschreibt die fachliche Lösung und stellt das Problem für die technische Lösung dar[8]. Es gibt verschiedene Anforderungen. Zum einen gibt es die Benutzeranforderungen, die beschreiben sollen, was für Dienste das System realisieren soll und die Randbedingungen, unter denen es betrieben wird. Diese Anforderungsspezifikation, die auf den Anforderungen des Kunden basieren, werden in Form eines Lastenheftes beschrieben. Diese ist die Grundlage für die Vertragsgestaltung und somit die wichtigste Vorgabe für die Angebotserstellung[8]. Mit diesen Anforderungen werden die Rahmenbedingungen festgelegt. Zum anderen müssen die technischen Anforderungen festgelegt werden, die zum Realisieren der Anforderungsspezifikation notwendig sind. Diese bilden die Systemanforderungen und werden in einem Pflichtenheft festgehalten. Sie legen Funktionen und Dienste mit den Beschränkungen fest. Im Rahmen dieser Arbeit wurden die Anforderungen mündlich mit der Dozentin besprochen und festgehalten, sodass es keine weiteren direkt eingebundenen Stakeholdern gab.

Technical documents: Installation, Data cleaning, Caveat, Citing

Due to the nature of the method of collecting pharmacovigilance data and the nature of the data itself, several precautions need to be taken for high-quality analyses of drugs and their putative adverse drug reactions. This is especially important if you chose to install OpenVigil yourself.

- Cave at documents: Methodological mistakes when crafting or interpreting queries
  - OpenVigil 1 & 2 caveat v2.0.2 (old version: OpenVigil 1 cave-at v1.0: http://openvigil.pharmacology.uni-kiel.de/caveat.html)
  - FDA caveat [mirrored from http://www.fda.gov/downloads/Drugs/GuidanceComplianceRegulatoryInformation/Surveillance/AdverseDrugEffects/UCM350392.pdf]
  - WHO UMC caveat [mirrored from http://www.who-umc.org/graphics/25300.pdf]
  - BfArM caveat [mirrored from BfArM cave-at which is now offline]
  - MedDRA: Points to consider (instructions for coders which are also useful for decoders) [mirrored from http://www.meddra.org/sites/default/files/guidance/file/9491-1800_termselptc_r4_9_mar2015.pdf]
- Software validation reports/bug lists
  - Known issues in OpenVigil 1 & 2 and in OpenVigilFDA
  - Validation report for OpenVigil FDA v1.0rc4
- Before using an installation of OpenVigil, be sure to know your data, e.g., check which files were successfully imported:
  - Imported files and record import failures in OpenVigil 1, e.g., at CAU Kiel: http://openvigil.pharmacology.uni-kiel.de/openvigil.php?cd=if
  - Overview of an installation of OpenVigil 1, e.g., at CAU Kiel: http://openvigil.pharmacology.uni-kiel.de/openvigil127.php?cd=vs
  - Imported files in OpenVigil 2, e.g., at CAU Kiel: http://www.is.informatik.uni-kiel.de:8503/OpenVigil/admin/public/imported_files.jsp
  - Overview of an installation of OpenVigil FDA, e.g., at CAU Kiel: http://openvigil.pharmacology.uni-kiel.de/openvigilfda.php?about=1
- Basic methodologies for validation and interpretation
  - Data validation, cleansing procedures and quality assessment primer
  - Disproportionality analysis primer
- How to install you own OpenVigil instance
  (For citation of an installation, give a summary/overview of imported files and software version, see above!)
  - Installing OpenVigil 1
  - Installing OpenVigil 2
- Data structure and relations
  - Relational schema for drugs/brands in OpenVigil 2 for SQL queries

Abbildung 4.1: How to install Open Vigil 2.0. [5]

4.1 IST-Zustand

Wie bereits in der Einleitung beschrieben, geht es in dieser Arbeit darum, das bestehende Projekt OpenVigil 2.1. durch Visual-Analytics-Komponenten zu erweitern. Das Projekt OpenVigil 2.1. wurde in Java geschrieben, verwendet das objektrelationale Datenbankmanagementsystem PostgreSQL für die Datenverwaltung und läuft auf einem Tomcat Server. Auf der Seite des Open Vigil Projektes (http://openvigil.sourceforge.net/) findet man unter *Technical docs and other resources* die technischen Dokumente, die Referenzen zur Installation und u.a. zur Daten-Säuberung abbildet (s. Abb. 4.1). Des weiteren sind ebenfalls Verweise auf andere allgemeine Daten-Analyse-Tools und Pharmakovigilanz-Daten-Quellen erwähnt. In Abbildung 4.1 ist das aktuellste Installations-Dokument zu sehen, das man zu Open Vigil 2 auf der Seite `http://openvigil.sourceforge.net/doc/HowtoInstallOpenVigil.pdf` findet, das *How to Install OpenVigil 2.0*. Dieses ist aus dem Jahr 2015. Für die aktuellste Version von OV 2 gibt es kein öffentlich zugängliches Dokument, welches die Installation und die Konfiguration des Systems beschreibt. Dieses Dokument beschreibt, dass für die Installation von OV Apache Tomcat und PostgreSQL notwendig sind. Zudem wird auch die Konfiguration von PostgresSQL, Open Vigil WAR und dem WebApp Access in Tomcat beschrieben und dem anschließenden Deployen der Web-Applikation in Tomcat. Alle diese Einstellungen beziehen sich auf den Apache Tomcat 7 und PostgreSQL 9.2. Zudem greift das Dokument auch den Import der Daten in die Datenbank auf. Bevor Pharmakovigilanz-Daten importiert werden kön-

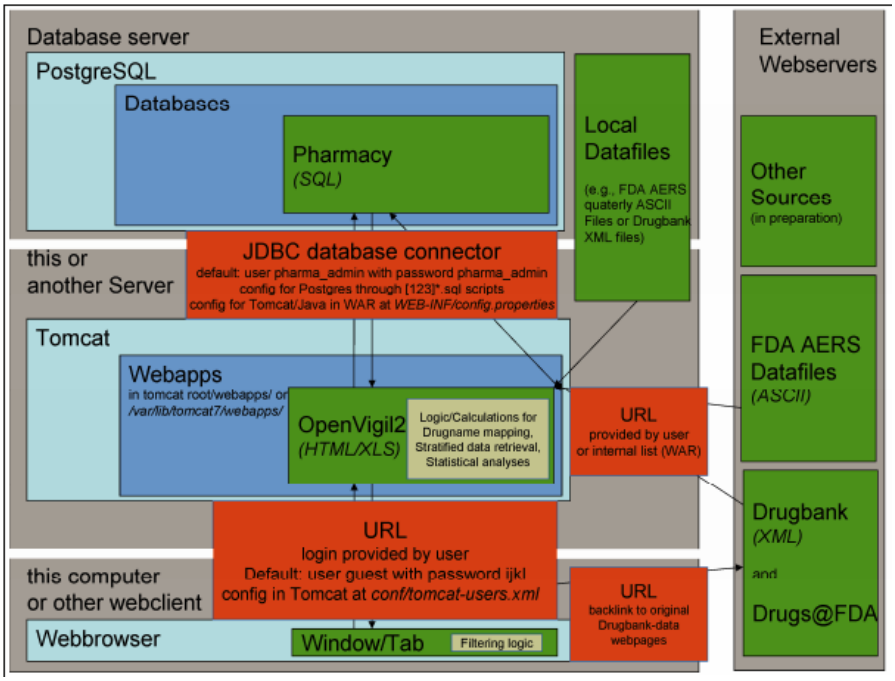


Abbildung 4.2: Open Vigil Laufzeit Umgebung [5]

nen, müssen die Drugname-Daten (Drugbank bzw. Drugs@FDA) importiert werden, da die Freitext-FDA Medikamentennamen eindeutigen Medikamentennamen (US Adopted Names-USAN, International Nonproprietary Name-INN) zugeordnet werden. Darüber hinaus zeigt die Installations-Datei die in Abbildung 4.2 abgebildete Übersicht über die OpenVigil Laufzeit-Umgebung. Man sieht, wie OV im Tomcat eingebettet ist und dass es über eine JDBC Verbindung mit dem Datenbank-Server kommuniziert, um SQL-Queries zu schicken und die Antworten entgegen zu nehmen.

Um Daten zu importieren, muss man als Administrator angemeldet sein und dann unter dem Menüpunkt *Administration* eine der folgenden drei Datenquellen auswählen.

- AERS: FDA Adverse Event Reporting System ist eine Datenbank, die Informationen über Nebenwirkungen und Medikationsfehler hält (`https://fis.fda.gov/extensions/FPD-QDE-FAERS/FPD-QDE-FAERS.html`).
- Drugbank: Von der University of Alberta erzeugte Datenbank, die über 13344 biologische und chemische Informationen über pharmazeutische Wirkstoffe enthält (https://www.drugbank.ca/).

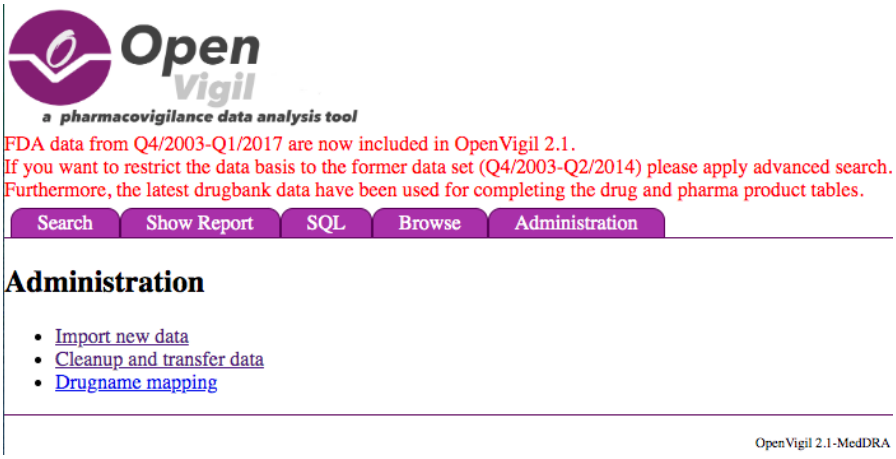


Abbildung 4.3: Open Vigil 2.1. Administration [5]

- Drugs@FDA: Enthält die meisten seit 1939 zugelassene Medikamente (https://www.fda.gov/drugs/drug-approvals-and-databases/drugsfda-data-files).

Der Import bei OpenVigil 2.1. wird in Abbildung 4.3 abgebildet. Wie oben beschrieben, ist anfangs der Import der Daten der Drugbank bzw. Drugs@FDA notwendig. Also werden zuerst diese Datensätze importiert und dann die quartalsweise von der FDA zur Verfügung gestellten AERS-Daten. Was nicht in der oben erwähnten Datei berschrieben ist, ist dass man nach einem Import über den Menüpunkt *Administration* die nächsten Schritte manuell triggern muss, um die Daten zu säubern und Duplikate zu entfernen. Erst dann werden die gesäuberten Daten in die Datenbank transferiert.

Nachdem die Datenbank befüllt worden ist, kann über die Startseite, die aus den drei verschiedenen Reitern *Search*, *Show Report* und *Browse* besteht, über den Such-Reiter eine Anfrage verschickt werden. Für den Administrator stehen noch zwei weitere Reiter *Show Report* und *Administration* zu Verfügung, mit denen dieser nach bestimmten ISRs (Individual Safety Report) suchen kann oder über den Administrations-Reiter wie oben beschrieben Daten importieren kann. Über die Search-Seite nach verschiedenen Drugs und/oder verschiedenen Events gesucht werden. Dabei können ein oder mehrere Medikamenten- bzw. Wirkstoffnamen eingegeben werden. Bei einer Anfrage mit mehr als einem Medikament werden die einzelnen Medikamente logisch miteinander verknüpft. Bei den Nebenwirkungen (Adverse Events) gilt dieselbe Vorgehensweise. Die dem User zur Verfügung stehenden logischen Operatoren sind: AND, OR, XOR und NAND. Je nach Operator wird die Konjuntion (AND), Disjunktion (OR), Differenzmenge (XOR) oder die NAND-Gatter (NAND) auf die Daten angewendet. Bei der Konjunktion werden nur

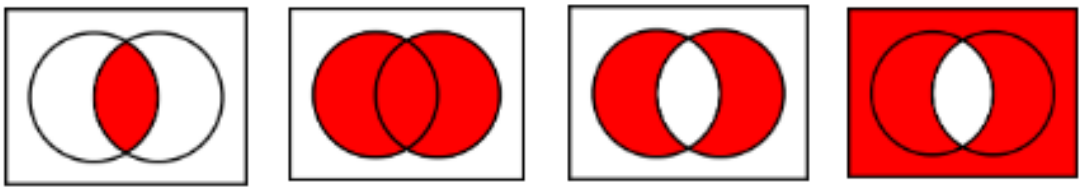


Abbildung 4.4: Logische Gatter: AND, OR, XOR, NAND

die Daten im Ergebnis angezeigt, die auch beide Elemente aufweisen (s. Abb. 4.4). An der Wahrheitstabelle ist gut zu erkennen, dass die AND-Verknüpfung der beiden Mengen A und B (also A AND B) nur dann wahr bzw. 1 ergeben, wenn beide wahr sind (s. 1. Tabelle). Bei dem Ergebnis handelt es sich um die Schnittmenge der beiden Mengen A und B. Bei der OR-Verknüpfung werden alle Elemente betrachtet, die sich in den Mengen A und B befinden. D.h. wenn eines der Elemente true ergibt, ergibt auch das Resultat der Verknüpfung true. Beide Mengen werden vereinigt (s. 2. Tabelle). Die sich aus dem XOR-Operator ergebende Menge ist eine symmetrische Differen von den Mengen A und B. In dieser Menge liegen alle Elemente, die jeweils einer, aber nciht beiden Mengen angehören (s. 3. Tabelle). Bei dem NAND-Operator wird die AND-Menge der beiden Menge ausgeschlossen, d.h. die Schnittmenge wird ignoriert (s. 4. Tabelle).

