ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Medizinische Systembiologie

Automatische Generierung von Netzwerkregeln mit Hilfe von OCR und NLP

Abschlussarbeit an der Universität Ulm

Vorgelegt von:
Susanne Jessica Bair
susanne.bair@uni-ulm.de
920170

Gutachter:
Prof. Dr. Hans A. Kestler

Betreuer:
Dr. Axel Fürstberger, Dr. Julian Schwab, Dr. Silke Werle

2021

Fassung 30. Oktober 2021

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Um langfristige Verhaltensweisen von biologischen Prozessen vorherzusagen, können Netzwerkmodelle simuliert und analysiert werden. Eine Möglichkeit dazu bilden die Bool'schen Netzwerke, deren Aufbau durch Bool'sche Funktionen, auch genannt Netzwerkregeln, erfolgt. Dabei können die benötigten Informationen, die zum Erstellen von Netzwerkregeln benötigt werden, aus biomedizinscher Literatur extrahiert werden. Dieser Vorgang ist jedoch zeitaufwendig, weshalb in dieser Arbeit ein Prototyp mit geeigneter Nutzeroberfläche entwickelt wurde, der die Generierung von Netzwerkregeln aus wissenschaftlichen Arbeiten im PDF-Format automatisiert. Dieser Vorgang erfolgt mit Hilfe von „Optical Character Recognition “ (OCR) und „Natural Language Processing “ (NLP), indem der enthaltene Text vollständig extrahiert und anschließend mit natürlicher Sprachverarbeitung untersucht wird. Zusammen mit „Integrated Network and Dynamical Reasoning Assembler“ (INDRA) können die Ergebnisse in das Format der Bool'schen Funktionen überführt werden. Eine Evaluation verschiedener Parameter hat gezeigt, dass der Prototyp in angemessener Zeit vollständige Publikationen verarbeiten und korrekte Netzwerkregeln generieren kann.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meines Studium begleitet und unterstützt haben.

Weiterhin möchte ich Herrn Prof. Dr. Hans Armin Kestler für die Möglichkeit danken, diese Bachelorarbeit in seinem Institut schreiben zu dürfen.

Besonderer Dank gilt meinen Betreuern Herr Dr. Axel Fürstberger, Herr Dr. Julian Schwab und Frau Dr. Silke Werle, für die stetige Unterstützung im Laufe der Arbeit und das ausführliche Feedback, das stets in kurzer Zeit bereitgestellt wurde.

Zuletzt möchte ich mich bei Christoph Eggart bedanken, der mich nicht nur motiviert, sondern mir auch stets mit Rat zur Seite gestanden hat.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 1

1.2 Zielsetzung der Arbeit 2

1.3 Aufbau der Arbeit 2

2 Grundlagen **4**

2.1 Bool'sche Netzwerke 4

2.2 Optical Character Recognition (OCR) 6

2.2.1 Tesseract 6

2.3 Natural Language Processing (NLP) 7

2.3.1 REACH 7

2.4 INDRA 8

3 Anforderungsanalyse **9**

3.1 Anwendungsfälle 10

3.2 Funktionale Anforderungen 11

3.3 Nicht-Funktionale Anforderungen 13

4 Konzeption und Implementierung **15**

4.1 Konzeption 15

4.1.1 Ablauf 15

4.1.2 Mockups 18

4.2 Implementierung 22

4.2.1 Inputverarbeitung 22

4.2.2 OCR 24

Voraussetzungen für Bilder 24

Konfiguration der Engine 25

4.2.3 INDRA 27
NLP 28
INDRA-Statements 30
Assembler 30
4.2.4 Datenbank 31

5 Evaluation **35**
5.1 Auswertung der OCR 35
5.1.1 Prüfen des erkannten Textes 36
5.1.2 Weitere Auswertungen 37
5.2 Auswertung der generierten Netzwerkregeln 39
5.2.1 Prüfen der molekularen Einheiten 41
5.2.2 Prüfen der logischen Operatoren 41
5.2.3 Weitere Auswertungen 44
Test 2 44
Test 3 47
Vergleichen der Ergebnisse 49
5.2.4 Prüfen der Laufzeit 50