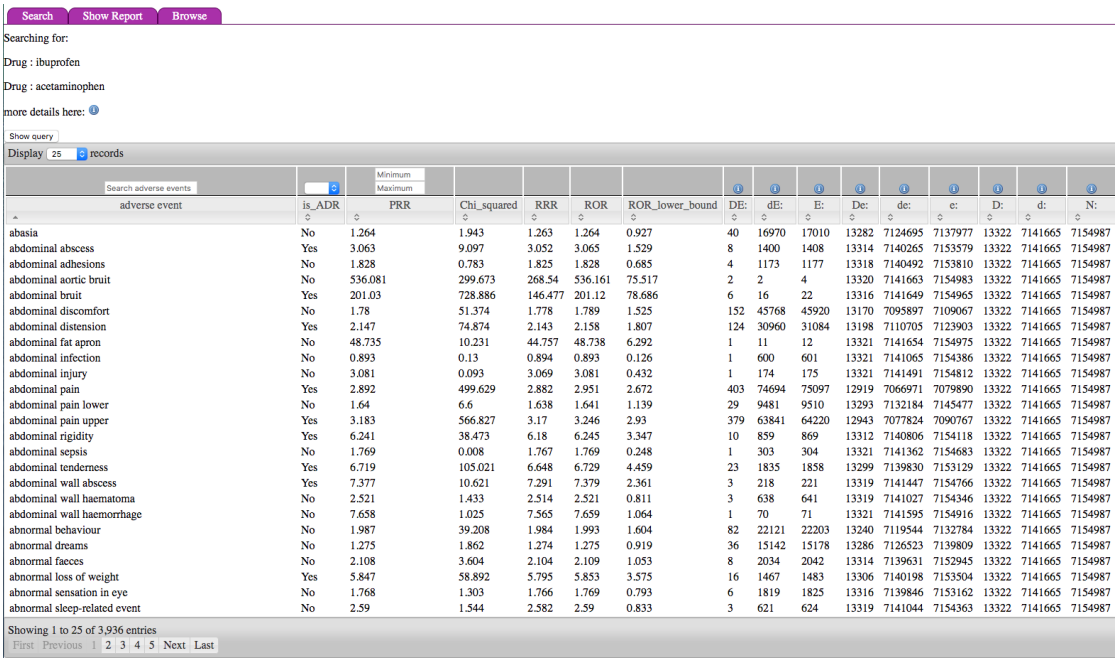
A	B	(A AND B)
1	1	1
1	0	0
0	1	0
0	0	0

A	B	(A OR B)
1	1	1
1	0	1
0	1	1
0	0	0

A	B	(A XOR B)
1	1	0
1	0	1
0	1	1
0	0	0

A	B	(A NAND B)
1	1	0
1	0	1
0	1	1
0	0	1

Bei der Eingabe der Input-Felder (sowohl Drug als auch Adverse Event) wird erst ab einer Wort-Länge von drei Buchstaben die Autocomplete-Funktion getriggert (bei LLT-Type schon ab einem eingegeben Buchstaben). Nach Eingabe eines Inputfeldes wird die Abfrage an die Datenbank geschickt und auf der result.jsp-Seite werden die Ergebnisse



Search | Show Report | Browse

Searching for:

Drug : ibuprofen

Drug : acetaminophen

more details here:

Show query

Display 25 records

adverse event	is_ADR	PRR	Chi_squared	RRR	ROR	ROR_lower_bound	DE:	dE:	E:	De:	de:	e:	D:	d:	N:
abasia	No	1.264	1.943	1.263	1.264	0.927	40	16970	17010	13282	7124695	7137977	13322	7141665	7154987
abdominal abscess	Yes	3.063	9.097	3.052	3.065	1.529	8	1400	1408	13314	7140265	7153579	13322	7141665	7154987
abdominal adhesions	No	1.828	0.783	1.825	1.828	0.685	4	1173	1177	13318	7140492	7153810	13322	7141665	7154987
abdominal aortic bruit	No	536.081	299.673	268.54	536.161	75.517	2	2	4	13320	7141663	7154983	13322	7141665	7154987
abdominal bruit	Yes	201.03	728.886	146.477	201.12	78.686	6	16	22	13316	7141649	7154965	13322	7141665	7154987
abdominal discomfort	No	1.78	51.374	1.778	1.789	1.525	152	45768	45920	13170	7095897	7109067	13322	7141665	7154987
abdominal distension	Yes	2.147	74.874	2.143	2.158	1.807	124	30960	31084	13198	7110705	7123903	13322	7141665	7154987
abdominal fat apron	No	48.735	10.231	44.757	48.738	6.292	1	11	12	13321	7141654	7154975	13322	7141665	7154987
abdominal infection	No	0.893	0.13	0.894	0.893	0.126	1	600	601	13321	7141065	7154386	13322	7141665	7154987
abdominal injury	No	3.081	0.093	3.069	3.081	0.432	1	174	175	13321	7141491	7154812	13322	7141665	7154987
abdominal pain	Yes	2.892	499.629	2.882	2.951	2.672	403	74694	75097	12919	7066971	7079890	13322	7141665	7154987
abdominal pain lower	No	1.64	6.6	1.638	1.641	1.139	29	9481	9510	13293	7132184	7145477	13322	7141665	7154987
abdominal pain upper	Yes	3.183	566.827	3.17	3.246	2.93	379	63841	64220	12943	7077824	7090767	13322	7141665	7154987
abdominal rigidity	Yes	6.241	38.473	6.18	6.245	3.347	10	859	869	13312	7140806	7154118	13322	7141665	7154987
abdominal sepsis	No	1.769	0.008	1.767	1.769	0.248	1	303	304	13321	7141362	7154683	13322	7141665	7154987
abdominal tenderness	Yes	6.719	105.021	6.648	6.729	4.459	23	1835	1858	13299	7139830	7153129	13322	7141665	7154987
abdominal wall abscess	Yes	7.377	10.621	7.291	7.379	2.361	3	218	221	13319	7141447	7154766	13322	7141665	7154987
abdominal wall haematoma	No	2.521	1.433	2.514	2.521	0.811	3	638	641	13319	7141027	7154346	13322	7141665	7154987
abdominal wall haemorrhage	No	7.658	1.025	7.565	7.659	1.064	1	70	71	13321	7141595	7154916	13322	7141665	7154987
abnormal behaviour	No	1.987	39.208	1.984	1.993	1.604	82	22121	22203	13240	7119544	7132784	13322	7141665	7154987
abnormal dreams	No	1.275	1.862	1.274	1.275	0.919	36	15142	15178	13286	7126523	7139809	13322	7141665	7154987
abnormal faeces	No	2.108	3.604	2.104	2.109	1.053	8	2034	2042	13314	7139631	7152945	13322	7141665	7154987
abnormal loss of weight	Yes	5.847	58.892	5.795	5.853	3.575	16	1467	1483	13306	7140198	7153504	13322	7141665	7154987
abnormal sensation in eye	No	1.768	1.303	1.766	1.769	0.793	6	1819	1825	13316	7139846	7153162	13322	7141665	7154987
abnormal sleep-related event	No	2.59	1.544	2.582	2.59	0.833	3	621	624	13319	7141044	7154363	13322	7141665	7154987

Showing 1 to 25 of 3,936 entries

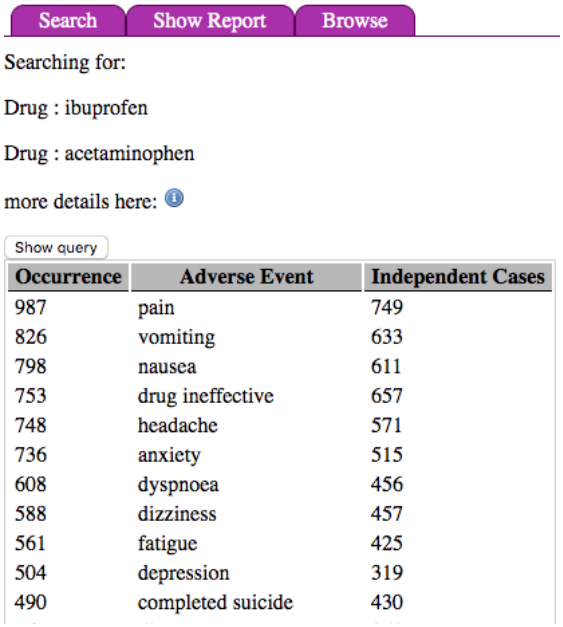
First Previous 1 2 3 4 5 Next Last

Abbildung 4.5: Ausschnitt der Suche mit den übergebenen Werten Ibuprofen und Acetaminophen mit der Evaluations-Methode Frequentist Methods.

dargestellt. Per Default sind die Methode *Frequentist Methods* und HTML als Ausgabe-Format gesetzt (s. Abb. 4.5). Die Methode *Frequentist Methods* stellt statistische Werte unterschiedlicher statistischer Analysemethodiken dar.

Das Ergebnis der Suche mit den gleichen übergebenen Medikamentennamen *Ibuprofen* und *Acetaminophen* sieht man in der Abbildung 4.6. Hier ist die verwendete Methode die *Frequency*. Ermittelt wird die Anzahl der Vorkommnisse der Nebenwirkungen und wie oft unterschiedliche Adverse Events in Kombination mit den übergebenen Medikamenten vorkommen.

Die dritte Evaluations-Methode, die zur Auswahl steht, ist die *Raw data*, die alle Präparate abbildet, zu denen die übergebenen Arzneimittel gehören und die dazugehörigen Nebenwirkungen sowie die ISR-Nummern. Wie man sieht, werden alle Ergebnisse der Auswertungen in Form von Tabellen dargestellt. Mit dem Button *Show query* wird zusätzlich die Möglichkeit bereit gestellt, sich das Query anzeigen zu lassen. Zudem lassen sich bei der erweiterten Suche verschiedene zusätzliche Filter, wie z.B. Age, Gender oder Reporter Country, setzen und/oder andere Daten-Repräsentations und Statistik-Analysen wählen. (s. Abb. 4.8). Je nachdem welche Evaluationsmethode man gewählt hat, kann



Occurrence	Adverse Event	Independent Cases
987	pain	749
826	vomiting	633
798	nausea	611
753	drug ineffective	657
748	headache	571
736	anxiety	515
608	dyspnoea	456
588	dizziness	457
561	fatigue	425
504	depression	319
490	completed suicide	430

Abbildung 4.6: Ausschnitt der Suche mit den übergebenen Werten Ibuprofen und Acetaminophen mit der Evaluations-Methode Frequency.

man ebenfalls zusätzlich unten bei den Output-Items verschiedene Output-Filter setzen oder entfernen.

4.2 SOLL-Zustand

Die Anforderungen dieser Arbeit waren es, die Webapplikation OpenVigil 2.1. durch Visual Analytics Komponenten zu erweitern, sodass die einzelnen Funktionalitäten der Seite für den Endanwender ansprechender sind und vor allem verständlicher. Dadurch sollte mehr an Information vermittelt werden können. Denn für einen Nicht-Mathematiker könnten z.B. die Operatoren auf der Suchseite nicht auf Anhieb die Informationen transportieren, die gedacht waren. Zudem sollten Wege gefunden werden, wie Daten und die Ergebnisse der Statistik-Analysen repräsentiert werden konnten. Die Integration von Visual Analytics Komponenten gliederte sich in zwei Teile. Zum einen sollte die Erweiterung auf seitens der Suche der Webapplikation stattfinden und zum anderen auf der Seite der Ergebnisse.

ISR	Date received	Event	Case_id	Drug_seq	Original name	Role code	Route	Dosis	Calculated daily dosis
131499991	2017-01-25	• condition aggravated • dissociation • hypoaesthesia • mental disorder • product use issue • sensory loss	13149999	1314999911	PREGABALIN	Primary Suspect Drug	Oral	75MG AM, 150MG PM	
				1314999912	BACLOFEN	Secondary Suspect Drug	Oral	20 MG, UNK	
				1314999913	ACETAMINOPHEN	Concomitant		UNK	
				1314999914	AMITRIPTYLINE	Concomitant		UNK	
				1314999915	DEXTROAMPHETAMINE SULFATE	Concomitant			
				1314999916	IBUPROFEN	Concomitant		UNK	
4915269	2006-02-16	• dyspnoea • nightmare	5990180	1006947898	ROBAXIN	Primary Suspect Drug	ORAL	1500MG PO X1 DOSE	
				1006970868	IBUPROFEN	Concomitant			
				1006970869	AMBIEN	Concomitant			
				1006970870	VICODIN	Concomitant			
				1006970871	ALBUTEROL	Concomitant			
146835663	2018-06-06	• drug ineffective	14683566	1468356631	IBUPROFEN	Primary Suspect Drug		200 MG, 3X/DAY	600.0
				1468356632	ACETAMINOPHEN	Concomitant		UNK	
131663821	2017-01-31	• intentional product misuse	13166382	1316638211	ACETAMINOPHEN	Primary Suspect Drug	Oral		
				1316638212	IBUPROFEN	Secondary Suspect Drug	Oral		
7929964	2011-11-21	• acute hepatic failure • dermatitis exfoliative • drug interaction • pneumonia • respiratory failure	8245348	1018098102	CLARITHROMYCIN	Primary Suspect Drug			
				1018098103	CLARITHROMYCIN	Interacting			
				1018098104	CLARITHROMYCIN	Interacting			
				1018098105	IBUPROFEN	Interacting			
				1018098106	IBUPROFEN	Interacting			
				1018098107	IBUPROFEN	Interacting			
				1018098108	ACETAMINOPHEN	Interacting			
				1018098109	ACETAMINOPHEN	Interacting			
				1018098110	ACETAMINOPHEN	Interacting			
111052421	2015-05-11	• liver injury	11105242	1110524211	IBUPROFEN	Primary Suspect Drug	Unknown		
				1110524212	IBUPROFEN	Secondary Suspect Drug	Unknown		
				1110524213	ACETAMINOPHEN	Secondary Suspect Drug	Unknown		
				1110524214	ACETAMINOPHEN	Secondary Suspect Drug	Unknown		
5406799	2007-08-07	• arthralgia • dysstasia • non-cardiac chest pain • rash generalised	6294008						
				1008779260	MIDOL	Primary Suspect Drug	ORAL	AS USED: 500/15/60 MG UNIT DOSE: 500 MG	
114394831	2015-09-01	• headache	11439483	1143948311	PEGINTERFERON ALFA-2A	Primary Suspect Drug	Subcutaneous		
				1143948312	RIBAVIRIN	Secondary Suspect Drug	Oral	DIVIDED DOSES	
				1143948313	IBUPROFEN	Concomitant			
				1143948314	ACETAMINOPHEN	Concomitant			
129173393	2019-01-29	• iron deficiency • pelvic pain • prolapse	12917339	1291733931	SODIUM OXYBATE	Primary Suspect Drug	Oral	2.25 G, BID	
				1291733932	SODIUM OXYBATE	Secondary Suspect Drug	Oral	DOSE ADJUSTMENTS	
				1291733933	SODIUM OXYBATE	Secondary Suspect Drug	Oral	4.5 G, BID	
				1291733934	PROPRANOLOL HYDROCHLORIDE	Concomitant		UNK	
				1291733935	HYDROXYZINE PAMOATE	Concomitant		UNK	
				1291733936	IRON	Concomitant		UNK	
				1291733937	BUSPIRONE HYDROCHLORIDE	Concomitant		UNK	
				1291733938	AMITRIPTYLINE	Concomitant		UNK	

50 per page Page 1 of 2017

Abbildung 4.7: Ausschnitt der Suche mit den übergebenen Arzneimittel Ibuprofen und Acetaminophen mit der Evaluations-Methode Raw Data.

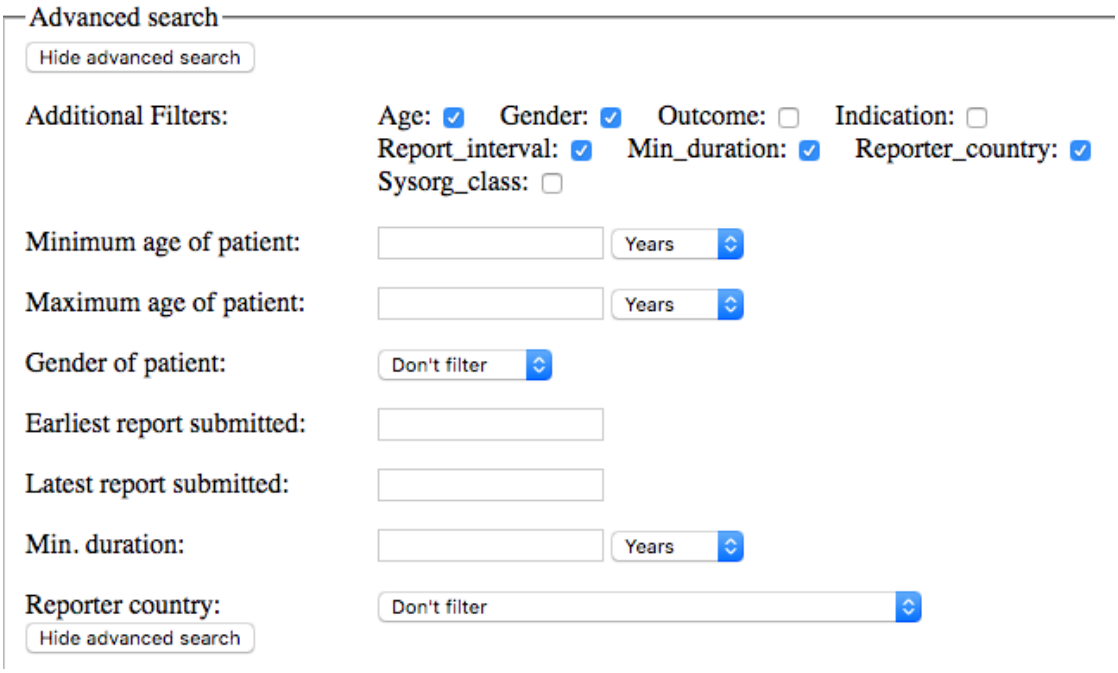


Abbildung 4.8: Advanced Search

4.2.1 Funktionale Anforderungen

Funktionale Anforderungen sind laut Balzert[8] Fähigkeiten eines Systems, die ein Anwender erwartet, um damit fachliche Probleme lösen zu können. Sie können z.B. durch Anwendungsfälle (Use Cases) beschrieben werden. Anwendungsfälle wiederum stellen Teilaspekte des gesamten Systemverhaltens dar. Nach Balzert legt eine funktionale Anforderung einen vom Softwaresystem oder von einem Teilsystem bereitzustellenden Dienst fest. Das heißt, sie entscheidet, *was* ein System tun soll. Auf der Startseite sollen folgende funktione Anforderungen realisiert werden:

- Hinzufügen von neuen visuellen Bereichen (sowohl für die Drug- als auch für die Adverse Event-Eingabe)
- Beibehalten der bisherigen Funktionalität des Systems. Neue hinzufügende Komponenten sollen in keiner Weise die bisherigen Funktionen des Systems beeinträchtigen (z.B. Hinzufügen/Löschen von Search-Kindknoten, Ändern der Operatoren, Ändern der Drug-/Adverse-Events-Typs).
- Daten sollen unverändert bleiben.

- Mengendarstellung der ausgewählten Arzneimittel.
- Färbung der unterschiedlichen Verknüpfungen durch die Auswahl der Operatoren.
- Erzeugung von neuen Kind-Knoten führt zu dem Hinzufügen eines neuen Elements in den Visualisierungs-Komponenten
- Löschen von Kind-Knoten führt auch zu dem Löschen des Elements in den Visualisierungs-Komponenten
- Updating der Visualisierungs-Komponenten bei Veränderungen der einzelnen Elemente (Inputfield für die Drug/Adverse Event-Eingabe, Select-Box der Operatoren, etc.).
- Weiterreichen der Eigenschaften der einzelnen Felder an die Visualisierungs-Komponenten

Die Result-Seite soll mit den folgenden neuen funktionalen Anforderungen ausgestattet werden:

- Ergebnis-Daten sollen weiterhin erhalten und unverändert bleiben.
- Ersetzen und/oder Erweitern der bisherigen Tabellen um neue VA-Graphiken.
- Ermöglichen einer Interaktion mit dem User.

Es galt die oben beschriebenen getroffenen Auswahlen der einzelnen Drugs (bzw. Adverse Events) mit den ausgewählten Operatoren abzubilden, sodass man anhand der Färbungen die Mengenauswahl sieht.

4.2.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen sind technische Anforderungen (*Quality of Service*), die die funktionalen Anforderungen betreffen und ermöglichen.

Im Gegensatz zu den funktionalen Anforderungen legt eine nicht-funktionale Anforderung nicht fest, *was* ein System tun soll, sondern *wie gut* die zuvor bestimmten funktionalen Anforderungen erledigt werden bzw. erledigt werden sollen.

- Erhaltung der bisherigen Software-Architektur (bzw. nur leichte Veränderung)
- Wahrung der bisherigen Konfigurationen

- Erhaltung der verwendeten Technologien und Erweiterung um neue Technologien wie React
- Veränderung und Optimierung der Benutzbarkeit
- Verbesserung der allgemeinen Qualität der Applikation durch unterstützende interaktive User-Taks

Realisiert werden sollen diese nicht-funktionalen Anforderungen durch die Realisierung der funktionalen Anforderungen.

5 Umsetzung

Um eine einwandfreie Integration der neuen Komponenten in das Zielsystem gewährleisten zu können, musste erst einmal die Entwicklungsumgebung aufgebaut werden. Es lagen nur wenige Informationen über den Source-Code des Open Vigil Projektes vor und wenn etwas vorlag, ist es veraltet gewesen. Dadurch gestaltete sich die lokale Installation und Konfiguration des Projektes schwieriger als erwartet. Aus diesem Grund orientierte ich mich an der Version, die unter dem folgenden Link zu finden ist: http://openvigil.sourceforge.net/doc/HowtoInstallOpenVigil.pdf. Diese Version des Dokuments ist von 2015 und beschreibt die Installation und Einrichtung von OpenVigil 2.0. Im Folgenden gehe ich auf die Installationsschritte auf einem Windows 10 Betriebssystem ein: Da ich bereits pgAdmin 4, einer Open-Source-Software, die für die Entwicklung und Administration von PostgreSQL-Datenbanken verwendet wird, vorinstalliert hatte, konnte ich direkt mit der Einrichtung der notwendigen *pharmacy*-Datenbank beginnen: Dafür mussten zuerst die User *pharm_ data* und *pharm_ admin* angelegt werden, indem folgende SQL Statements im pgAdmin 4 ausgeführt werden:

```
-- pharm_data
CREATE ROLE pharm_data LOGIN ENCRYPTED PASSWORD '<pwEncrypted>'
NOINHERIT
VALID UNTIL 'infinity';
-- pharm_admin
CREATE ROLE pharm_admin LOGIN ENCRYPTED PASSWORD '<pwEncrypted>'
NOINHERIT
VALID UNTIL 'infinity';
```

Anschließend wurden die Datenbank-Schemata erstellt, indem das im Projekt zu findende SQL-Skript *2-create_ database.sql* ausgeführt wurde, welches sich im OpenVigil-2.1-Projekt-Verzeichnis *./OpenVigil 2.1/install* befindet.

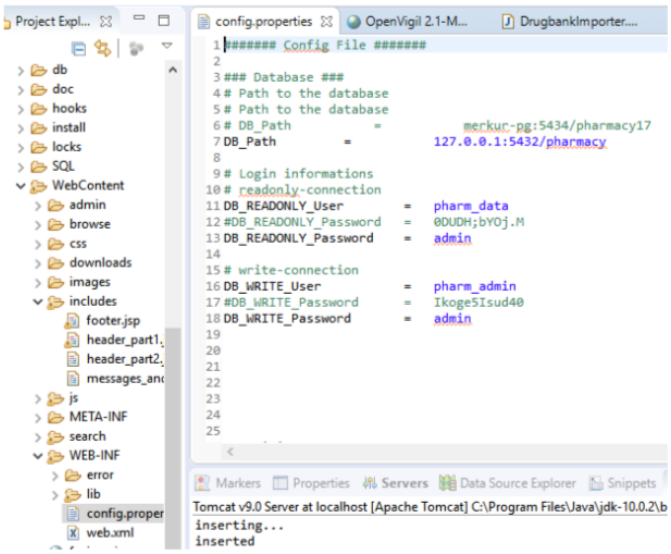


Abbildung 5.1: Konfiguration von OpenVigil für den Zugang zur PostgreSQL

Das zweite Skript *3-structure_ and_ data.sql*, was laut der OV2.0-Installations-Datei (http://openvigil.sourceforge.net/doc/HowtoInstallOpenVigil.pdf), anschließend ausgeführt werden soll, ist nicht im Projekt zu finden. Zusätzlich muss der Apache Tomcat Server installiert werden. Auch diesen hatte ich bereits in der Version 9.0.17 vorliegen. Nun konnte OpenVigil in die IDE (hier Eclipse) importiert werden und so eingrichtet werden, dass ein Zugang zu der PostgreSQL möglich ist. Dafür muss die *./WebContent/WebINF/config.properties* angepasst werden (s. Abb. 5.1).