6 Diskussion **53**
6.1 Prototyp 53
6.2 Verwendete Software 54
6.3 Evaluationsergebnisse 56

7 Zusammenfassung **58**

8 Datenabhängigkeit **59**

Literatur **60**

1 Einleitung

1.1 Motivation

In jeder Zelle eines Organismus befinden sich Gene, die verschiedene Stoffwechselvorgänge steuern. Dabei können Gene auf vielzählige Weisen beeinflusst werden, was Änderungen im qualitativen Verhalten zur Folge hat. Um die Auswirkung solcher biologischen Vorgänge zu beobachten, können verschiedene Netzwerke und Graphen simuliert werden, die mathematisch aufgebaut sind und reale Mechanismen wiedergeben. Solche Netzwerke können Aufschluss über diverse Vorgänge wie Tumorwachstum oder die Auswirkung von Impfungen geben, was den Einsatz sowohl für die Biologie als auch die Medizin interessant macht. Eine Möglichkeit dazu bildet die Simulation Bool'scher Netzwerke, die trotz ihrer starken Abstrahierung Aussagen über das langfristige Verhalten von natürlichen Prozessen liefern [2]. Der Aufbau eines solchen Netzwerks erfolgt dabei durch sogenannte Bool'sche Funktionen, auch genannt Netzwerkregeln. Mit Aufstellen dieser Netzwerkregeln können Frameworks wie VisiBool genutzt werden, um Bool'sche Netzwerke zu visualisieren und simulieren und damit Änderungen im Netzverhalten aufzeigen [25]. Die Informationen, die man dazu benötigt, kann man unter anderem aus wissenschaftlicher Literatur gewinnen, da sich solche genregulatorischen Aussagen in natürlicher Sprachbeschreibung etablierten. Das Lesen solcher Veröffentlichungen und das Extrahieren dieser Genregulationsprozesse ist jedoch ein aufwendiger und zeitintensiver Prozess für den es bisher keine vollständige, automatisierte Lösung gibt. Mit dem Zuwachs moderner Techniken steigen jedoch auch die Möglichkeiten, die biomedizinische Forschung mit entsprechender Software zu unterstützen und diese Lücke zu schließen. Mit Hilfe solcher Techniken konnte somit ein Ansatz aufgestellt werden, der sich mit dem Problem der automatischen Regelgenerierung befasst.

1.2 Zielsetzung der Arbeit

Das Ziel ist also die Automatisierung dieses aufwendigen Verfahrens zur Regelgenerierung, um künftig eine effizientere Simulation und Analyse von Bool'schen Netzwerken zu ermöglichen. Den Kern der Arbeit bildet die Extraktion von genregulatorischen Informationen aus vollständigen, wissenschaftlichen Arbeiten und die anschließende Generierung von Bool'schen Funktionen bzw. Netzwerkregeln. Dabei kann man den Prozess in mehrere Teilprobleme untergliedern:

Zunächst muss der Text aus den Publikationen ausgelesen werden, um die weitere Verarbeitung zu ermöglichen. Um korrekte Ergebnisse zu erhalten, müssen dabei mehrere Faktoren berücksichtigt werden, darunter das variierende Layout der Arbeiten. Die Software-Technik „Optical Character Recognition“ (OCR) ermöglicht das Erkennen und Auslesen von Text und soll den ersten Teil des Verfahrens übernehmen. Daraufhin muss sowohl die Syntax als auch die Semantik des Textes untersucht werden, damit nach genregulatorischen Aussagen gefiltert werden kann, die zum Aufbau von Netzwerkregeln benötigt werden. Dieser Teil der Arbeit soll mit der Technik „Natural Language Processing“ (NLP) erfolgen, welche in Verbindung mit entsprechenden Datenbanken zu einer menschenähnlichen Sprachverarbeitung fähig ist. Zusammen mit der Software INDRA, die solche NLP-Systeme und andere Tools bereitstellt, können die gewonnenen Informationen verarbeitet und in verschiedene Formate überführt werden. Die Kombination dieser Techniken schafft ein Konzept, um die Generierung von Netzwerkregeln zu automatisieren. Um die allgemeine Verwendung dieses Prozesses zu ermöglichen, soll auch ein geeignetes User Interface (UI) konzipiert werden, das den Nutzern effizientes Arbeiten ermöglicht und somit auch ein nutzbares Tool entwickelt wird.

Im Laufe dieser Arbeit wird dieses Konzept als eigenständiges System implementiert und eine anschließende Evaluation soll aufzeigen, inwieweit sich der Prototyp für die Problemstellung einsetzen lässt.