Der nächste Schritt ist das Einrichten des Tomcat Servers. Dafür sind folgende Schritte notwendig: Erzeugen der User für den Zugriff auf die Web-Applikation durch das Einfügen der Roles *openvigil, openvigil_ sql, openvigil_ admin* und der User *admin, sq, guest* in der Tomcat-Server-Datei *./apache-tomcat-9.0.14/conf/tomcat-users.xml* Nun kann die Webapplikation deployed und gestartet werden. Zu diesem Zeitpunkt hat die Datenbank aber noch keinerlei Daten, da diese noch nicht gefüllt wurde. So müssen nun der Datensatz der DrugBank und der der Drugs@FDA importiert werden. Dies geht wie in Kapitel vier beschrieben über den Administrations-Bereich. In Abbildung 5.3 ist ein Ausschnitt der Auswahlmöglichkeiten des Imports zu sehen, die zur Verfügung stehen. Bei dem Import der Drugbank-Datei trat der folgende Error auf, der in Abbildung 5.4 zu sehen ist. Das Problem war, dass der Name der XML-Datei hart codiert war, sich aber im Laufe der Zeit geändert hat. Die Lösung für diesen Fehler war es, diesen Namen in *drugbank.xml* zu ändern. Hier sollte man den Code ändern, indem der Name der Datei geholt wird,

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<role rolename="manager"/>
<role rolename="manager-gui"/>
<user username="meinlokaleradmin" password="meinlokaleradmin1234" roles="manager,manager-gui"/>

<role rolename="openvigil"/>
<role rolename="openvigil_sql"/>
<role rolename="openvigil_admin"/>
<user username="admin" password="abcd" roles="openvigil_admin"/>
<user username="sql" password="efgh" roles="openvigil_sql"/>
<user username="guest" password="ihkl" roles="openvigil"/>

</tomcat-users>
```

Abbildung 5.2: Konfiguration der Datei *tomcat-users.xml*

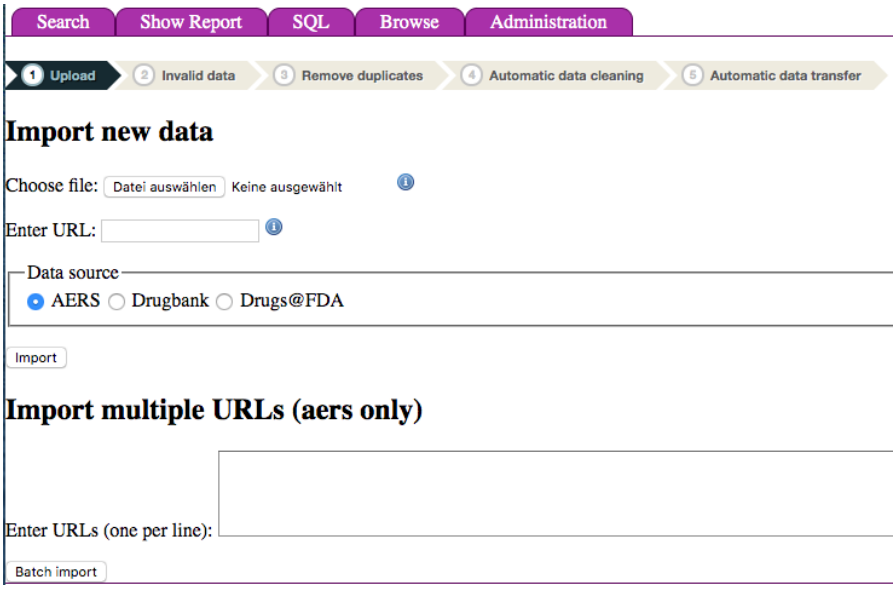


Abbildung 5.3: Import mit den drei verschiedenen Auswahlmöglichkeiten für die Datenquelle.

Abbildung 5.4: Import-Error beim Versuch des Drugbank-Imports.

statt ihn als String im Code zu hinterlegen.

Nach dem erfolgreichen Import der beiden Dateien *drugbank.xml* und der *drugsatfda.zip* können nun die FDA-AERS-Daten importiert werden. Die zu importierenden Dateien müssen im ASCII-Format vorliegen und die folgenden sieben Dateien enthalten: demo, drug, indiz, out, reac, rpsr, ther. Dass diese nur und unbedingt im ASCII-Format vorliegen müssen (FDA bietet sie sowohl als ASCII- als auch als XML-Files an), wird auch nicht explizit erwähnt, sodass diese auch zu Anwendungsfehlern und missverstandenen Ausführungen führen kann. Diese Daten werden als Zip-File importiert. Unter *Browse -> Show imported files* kann man sich die in die Datenbank erfolgreich importierten Dateien ansehen. Der Transfer dieser Imports lief leider nciht erfolgreich durch, sodass die wichtigen Tabellen wie public.event und public.rep_event leer blieben. In dem letzten Verarbeitungsschritt dem automatischen AERS-Daten-Transfer, bei dem versucht wurde, die gereinigten Daten in die Datenbank zu transferieren, wurde ein Consistency-Error ausgegeben (in der Java-Klasse AersAutomaticDataTransfer: Ërror on transfer. Consistency not guaranteed."). Zuerst wurde vermutet, dass es an den zu importierenden Daten liegen könnte, was nicht der Fall ist, da in einem zuvorgegangen angelegten OV-Projekt

```
/**
 * Parses the current drugbank version to an arraylist of drug objects and returns it
 * @return ArrayList of all Drugs listed in the Drugbank
 */
    private ArrayList<DrugbankDrug> parse(File inputFile) {

        try {
            /**System.setProperty("jdk.util.zip.ensureTrailingSlash","false");
            */
            ZipFile zipfile;

                zipfile = new ZipFile(inputFile);

            //Get drugbank.xml
//          ZipEntry entry = zipfile.getEntry("full database.xml");
            ZipEntry entry = zipfile.getEntry("drugbank.xml");
            BufferedInputStream bis = new BufferedInputStream(
            zipfile.getInputStream(entry));

            XMLInputFactory factory = XMLInputFactory.newInstance();
            XMLStreamReader parser = factory.createXMLStreamReader(bis);

//          DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss Z"); old date format
            DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
            DrugbankDrug drug = new DrugbankDrug();

            //First apperance of "name" is the name of the current Drug
            Boolean namedone = false;
            //Depth == 1 -> Current Drug , Depth > 1 -> Drug in a Drug, which should not be read
            int drugDepth = 0;

            //Parse every element of the xml-file and save relevant elements to a Drug object
            for (int event = parser.next(); event != XMLStreamConstants.END_DOCUMENT; event = parser
                    .next()) {
```

Abbildung 5.5: Ausschnitt der Parser-Methode in der DrugbankImporter-Klasse.

dieselben Daten erfolgreich importiert werden konnten. Leider konnte dem Fehler aus zeitlichen Gründen nicht weiter im Detail auf den Grund gegangen werden. Dies wäre ein weiterer zentraler Punkt, der in der zukünftigen Arbeit an dem OV-Projekt angegangen werden wird. Mit einem aus einem zuvorgegangenen Projekt gesicherten Dump wurde die Datenbank neu aufgesetzt, sodass Daten zu den Events, etc. vorlagen.

5.1 Entwurf

Das Vorhaben war, die vorhandene Webapplikation so wie sie ist, zu analysieren und an den passenden und richtigen Stellen zu erweitern, ohne dabei den bereits vorliegenden Sourcecode großartig zu verändern. Es sollte nur an der View des Projektes gearbeitet werden, da das Ziel der Aufbau der Visual Analytics Komponenten war. Das heißt, dass auf der ersten Seite innerhalb des div-Elements sowohl bei dem Drug- als auch bei dem Adverse Event-Bereich die neuen VA-Komponenten rechts abgebildet werden sollen.

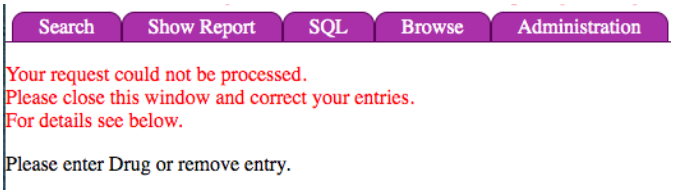


Abbildung 5.6: Ergebnisse einer Suche mit mindestens einem leeren Suchfeld.

5.1.1 Visual Analytics Komponenten

Visualisierung der Suchkriterien

Bei der Visualisierung der Search-Komponenten wurde schnell klar, dass durch die Möglichkeit der vielen verschiedenen Search-Konstellationen die visuelle Realisierung recht komplex werden würde. Denn bei den aktuellen Suchfeldern kann der User beliebig leere Kind-Knoten erstellen und auch mit so einer Suchkonstellation die Anfrage senden. Auf der Result-Seite wiederum ergibt so eine Suche keinerlei Suchergebnisse (s. Abb. 5.6). Im Folgenden verwende ich Parent, Parent-Knoten als Synonyme für das erste Suchfeld und Kind, Kind-Knoten als Synonyme für alle darauffolgenden Suchfelder. Zudem kann auch das erste Suchfeld, der Parentknoten, auch leer abgeschickt werden und das auch in der Konstellation, dass zwar Kind-Knoten vorhanden und ausgefüllt sind, aber der Parent leer ist. Auch dann endet die Suche in dem gleichen Fehler bzw. Endresultat. Dies sollte aus Sicht des Users vermieden werden, sodass z.B. das Hinzufügen von Kindern gar nicht erst möglich ist, wenn das Suchfeld des Parents nicht gefüllt ist. Man könnte z.B. den Plus-Button deaktivieren, solange das Parent-Suchfeld leer ist. Bei der Visualisierung der einzelnen Suchkriterien, die aus einem Inputfield, einem Dropdown der Operatoren und einem Dropdown für die Drug-Art bestehen, war es wichtig die Verbindungsart der einzelnen zu suchenden Medikamente darzustellen. Das heißt je nach selektiertem Operator veränderte sich die Ausgabe-Menge der Suche. Dieses überlappende Set an Daten sollte visuell dargestellt werden, sodass die Bedeutung und die Aussagekraft der einzelnen Verknüpfungsmöglichkeiten dem User intuitiv vermittelt werden kann. Denn nicht jedem User ist auf Anhieb die Bedeutung von Logikgattern klar, wodurch der Zweck und Sinn dieses Suchtasks der Webapplikation nicht voll ausgeschöpft wird. Daher ist u.a. ein wichtiges Ziel dieser Arbeit diese Mengen-Darstellung durch Visual Analytics an dieser Stelle anzugehen und zu optimieren. Das Problem dabei stellte die überlappende Mengenvisualisierung dar. Denn diese Mengen können nur bestimmte, gar kein oder alle Elemente enthalten und zudem kann eines der Elemente zu mehreren Mengen

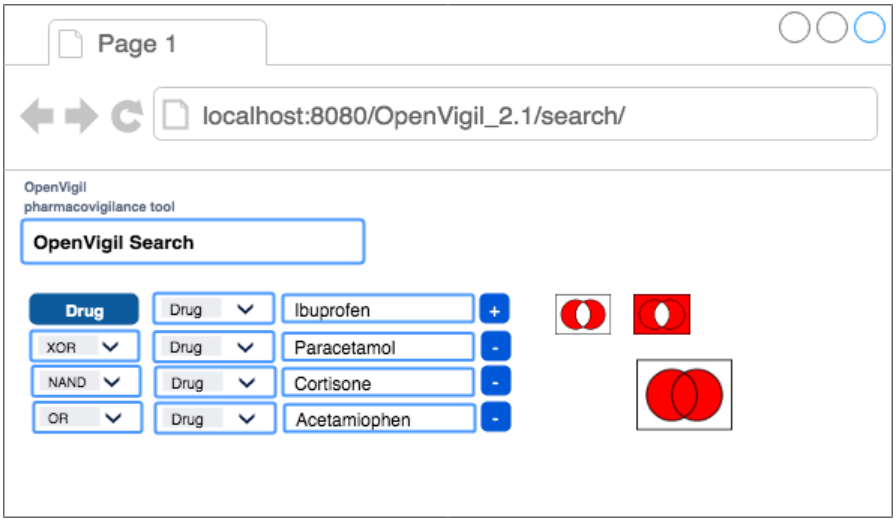


Abbildung 5.7: Mockup Search Komponenten

gehören. In der Pharmakovigilanz sind es z.B. die Nebenwirkungen mehrerer zum Teil ähnlicher Medikamente. Jedes Medikament kann viele verschiedene Nebenwirkungen haben, von denen einige auch von anderen Medikamenten verursacht werden können. Da es sehr wahrscheinlich ist, dass man bei Mengen, Überschneidungen und Überlappungen von diesen intuitiv an Kreise denken muss und Kreise den Mengen-Charakter sehr gut wiederspiegeln, war die Idee dieses Vorhaben zu realisieren, die einzelnen Mengen als Kreise darzustellen. In Anlehnung an Venn-Diagramme sollte die Operatoren-Auswahl anhand der Färbung dargestellt werden. Das folgende Mockup geht auf diese Art von Darstellungsformen ein. Man sieht im Beispiel vier verschiedene Medikamente, die mit drei verschiedenen Operatoren verknüpft sind. Rechts daneben sind die Kreise abgebildet, die je nach Operator unterschiedlich gefärbt sind.

Visualisierung der Resultate

Die Ergebnismenge ist von den einzelnen Eingaben abhängig. Zum einen kann der User zwischen verschiedenen Evaluations-Methoden wählen und zum anderen kann er durch die erweiterte Suche an den Eingangsparametern Änderungen vornehmen. Die voreingestellte Evaluations-Methode ist die *Frequentist_ methods*. Diese zeigt in Abhängigkeit zu den zum eingegebenen Wirkstoff aufgezeichneten Nebenwirkungen die durch statistische Analysen ermittelten Werte an. Bei nicht fundierten Kenntnissen über die statistischen Methoden ist es schwer zu sagen, in welchem Zusammenhang diese stehen und welche

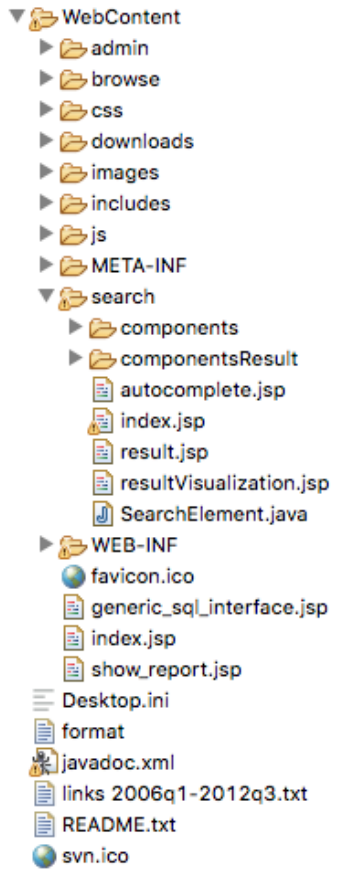


Abbildung 5.8: Eclipse View des WebContents

Um zu sehen, wo die neuen Komponenten im Sourcecode eingebettet werden können, wurde untersucht, wo die Search-Seite aufgebaut worden ist. Die Struktur der Startseite wird in dem File index.jsp in dem WebContent-Ordner überwiegend aufgebaut, wobei man sich Scriplets zu Hilfe nimmt, sodass Teile des HTML-Code-Aufbaus in einer FormHelper-Klasse ausgelagert wird 5.8. Was gut zu sehen ist, ist dass sowohl der ganze Drug- als auch Advent Event-Search-Bereich jeweils nur in einem div-Element leben. Das bedeutet, dass auch die Such-Elemente innerhalb des jeweiligen div-Elements expandieren (mit den Ids searchCriteria und eventCriteria). Dies bestätigte erst einmal die Annahme, dass man rechts neben den Such-Elementen ein weiteres abgeschlossenes Parent-Element aufbauen könnte, was die Visualisierung realisierte.

Korrelation eventuell vorhanden sein könnte. Aus diesem Grund habe ich mir erst einmal die Evaluations-Methode *Raw_ data* betrachtet. Diese stellt wie der Name schon sagt Rohdaten da. Beginnend bei der Reportnummer (Individual Safety Report, ISR) gibt sie u.a. Aufschluss über folgende Parameter:

- ISR
- Event
- Case_id: Fall-ID
- Originalnamen

- Dosis
- Geschlecht
- Reporter_country
- Indikation
- Age_in_Report
- Calculated_daily_dosis
- Outcome
- Drug_seq
- Role_Code
- Pharmaproduct

Dies sind ziemlich viele Ausgaben und in einer Tabelle dargestellt, verliert man leicht den Überblick. Es sind zu viele Werte und in Tabellenform kaum überschaubar. Daher eigent sich an dieser Stelle eine Darstellungsform, die auf einen Blick die Informationen und die Zusammenhänge der einzelnen Werte, was bei einer Tabelle nur schwer gelingt, schon einmal auf der oberen Detail-Ebene darstellt. Da habe ich für den ersten Versuch an ein Netzwerk gedacht. Da sich dadurch vor allem ein sehr guter Überblick verschaffen lässt. Sie initialisieren im Endeffekt die darauf aufbauenden Schritte, die weiterführende und detailliertere Analysen erlauben. In der Abbildung 5.9 steht die Suchanfrage in der Mitte, aus der die in Verbindung stehenden Reports (ISRs) entspringen. Dadurch wird direkt sichtbar, in wie vielen Reports die betrachtenden Wirkstoffe vorkommen. An dem jeweiligen Report-Fall hängen wiederum die darin vermerkten Informationen, wie z.B. das Alter des Patienten, die Dosis, die Indikation und das Land, aus dem der Report stammt. Natürlich muss beachtet werden, dass dieses Rohdaten sind und auch nicht in jedem Report alle Parameter einen Wert haben. Dies darf nicht vernachlässigt werden. Doch es ist ein gutes Instrument, um darauf weitere Analysen aufzubauen. so könnte man nun nach dem jeweiligen Report recherchieren oder man könnte sich anhand dieser visuell verdeutlichten Informationen statistische Analysen vornehmen. Ich habe mir bewusst v.a. für den Anfang der Arbeit mit OV diese Auswertungs-Methode ausgesucht, um auch selbst einen Überblick über die Daten und über die Möglichkeit von Visual Analytics-Aspekten bezüglich der Rohdaten zu bekommen.

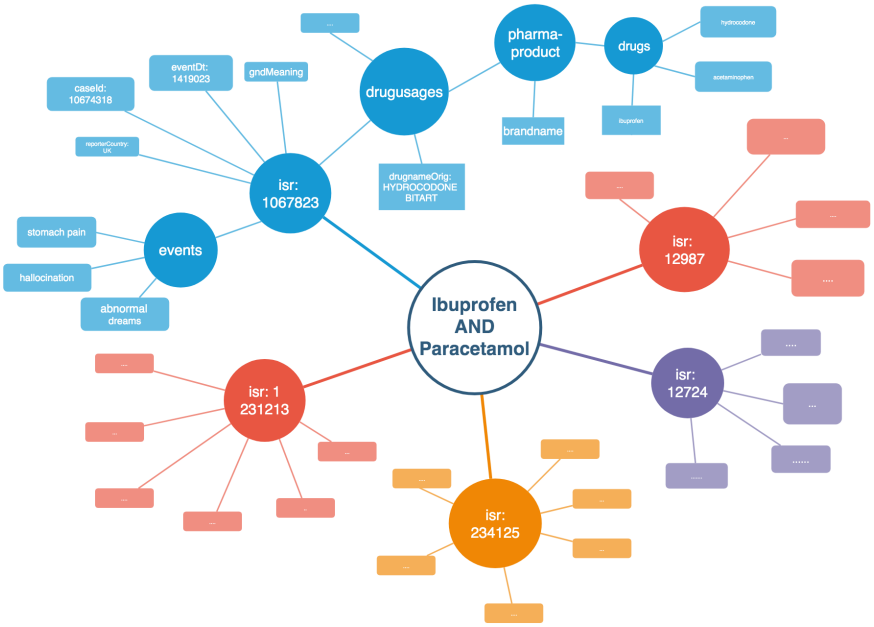


Abbildung 5.9: Mockup Result Network

5.2 Prototyp

Nachdem die Analyse der Anforderungen und die Entwurfsvorstellungen standen, wurden Überlegungen getätigt, welche Technologien dafür gebraucht werden und welche sich für das Realisieren der Anforderungen eignen. Da wir im Studium keinerlei Berührungspunkte mit Frontendentwicklung haben und ich leider auch privat keine Erfahrungen im Frontend-Bereich machen konnte, war diese Aufgabe für mich komplettes Neuland. Also recherchierte ich erst einmal, was es im Bereich Frontend für Libraries oder Frameworks bezogen auf die Daten-Visualisierung momentan gibt. Es stellte sich schnell heraus, dass es in diesem Bereich sehr, sehr viele Libraries und Frameworks gibt. Nach vielen Berichten und Meinungen viel auf, dass React und D3 immer wieder erwähnt wurden und herausstachen, was die Popularität und die Funktionalität angingen. Sowohl D3 als auch React sind beide JavaScript-Libraries, die beide er ermöglichen, Datensätze im Web zu visualisieren. React hat eine überschaubare und leicht verständliche API. Das heißt React ist relativ schnell zu lernen und leicht anzuwenden. Zudem ist React ausschließlich für den View-Part zuständig, was es unabhängig von anderen Techniken macht. Außerdem gibt es für React sehr gute Entwicklungs-Tools und eine aktive Community. Durch diese Faktoren und mit Rücksprache mit meiner Betreuerin entschieden wir uns für React [6].

5.2.1 Technologien