1.3 Aufbau der Arbeit

Der Umfang dieser Arbeit umfasst acht Kapitel. Um Begrifflichkeiten und technische Aspekte besser zu verstehen, werden in Kapitel 2 zunächst die Grundlagen

aufgeführt. Diese unterteilen sich in vier Abschnitte, in denen genauer auf Bool'sche Netzwerke, „Optical Character Recognition“, „Natural Language Processing“, sowie die dazu verwendete Software, und INDRA eingegangen wird. In Kapitel 3 werden anhand einer Anforderungsanalyse durch Anwendungsfälle die funktionalen und nicht-funktionalen Anforderungen an den Prototyp festgelegt. Daraus wird in Kapitel 4 ein Ablauf- und UI-Konzept entwickelt, das als Basis für die Implementierung dienen soll. Die Realisierung des Prozesses zur Regelgenerierung wird darauffolgend komponentenweise aufgezeigt. Die Ergebnisse, die der Prototyp liefert, werden in Kapitel 5 in mehreren Kategorien evaluiert. In Kapitel 6 folgt eine anschließende Diskussion, in der Anmerkungen für den Prototyp genannt werden, genutzte Software begründet und die Ergebnisse der Evaluation kritisch betrachtet werden. Zuletzt soll eine Zusammenfassung in Kapitel 7 die Arbeit abschließen. In Kapitel 8 wird die Datenabhängigkeit des implementierten Prototyps aufgelistet.

2 Grundlagen

In diesem Kapitel werden Grundlagen und Begriffe erklärt, die zum Verständnis der Arbeit benötigt werden. Hierbei wird genauer auf Bool'sche Netzwerke, „Optical Character Recognition“ (OCR), „Natural Language Processing“ (NLP) und „Integrated Network and Dynamical Reasoning Assembler“ (INDRA) eingegangen. Auch die für den Prototyp verwendete Software „Tesseract“ und „REACH“ wird aufgeführt.

2.1 Bool'sche Netzwerke

In der Systembiologie werden genetische Netzwerke modelliert, um langfristige Verhaltensweisen von vielzelligen Organismen zu beobachten. Eine Zelle funktioniert und entwickelt sich dann normal, wenn bestimmte Proteine in den korrekten Zellen synthetisiert werden, und das zum richtigen Zeitpunkt [16]. Diese regulatorischen Prozesse können mathematisch beschrieben und in ein Modell überführt werden, welches somit ein reales System abstrahiert und vereinfacht darstellen kann [2]. Zu diesen genetischen Netzwerkmodellen zählen auch die Bool'schen Netzwerke.

Ein Bool'sches Netzwerk ist mathematisch definiert durch ein Tupel (V, F), wobei $V = \{x_1, ...x_n\}$ für die Menge der Knoten und $F = \{f_1, ...f_n\}$ für die Menge Bool'scher Funktionen steht [16]. Jedes $x_i \epsilon V$ mit $i = 1, ..., n$ ist eine binäre bool'sche Variable, die zwei mögliche Aktivitätszustände haben kann: 1 (aktiv) und 0 (inaktiv) [11]. Diese Variablen sind so miteinander verknüpft, dass die Aktivität einer Variable von der Aktivität einer oder mehrerer vorheriger Variablen abhängt. Dieses Verhalten wird durch die entsprechenden bool'schen Funktionen $f_i \epsilon F, i = 1, ..., n$ bzw. Netzwerkregeln beschrieben. Im Folgenden wird ein Bool'sches Netzwerk als $G(V, F)$ bezeichnet. Die Abbildung 2.1 zeigt ein simples, allgemeines Beispiel eines solchen Netzwerks mit $n = 4$.