Die zur Implementierung ausgewählte Library ist eine Frontend-Technologie, die von Facebook entwickelt worden ist und 2013 als Open-Source-Lösung zur Verfügung gestellt worden ist. React zeichnet sich durch die Komponenten-Bildung und durch ihre Komposition aus [13]. Eine in React geschriebene Webanwendung wird im DOM des Browsers gerendert. Die die Anwendung kann auf dem Server gerendert werden und dann kann das fertige HTML an den Browser übermittelt werden, der das HTML an den DOM weiterreicht, der für die Repräsentation der dargestellten Objekte zuständig ist. Ab dem Zeitpunkt werden alle Updates im Browser gerendert, was sehr praktisch und schnelle Anzeige-Antworten verspricht. React über CDN-Links zu verwenden ist möglich und verbessert nicht nur die Performance sondern reduziert auch die Latenz. Zudem lassen sich dadurch React-Komponenten leicht integrieren. Doch durch CDNs hat man keinen Zugriff auf die meisten React Features. Bei React ist es eine Herausforderung sich nach einem schnellen Einstieg auf eine der vielen Module und Libraries zu entscheiden, die die Aufgaben der Anwendungsfälle erfüllt und den Mehrwert verspricht. React verspricht eine schnelle Lernkurve und schnelle Ergebnisse, auch wenn diese zu Beginn vielleicht etwas kleiner sind. Empfohlen wird, ein neues Projekt mit dem React-Tool *create-react-app* zu erstellen, da man direkt mit einer vorgegebenen einfachen Projektstruktur einsteigen kann und sich nicht selbst darum kümmern muss. Ohne Veränderung kann das Projekt direkt mit dem Node-Package-Manager (npm) mit dem Befehl *npm start* gestartet werden. Was für React wichtig ist, sind die *index.html*-Datei, die einen Platzhalter div inne hat, die *App.js*, die die React-Komponente ist und *index.js*, die die Komponente mit *ReactDOM.render()*. React in eine HTML-Seite einzubetten ist relativ leicht. Durch die Funktion der DOM-API *document.getElementById('root')* wird der Knoten mit der angegebenen Id gesucht und an der Stelle die neue React.Komponente dargestellt. Dabei wird der Inhalt des Knotens ersetzt. Einer der großen Vorteile, den React mit sich bringt, ist der des virtuellen DOMs. Denn DOM-Vorgänge, wie Abrufen, Ändern, Löschen oder Hinzufügen sind sehr teuer. Daher wurde der HTML-DOM abstrahiert und der neue virtuelle DOM entwickelt, der nichts anderes ist als ein leichtgewichtiges JavaScript-Objekt. Und daher sind die auf das virtuelle DOM angewendete Vorgänge weniger teuer als die des echten DOMs. In React können Komponenten auf zwei verschiedene Weisen beeinflusst werden:

- Properties: Eigenschaften, über die der Komponente von außen beim Erzeugen durch den Verwender Werte übergeben werden. Properties sind innerhalb dieser

Komponente unveränderlich, können aber von außen jederzeit neu gesetzt werden.

- States: Zustände einer Komponente, der nur innerhalb dieser Komponente sichtbar und veränderbar sind.

React ist eine Front-End-Library. Daher ist es nicht ihr Anliegen, zu wissen, wie die Daten abgerufen werden. React ist lediglich daran interessiert, wie diese Daten angezeigt werden. Um Daten von einem Server abzurufen, ist eine HTTP Library notwendig. Da React in Komponenten arbeitet und mit Hilfe dieser ist eine Zerlegung in kleine Module möglich, die stateful oder stateless sein können... Angefangen habe ich mit der Implementierung der React-Komponenten wie auch das OV-Grundprojekt, was sich aber dahin änderte, dass ich für React auf Visual Studio Code gewechselt habe, da es in Visual Studio Code angenehmer und leichter in der Handhabung für React ist.

Die in dieser Arbeit verwendeten Technologien sind die folgenden:

- Apache Tomcat Server 9.0
- pgAdmin 4, PostgreSQL
- IDE Eclipse
- IDE Visual Studio Code
- Java
- Ajax - axios
- Servlet
- HTML
- Javascript
- CSS
- React
- react-autosuggest
- @nivo/bubble

5.2.2 Search Prototyp

Angefangen habe ich mit der Implementierung der Visualisierung der Search-Komponenten unabhängig und los gekoppelt von dem Endsystem in Eclipse, in welches die implementierten Komponenten integriert werden können. Als Initial-Projekt wurde sowohl für den Search- als auch für den result-Prototypen mit dem react-create-Befehl erstellt. Wichtig ist hierbei die Datei *package.json*. Denn sobald sich der *node:modules*-Ordner oder die *package.json* verändert, wird die packagelock.json ebenfalls verändert. Die *package-lock.json*-Datei enthält den kompletten Dependency-Tree, um die Installation unabhängig von inzwischen aktuelleren Versionen zu reproduzieren. Zu Beginn war angedacht nur den Visualisierungs-Anteil in React zu schreiben und die Daten der Inputfelder der ursprünglichen Felder an React zu passen. Dies ließ sich nicht so realisieren, da dies keinem Standard entspricht und es nicht empfohlen ist, innerhalb eines logisch zusammenhängenden übergeordneten HTML-Elements so eine Trennung vorzunehmen und dennoch Daten von einer Komponente zu der anderen zu schicken. Also musste das komplette div-Element (searchCriteria bzw. eventCriteria) in React ausgelagert werden und somit auch die Autocomplete-Funktion, die in jedem Inputfield eingebunden ist und durch eine Ajax-Anfrage die Autosuggestions der Datenbank erhält. Denn als Javascript Funktion konnte sie so nicht in der React-Komponente verwendet werden. Für die Autosuggestions-Funktion wurde die react-autosuggest-Komponente verwendet (https://react-autosuggest.js.org/). Um das Befüllen der Autosugegstions zu realisieren, wurde AJAX verwendet. Um den Code so gut es geht nicht zu verändern, aber dennoch die alten Funktionen, wie die Autosuggest zu erhalten, wurde eine Servlet-Klasse erstellt *SearchAutocomplete.java* (im Package de.uni_kiel.pharmacy.openvigil) erstellt, die die eigehenden Anfragen steuert und verwaltet und eine Antwort im JSON-Format zurückliefert, sodass React diese dann darstellen kann [9]. Für die Visualisierung der überlappenden Mengen, habe ich versucht, nach geeigneten Libraries oder Modulen zu suchen, doch leider ohne Erfolg. Es gab vielleicht kreisförmige Darstellungen, aber keine Möglichkeit diese auf unterschiedliche Weise in den verschiedenen Sektoren mit Farbe zu befüllen. Also war die Lösung diese Venn-Diagramme mit flexiblen und interaktiven Veränderungen selbst zu gestalten und zu implementieren. Dafür verwendete ich u.a. die SVG-Elemente *circle* und *rect*. In Abbildung 5.10 ist ein Auszug aus dem DOM zu sehen, der die zur interaktiven Färbung verwendeten relevanten Elemente abbildet. Was sich zu dem als schwierig für die Visualisierung gestaltete, waren folgende Faktoren:

- Parent konnte nicht ausgefüllt werden und es konnten dennoch Children hinzuge-

Abbildung 5.10: DOM Ausschnitt der Kreis-Komponente

fügt werden.

- Beliebige Konstellation und Menge der Kinder möglich.
- Mögliches Löschen von jedem beliebigen Kind.
- Lange Arzneimittelnamen erschweren Darstellung.

Der erste Punkt wurde in soweit verändert, dass keine Kind-Knoten hinzugefügt werden kann, wenn das Parent-Feld nicht ausgefüllt ist. Für die Verwaltung der einzelnen Kinder wurde eine Liste von Kind-Komponenten gehalten, die dann Der Aufbau dieser gesamten Search-Komponente baut sich aus sechs verschiedenen Komponenten auf:

1. < SearchComp /> : Parent-Komponente, die die Kinder verwaltet.
2. < ChildComp /> : Child-Komponente, die sich um eine Zeile eines Suchfeldes
3. < TypeDropDown /> : Komponente, die den Typ-Select erstellt.
4. < OperatorDropDown /> : Komponente, die das Select für die Operatoren hält
5. < Autosuggest /> : Autosuggest-Funktion
6. < CircleVennComp /> : Venn-Komponente, die die ausgewählten Drugs und die verschiedenen Färbungen realisiert.

Für den Intersektions-Bereich zwischen zwei Drugs wurde die Schnittmenge ermittelt, sodass diese für die Färbung ansprechbar ist und mit Farbe gefüllt werden kann. Für die Färbung der Menge außerhalb der Kreise (NAND-Operator) wurde ein Rectangle erstellt. Die Färbung der einzelnen Bereiche der Kreis-Zusammenstellungen wird anhand der drei Füll-Parameter *fillColor* (setzt die Farbe der Kreissegmente, die nciht zu der Schnittmenge gehören), *fillRect* (setzt die Farbe des Rectangles, der für den Operator NAND notwendig ist) und *fillIntersect* (setzt die Farbe der Schnittmenge der Kreise).

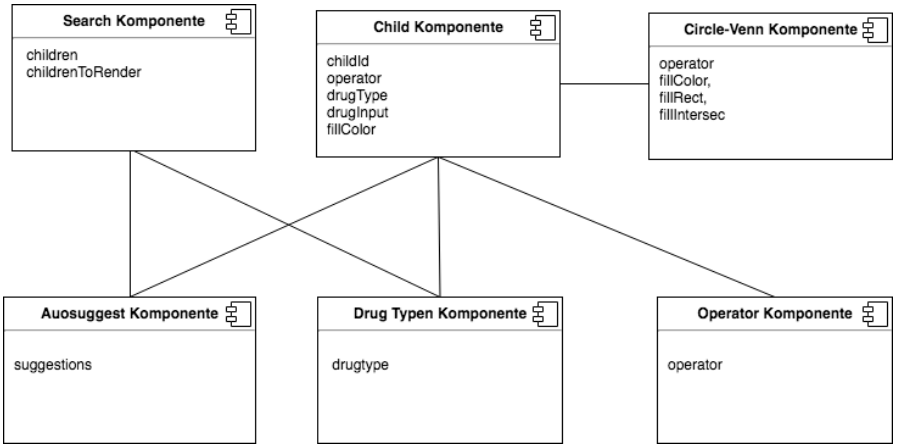


Abbildung 5.11: Komponenten der Search-Elemente

All diese drei Werte werden vom Parent an die veränderten Kinder weitergereicht. Das Endresultat der Erweiterung der Suchseite ist in der Abbildung 5.11 zu sehen. Um einen Überblick über die für die Suchelemente erstellten Komponenten zu erhalten, ist in Abb. 5.11 zu sehen. Für die Suche nach den Adverse Events ergibt sich analog bis auf das Drop-Down für die Adverse-Events-Typen der gleuche Aufbau wie für die Drugs. Der erste Prototyp der OpenVigil Such-Seite wird in Abbildung 5.12 abgebildet. Es sind für jedes Kind ein Kreis (Circle) abgebildet mit der Überschneidung seines zuvorgegangenen Kindes. Es werden immer nur die letzten sechs Kinder dargestellt. Im Vergleich zu OpenVigil ist der erste Button solange deaktiviert, solange das Inputfield des Parents leer ist. Das Gleiche gilt bei den Kindern für ihre DrugDropDown und OperatorDropDown. Solange das Childinputfeld leer ist, wird die nicht ausgefüllt. So ist dem User auf Anhieb klar, welche Mengen die verschiedenen Operatoren mit ihren Drugs darstellen

5.2.3 Result Prototyp

Der erste Versuch war es, die Daten durch das neu hinzugefügte Umleitungsservlet an die React-Komponente weiterzureichen. Dies war das erste Problem, da React-Komponenten unabhängig von der Umgebung leben sollen. Das heißt die Komponente sollte nicht warten müssen, bis Daten in die JSP-Seite geladen werden. Zwar wurde versucht durch ein verstecktes Element (hidden Input-Field) Daten auf der JSP-Seite zur Verfügung zu stellen, aber es war nicht möglich, diese aus React-Sicht entgegen zu nehmen. Der Versuch war vergebens, durch die Java-Server-Library JSTL die Daten an React zu vermitteln. Nach weiterer Recherche und Vertiefung in der React-Philosophie kam ich auf die Lösung

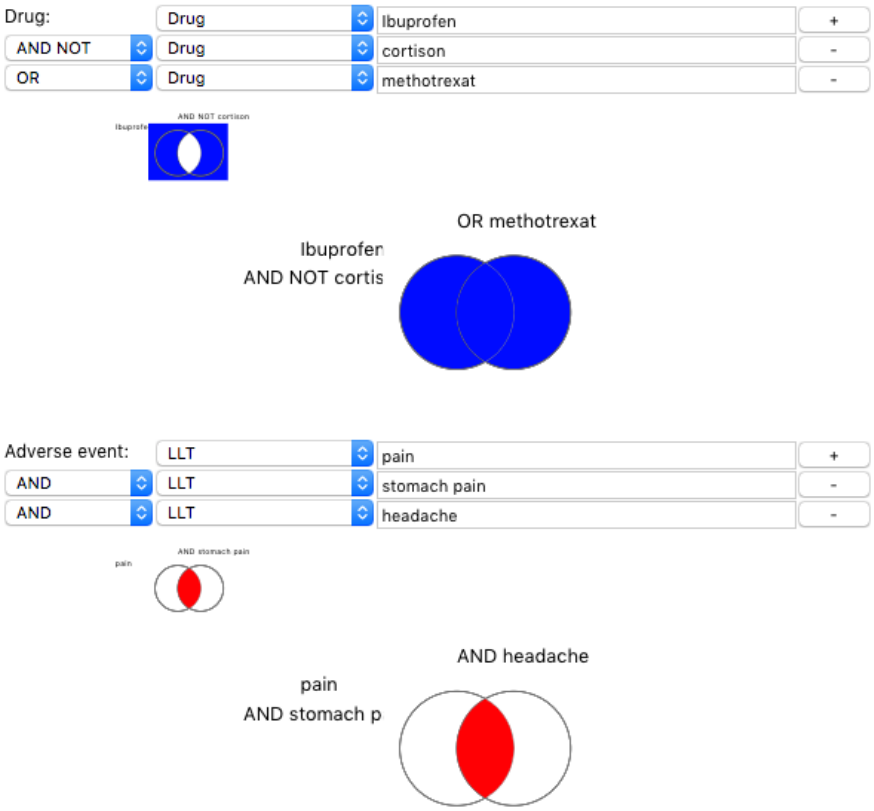


Abbildung 5.12: Visualisierung der Search-Elemente

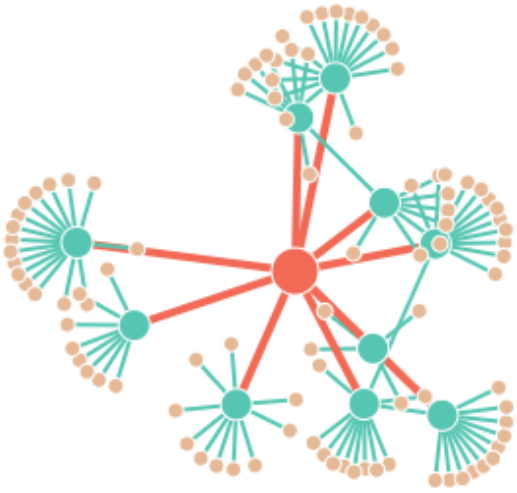


Abbildung 5.13: Network Komponente der Library Nivo.

der asynchronen AJAX-Calls. Diese würde es ermöglichen, Daten per URL anzufordern und dann die Antwort im JSON-Format vom Server entgegenzunehmen.

Der erste Versuch hierbei war es, mittels Parameter in der URL die Daten zu übertragen. Allerdings haben moderne Browser eine Begrenzung der URL-Länge, was diese Lösung für unbrauchbar machte. Denn man konnte nicht, vorhersehen, wie groß die zu übertragenden Daten werden würden. Für kleinere Datenmengen hat das auch gut geklappt, doch als diese größer wurden, versagte diese Methode.

Es musste eine neue Variante gefunden werden. So war der nächste Versuch ein Fetch-Call an das neue Servlet zu senden, welches die Anfrage verarbeitet und den Response als JSON an die React-Komponente weitersendet. Zuerst wollte ich die Visual Analytics Komponente zu den RawData aufbauen und dann anschließend nach einer sinnvollen visuellen Darstellungsform suchen.

Auf der Suche nach der geeigneten visuellen Graphik, bin ich auf die React basierte Daten-Visualisierungs-Library *nivo* gestoßen[4]. Die dadurch angebotene Komponente *network* schien auch in die Richtung meiner Vorstellungen zu gehen, doch leider war diese nicht interaktiv, sodass keine Interaktion mit dem User hätte stattfinden können (s. Abb. 5.13).

So fand ich die Bubble-Komponente, die immer weiter verschachtelte Kreise abbildet. Also hat der Parent auf der obersten Ebene alle Kinder inne, diese wiederum ihre eigenen Kinder, etc. Dies lässt sich gut mit den Raw-Data vereinbaren, vor allem um einen Überblick über alle vorliegenden Daten. In Abbildung 5.14 sieht man ein Anwendungsbeispiel der Komponente Bubble. Diese lassen sich interaktiv anklicken, woraufhin der angeklickte Bereich herangezoomt wird.

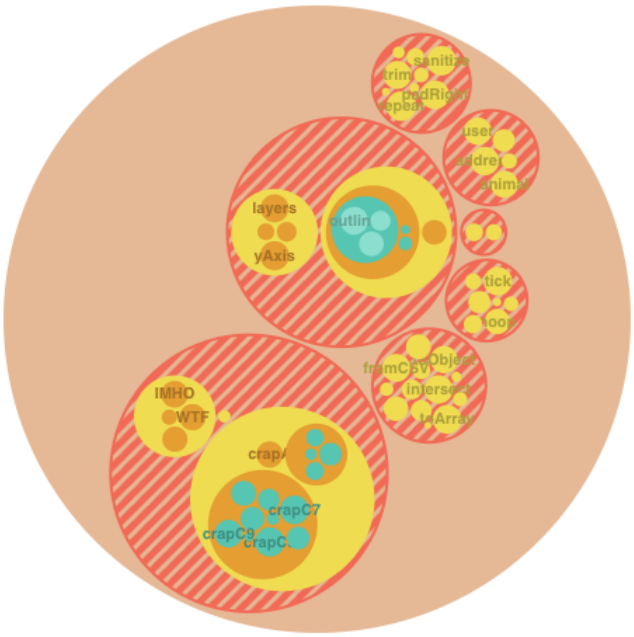


Abbildung 5.14: Bubble Chart der Library Nivo.

Nun mussten die RawData erhalten werden und in JSON-Format gebracht werden, um diese dann an React zu senden. Dies geschah alles in dem neuen Servlet *SearchUmleitungServlet*. Dieser nimmt die Anfrage entgegen, holt sich die Daten aus der Datenbank, bringt sie in das richtige Format und sendet diese an React zurück. Für die Result-Seite habe ich auf der Index.jsp Seite ein Radio-Button *rawDataVisualization* erstellt, der die Visualisierung auf einer neuen Seite *resultVisualization.jsp*. Für die Darstellung wurde das Bubble-Chart anhand der oben genannten Nivo-Library gewählt. Die aus der RawData-Methode resultierten Daten mussten in JSON umgewandelt werden und erst dann konnten sie für die Visualisierung verwendet werden. Auf der ResultVisualization-Seite befinden sich nun fünf Checkboxen, die dem User interaktiv eine davon auswählen kann. Diese fünf Checkboxen sind die folgenden:

- RawData: Zeigt alle in den RawData auftretenden ISRs mit den dazugehörigen Daten.
- Events: Zeigt alle in den Reports vorgekommenen.
- Gender: Zeigt drei verschiedene Kategorien: M: Männlich, F: Weiblich, UNK: Unknown
- Country: Zeigt die in den Reports hinterlegten Länder an.
- Age: Zeigt die Anzahl mit den vorkommenden Altersklassen an, wobei ich diese in Altersgruppen von 0 bis 100 und in 20iger Jahren unterteilt habe (0-20, ...80-100).

Um die Rohdaten in eine Form zu bringen, die sich für diese Art von Darstellung eigneten, mussten einige Veränderungen vorgenommen werden. Eine davon war die folgende: Um die Anzahl des jeweils ausgewählten Items zu ermitteln, habe ich eine Dictionary zu Hilfe genommen, deren Key den Namen des gewählten Attributs darstellte und der Key die Anzahl. In Abblidung 5.15 sieht man die Darstellung der RawData.

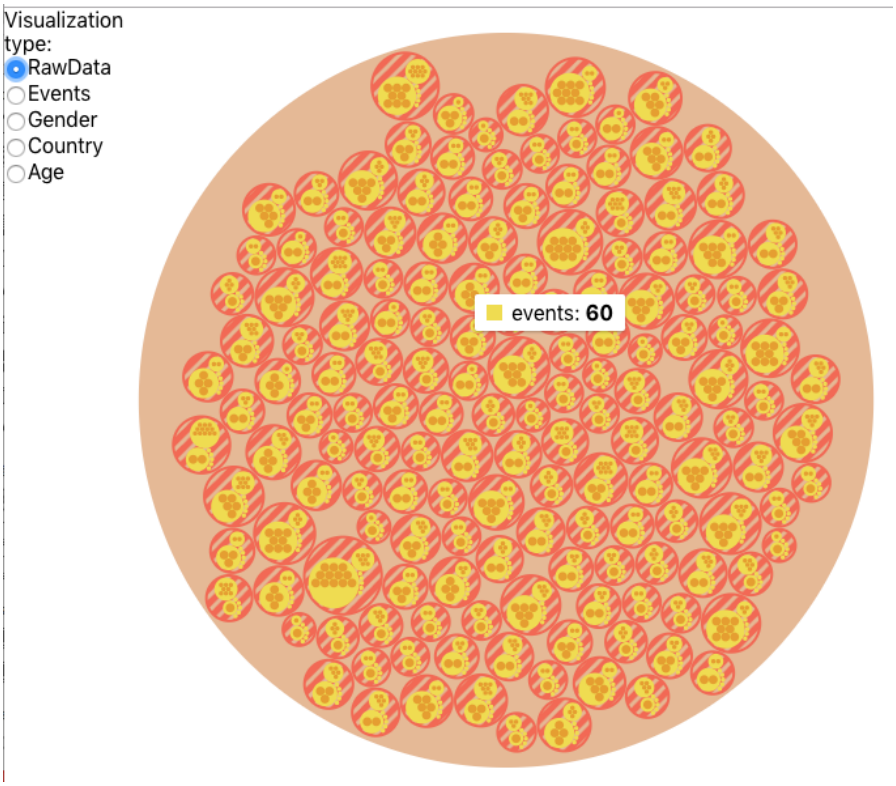


Abbildung 5.15: Resultate Methotrexate RawData

Und zum anderen sieht man die visualisierten Daten Altersklassen (s. Abb. 5.16):

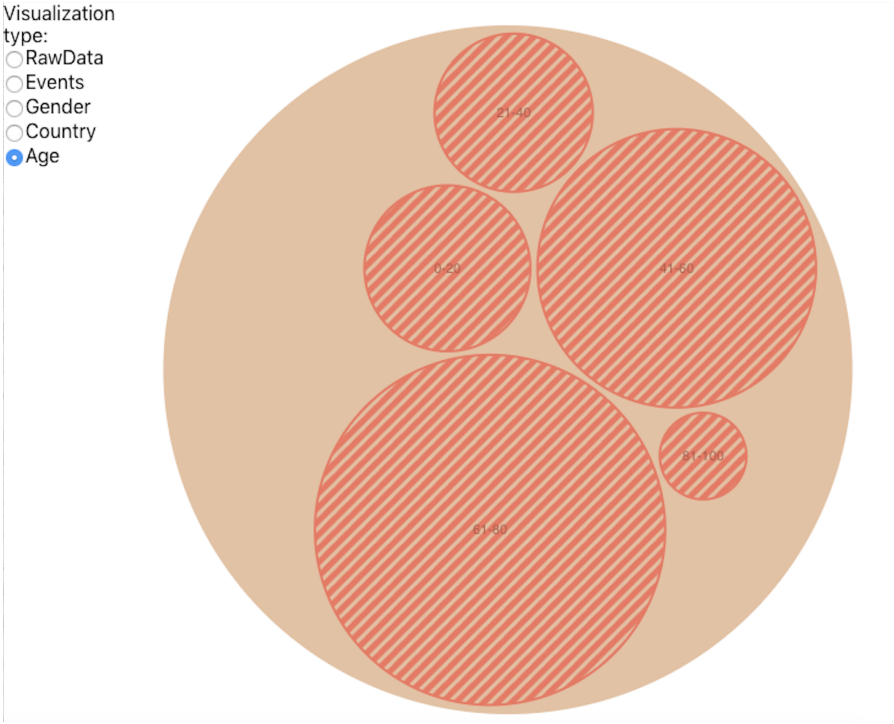


Abbildung 5.16: Resultate Methotrexate Alters-Klassen

5.3 Integration und Herausforderungen

Die erste große Herausforderung bestand schon darin, dass keinerlei Dokumentation zu diese OpenVigil-Version vorliegt. So muss viel ausprobiert und viel analysiert werden und dies kostet besonders viel Zeit. Die nächste Problematik war die Programmierung mit Scriplets und das Vermischen der einzelnen Aspekte des Model-View-Controller (MVC-Pattern). Das MVC Pattern besagt, dass eine scharfe Trennung zwischen Elementen vorliegen muss, die dem User dargestellt werden. Als Basis-Data sollen Model-Klassen zur Verfügung stehen und als Zwischenschicht für die Kommunikation zwischen Model und View sollen die Controller dienen. In diesem Projekt gibt es zwar einige Basis-Klassen, die man als Model verstehen könnte. Allerdings stimmen die Konzepte Controller und View nicht mit dem MVC-Pattern überein. Denn es gab keine richtige Trennung der View und der Controller. Bei den Search-Elementen scheiterte der erste Versuch der Integration daran, dass die Autocomplete-Funktion nicht in der React-Komponente anwendbar ist.

So musste alles in die React-Komponente ausgegliedert werden. Bei meiner Analyse und meinen Versuchen, musste ich oft feststellen, dass der vorhandene Code unterschiedliche Stile, wie z.B. Großkleinschreibung von Variablennamen bzw. keine Namenskonvention einhielt. So war z.B. die Bezeichnung der Values der Operatoren kleingeschrieben, wohingeegen alle anderen Values großgeschrieben waren. Dadurch wurden häufiger kleinere Bugs verursacht, die zum Teil auch schwer zu finden gewesen sind.

Um zu verstehen, welche und wie die Daten im Sourcecode zusammengesetzt werden, musste der Sourcecode untersucht und analysiert werden. Die JSP-Seite auf der die Ergebnisse in Tabellenformen zusammengestellt sind, ist das *result.jsp*. Dies passiert wieder mit vielen Scriplets und dadurch in der JSP-Seite eingebundenem Java-Code. Die größte Herausforderung war es, React von Seitens des Servers einen Datenpool zu senden, den React dann wieterverarbeiten kann. Dadurch dass direkt in der result.jsp-Seite ein Objekt der Servlet-Klasse *Search.java* erstellt wurde, war es schwierig auf dem Weg Zugang zu den Daten zu finden. Um eine saubere Trennung aufbauen zu können und die Daten in dem gewünschten JSON-Format an React weiterzuleiten, bietete sich ein Servlet an, der im Endeffekt als eine Art Umleitung funktioniert wie bereits erwähnt und je nach ausgewählter Methode zu der result.jsp-Seite oder zu einer neuen und sauberen resultVisualization.jsp weiterleitet. So kann die Visualisierung sauber und ohne Scriplets auf einer neuen JSP-Seite aufgebaut. So ist zumindest der Visualisierungs-Teil sauber als View vom restlichen Code getrennt. Eine bessere Lösung war in der vorhandenen Umgebung leider nicht möglich.

Zudem war es auch zu Beginn schwierig, herauszufinden, welche Möglichkeiten und auf welche Weise, man ein fertig gestelltes React-Projekt in ein altes Java-Projekt einbaut und integriert. Nach gesammelten informationen im Internet kam ich auf eine mögliche Art und Weise die Systeme zusammenzuführen. So kann ein Ordner mit den Build-Dateien im Projekt hinterlegt werden und bei der Verwendung einer Komponente daraus wiederum auf dieses verwiesen werden. Zusätzlich müssen auf der Seite, in der die React-Komponenten intergriert werden sollen, Scripts des gebauten Projektes hinterlegt werden.

5.3.1 Integration Search-Elemente

Integration gestaltete sich schwieriger als gedacht. Die fertigen Komponenten für die Visualisierung ließen sich nicht ohne Weiteres in den Bereich der *Searchcriteria* bzw.

Eventcriteria einbinden. Da an den Eingabefeldern der Ausgangssuche und den hinzufügbaren Kinder-Suchfeldern eine Autocomplete-Funktionalität gebunden ist, die durch ein JQuery-Widget im ursprünglichen Projekt realisiert wurde. Das heißt an dieser Stelle hätte diese Funktion an die React-Komponenten weitergereicht werden müsste, was nicht funktioniert, da diese Funktion interaktiv je nach Eingabe des Users über den Tomcat-Server mit der Postgres-Datenbank kommuniziert.

5.3.2 Integration Result-Elemente

Um die Daten für React vorzubereiten und in einen logischen Zustand zu übertragen, wurde versucht eine POJO-Klasse (Plain Old Java Object) zu erzeugen (RawDataVis). Damit anhand eines ObjectMappers die Felder der Modell-Klasse leicht in einen JSON-String transformierbar sind, sodass man in React in einer einfachen Weise an weitere Libraries übertragen und im Sourcecode direkt benutzen könnte. Die Verwendung der POJO-Klasse Report. Damit wurde der JSON-String erstellt, der alle Daten zu den eingegebenen Feldern enthält. Dieser wurde dann versucht per URL an React geschickt zu werden. Leider war dieser zu groß (org.apache.coyote.http1.HeadersTooLargeException). So werden die Daten durch Post einmalig an React weitergereicht werden und stehen allen anzuwendenden Visualisierungformen zur Verfügung.

6 Fazit

Dadurch dass nicht nur React sondern auch der gesamte Bereich des Frontends neu für mich sind, hat die Einarbeitung sehr viel Zeit in Anspruch genommen. Ich musste mir erst einmal einen Überblick verschaffen und abwägen, welche Module es in React gibt, die in Frage kommen könnten. Zudem ist es sehr schwierig sich ganz ohne Dokumentation in dem Projekt zurecht zu finden. Dadurch dass immer wieder kleine Fehler (wie z.B. inkonsistente Daten) auftraten, bremste dies einen aus. So musste man sich teilweise mehr als gewollt und gedacht um Backend-Themen kümmern, da alles vermischt war. Es ist klar, dass die Software nicht regelmäßig und kontinuierlich auf dem Stand gehalten wird und kann. Doch wenn man dies nicht tut, liegt die Website irgendwann brach und stirbt irgendwann ganz. Ich finde soweit muss es nicht kommen. Auch wenn die Webapplikation etwas verwirrend und nicht an allen Stellen strukturiert aufgebaut ist und nicht standardisierten Patterns (wie MVC) folgt, ist sie dennoch eine Grundlage, um das Projekt neu zu strukturieren. Ich finde für den kurzen Zeitraum, ist dies ein richtiger Weg in Richtung Optimierung und Verbesserung dieser Website. Doch sehr gerne hätte ich weiter an der Ausarbeitung weiterer Visual Analytics Aspekte gearbeitet, sodass stärkere Aussagen getroffen werden können. Denn durch die Visual Analytics kann man vor allem in so großen Datensätzen, wie man sie auch hier in den Rawdata vorfindet.

6.1 Diskussion

Rückblickend auf das Projekt kann man sagen, dass leider ein großer Schwerpunkt darauf ausgelegt war, wie und ob man erfolgreich mit dem bestehenden System kommunizieren zu kann. Man musste Wege finden, wie man mit dem System interagieren kann und wie man Informationen weiterreichen und wie man diese erhalten könnte. Gefühlt war man vor allem am Anfang der Arbeit nur damit beschäftigt, kleine Bugs, die wiederum andere verursachen, zu beheben. Dadurch dass gar keine Dokumentation vorhanden war, lief man auch sebst vielleicht ein paar inkorrekte Wege, bevor man zu einem funktionierenden

Lösungsweg fand. Es ist vor allem bei so einem Projekt wie OpenVigil natürlich auch verständlich und klar, dass wenn es Open Source ist und keine wirkliche zentrale verantwortliche und vor allem aktive Stelle hat, die sich ausgiebig mit dem Projekt beschäftigt, es nicht zeitgemäß dem aktuellen Zustand der Technologie entspricht. Doch vor allem in so einem Fall, dass in relativ kurzer Zeit ein solches System erweitert werden soll, ist eine solche Erweiterung umso komplizierteren und mühsameren Aspekten ausgeliefert. Das System ist technisch gesehen eher mit älteren Technologien entwickelt worden, sodass eine Erweiterung von relativ modernen Libraries, Frameworks, etc. schwierig bis nicht zu empfehlen ist. So führt die Verwendung von Scriplets nicht nur zu einem unlesbaren Sourcecode, sondern erschwert erheblich das Debuggen von Fehlern. Denn auf Client-Seite werden Server-seitige Code-Schnipsel ausgeführt, die man mit einem normalen Debug-Flow nicht erkennen kann. Zudem wird dadurch ein grundlegender Aspekt in der Webapplikation missachtet. Es wird nicht nach dem MVC-Modell modelliert. So gestaltet sich nicht nur die Lesbarkeit als schwierig, sondern auch die Wartbarkeit und Erweiterung. Es wird nicht nach einem roten Faden implementiert, sodass auch die Struktur darunter leidet.

Einmal dass diese Hürden genommen worden sind, konnte sich auf die Visual Analytics Aspekte konzentriert werden

Dadurch dass keine Library odr Framework zur Visualisierung des überlappenden Mengen-Problems gefunden worden ist, musste hier für die Search-Seite eine eigene Visualisierungsform implementiert werden, was in der Zeit dazu führte, dass diese Graphik im Vergleich zu der fertigen Library, die auf der Result-Seite verwendet wird, vom Design her einfacher und simplerer gestaletet wurde. Allerdings mussten alle Sektionsbereiche, die abhängig von der interaktiven Operator-Auswahl des Benutzers auf verschiedene Weisen gefärbt werden sollten. Dies ist der erste Prototyp zu den Suchanfragen, der auf langfristige Sicht erweitert werden soll. Die Benutzer sind nun in der Lage, die Operatoren und die Suchbegriffe auf eine interaktive visuelle Weise zu analysieren und leichter zu verstehen, wie die gesuchten Begriffe in der Mengen-Relation zueinander stehen, sodass mathematische Grundkonzepte besser vermittelt werden können. Die Visualisierung der Mengen wird zusätzlich unterstützt, indem Farben eingesetzt werden. Die Adverse Events sollen mit der Signalfarbe Rot im Bezug auf das Blau der eingegbenen Drugs hervorgerufen werden. Das Ziel so wenige wie möglich in den vorhandenen Code einzugreifen, wurde auf zwei zentrale Art und Weisen erreicht. Zum einen wurde auf Client-Seite, wie bereits erwähnt, ein Radio-Button für die Visualisierung hinzugefügt und auf Server-Seite wurde ein Zweig mittels eines Servlets gebaut, sodass man auf eine eigene Weise umlei-

ten konnte. Dazu wurde das Projekt im WebContent um zwei Ordner erweitert, die die React-Komponenten zentral an einem Ort halten. Dieser Weg ist nachvollziehbar, sodass bei weiterer Entwicklung von React-Visualisierungskomponenten, das Projekt leichter und verständlich erweiterbar ist. Ein weiteres Ziel, das erreicht wurde, ist mir auf der result-Seite gelungen. Dort ist der User nun in der Lage anhand der Radio-Buttons interaktiv Wissen über die Raw-Data zu gewinnen. Per Klick kann der Benutzer z.B. sehen, wie sich die Verteilung von Männern und Frauen bezogen auf seine Anfrage gliedert. Zudem kann dem User auch ein Einblick über die Verteilung der Länder, in denen die Reports aufgenommen worden sind, gewährt werden. So kann der User direkt mit einem Klick diese Informationen visuell wahrnehmen, was in der vorherigen Version nicht möglich gewesen ist. In der Tabellenform wurde der User eher von Informationen überwältigt, sodass der Informationsgewinn aus so einer Tabellenform geringer ist als der der nun möglichen Darstellungsform. Zudem ist die Weiterentwicklung von Unterkategorien zu den bestehenden Auswahlmöglichkeiten leicht zu realisieren, die durch die Struktur der React-Komponente gegeben ist. React weist sehr viele verschiedene Module auf und stellt u.a. dadurch viele Anwendungsmöglichkeiten dar. Um so ein Projekt äm Lebenżu erhalten, sollte eine kontinuierliche Weiterentwicklung durch VA-Konzepte verfolgt werden. VA Methoden sollen das Programm unterstützen, sodass der Benutzer mehr und auf eine effizientere Art und Weise Informationen aus der großen zur Verfügung gestellten Datenmenge gewinnen kann.

6.2 Ausblick

Diese Arbeit könnte als Anstoß für einen kompletten Umbau der Webseite gesehen werden, angefangen mit explorativer Analysen bis hin zu ausgebauten interaktiven Komponenten. Dadurch könnte mehr aus der Website sowohl visuell als auch informativ herausgeholt werden.

Diese Arbeit soll als Grundlage für die weiterführende Gestaltung der Web-Applikation mit Hinblick auf Visual Analytics eventuell in Form des Master-Grundprojektes dienen, sodass ein längerfristiges Ziel anvisiert wird. Das langfristige Ziel ist es, diese Webapplikation mit Leben und mit Interaktionen zu füllen, sodass diese eine bessere Usability aufweisen wird und bessere Analysen ermöglichen wird, die durch Visual Analytics Konzepte unterstützt und optimiert wird. Die aktuelle Version von Open Vigil 2.1. könnte als Grundlage genommen werden für eine Neukonzeption und kompletten Umbau, begonnen

mit explorativen Analysen bis hin zu visuellen Analysen mit interaktiven Komponenten.

Literaturverzeichnis

[1] *AkdÄ. Arzneimittelkommission der deutschen Ärzteschaft.* 2019. – URL https://www.akdae.de/. – Letzter Zugriff am 20.07.2019

[2] BfArM (Veranst.): *BfArM. Bundesministerium für Arzneimittel und Medizinprodukte.* 2019. – URL https://www.bfarm.de/DE/Arzneimittel/Pharmakovigilanz/RisikenMelden/_node.html/. – Letzter Zugriff 30.07.19

[3] Bundesministerium der Justiz und für Verbraucherschutz (Veranst.): *Gesetz über den Verkehr mit Arzneimitteln.* 2019. – URL https://www.gesetze-im-internet.de/amg_1976/. – Letzter Zugriff am 19.07.2019

[4] *Nivo. A React Library.* 2019. – URL https://nivo.rocks/. – Letzter Zugriff am 16.08.19

[5] *OV. Open tools for data-mining and analysis of pharmacovigilance data.* 2019. – URL http://openvigil.sourceforge.net/. – Letzter Zugriff am 12.08.19

[6] *React. A JavaScript library for building user interfaces.* 2019. – URL https://reactjs.org/. – Letzter Zugriff am 16.08.19

[7] *WHO. The WHO Programme for International Drug Monitoring.* 2019. – URL https://www.who.int/medicines/areas/quality_safety/safety_efficacy/National_PV_Centres_Map/en/. – Letzter Zugriff am 18.07.2019.

[8] Balzert, Helmut: *Lehrbuch der softwaretechnik: Basiskonzepte und requirements engineering.* Springer-Verlag, 2009

[9] Basham, Brian ; Sierra, Kathy ; Bates, Bert: *Servlets and JSP von Kopf bis Fuß.* O'Reilly Verlag, 2009

[10] Böhm, Ruwen ; Hehn, Leocadie von ; Herdegen, Thomas ; Klein, Hans-Joachim ; Bruhn, Oliver ; Petri, Holger ; Höcker, Jan: OpenVigil FDA–inspection of US American adverse drug events pharmacovigilance data and novel clinical applications. In: *PLoS One* 11 (2016), Nr. 6, S. e0157753

[11] Cook, Kristin A. ; Thomas, James J.: Illuminating the path: The research and development agenda for visual analytics. (2005)

[12] Dill, John ; Earnshaw, Rae ; Kasik, David ; Vince, John ; Wong, Pak C.: *Expanding the frontiers of visual analytics and visualization.* Springer, 2012

[13] Hartmann, John ; Zeigermann, Oliver: *Expanding the frontiers of visual analytics and visualization.* dpunkt.verlag, 2016

[14] Keim, Daniel: Mastering the information age: solving problems with visual analytics. (2010)

[15] Keim, Daniel A. ; Mansmann, Florian ; Thomas, Jim: Visual analytics: how much visualization and how much analytics? In: *ACM SIGKDD Explorations Newsletter* 11 (2010), Nr. 2, S. 5–8

[16] Lamy, Jean-Baptiste ; Berthelot, Hélène ; Favre, Madeleine ; Ugon, Adrien ; Duclos, Catherine ; Venot, Alain: Using visual analytics for presenting comparative information on new drugs. In: *Journal of biomedical informatics* 71 (2017), S. 58–69

[17] Wong, Pak C. ; Thomas, Jim: Visual analytics. In: *IEEE Computer Graphics and Applications* (2004), Nr. 5, S. 20–21

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Integration von Visual-Analytics-Komponenten in die Plattform Open Vigil

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original