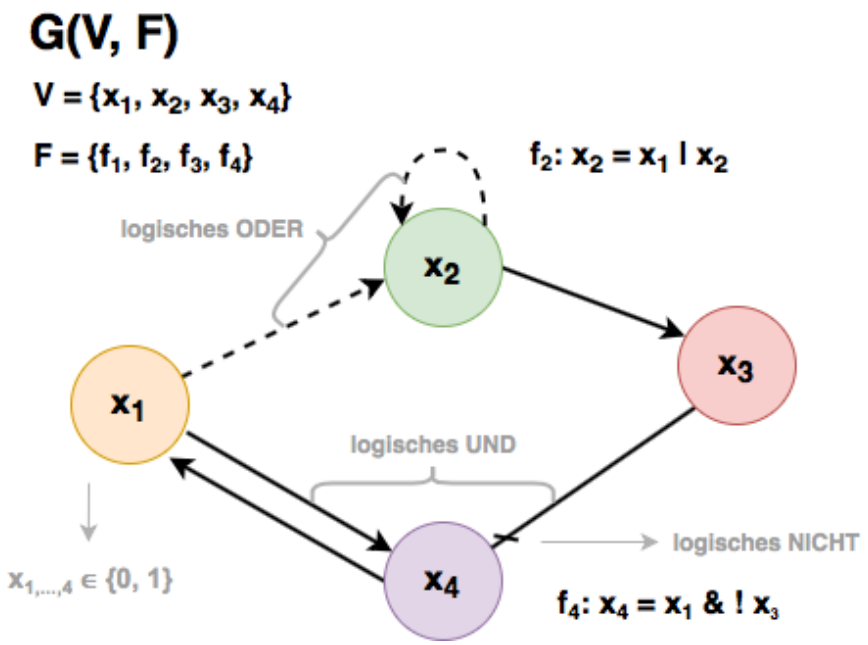

Abbildung 2.1: Beispiel eines Bool'schen Netzwerks $G(V, F)$ mit Knoten V und bool'schen Funktionen F, definiert wie abgebildet. Gestrichelte Pfeile stehen für das logische „ODER“, normale Pfeile für das logische „UND“, durchgestrichene Pfeilspitzen für das logische „NICHT“. f_2 und f_4 sind hier beispielhafte Netzwerkregeln, die von verschiedenen Knoten x abhängig sind.

Folgende Bool'schen Verknüpfungen sind hierbei wichtig, um den später gezeigten Aufbau einer Netzwerkregel zu verstehen [1]: Der logische Operator „UND“ wird als „$\&$“ dargestellt. In Bezug auf Bool'sche Netzwerke verknüpft er Variablen, die unbedingt aktiv sein müssen, um eine Variable im nächsten Schritt zu aktivieren. Der logische Operator „ODER“ wird dargestellt als „$|$“. Er verknüpft Variablen, von denen mindestens eine aktiv sein muss, um eine folgende Variable zu aktivieren. Zuletzt gibt es noch den logischen Operator „NICHT“, dargestellt durch „!“. Wird dieser Operator einer Variablen vorausgesetzt, verlangt es die Negation der Variable, um eine weitere zu aktivieren.

Die entsprechenden Netzwerkregeln werden aus solchen Verknüpfungen gebaut. Auf diese Weise können die Aktivitätszustände von Genen durch Funktionen dargestellt und als Netzwerk aufgebaut werden. Solche Bool'schen Funktionen können aus wissenschaftlichen Arbeiten extrahiert werden [15], was im Laufe dieser Arbeit mit verschiedenen Techniken in Form eines Prototyps verwirklicht wird.

2.2 Optical Character Recognition (OCR)

Um den Text aus solchen wissenschaftlichen Arbeiten zu extrahieren, wird eine Software-Technik zur Texterkennung eingesetzt. „Optical Character Recognition“ (OCR) ist die automatische Identifizierung von Text und Schrift in Bildformaten und die anschließende Umwandlung in verschiedene Output Formate.

Dabei kann man den Prozess in mehrere Schritte unterteilen [13, 19]: Zunächst wird das gegebene Bild segmentiert, indem die Schriftzeichen und Umrisse identifiziert werden. Diese werden zeilenweise betrachtet und gemessen am Zeichenabstand in Wörter unterteilt. Im nächsten und wichtigsten Schritt werden die Schriftzeichen auf ihre Merkmale untersucht und als Trainingsdaten an einen Klassifizierer bzw. Künstliches Neuronales Netz (KNN) geschickt, um die Worte zu klassifizieren. Ein erneuter Durchgang soll Worte ausbessern, die der Klassifizierer erst im Laufe der Bearbeitung gelernt hat, bevor der extrahierte Text ausgegeben wird. Damit gehört die OCR-Technik auch zum Teilbereich der Mustererkennung, des Maschinellen Lernens und der Computer Vision [13].

2.2.1 Tesseract

Als OCR-Engine wurde in dieser Arbeit Google’s „Tesseract“ gewählt, welche in der Programmiersprache C++ entwickelt wurde. Es bietet folgende Vorteile: Als Open Source Software steht Tesseract über ein Github Repository [29] zur Verfügung. Laut dieser Webseite unterstützt die Engine Unicode (UTF-8) und kann mehr als 100 Sprachen erkennen, sowie auf weitere Sprachen trainiert werden. Dabei werden die üblichen Bildformate JPEG, PNG und GIF auslesbar, aber auch weitere Formate werden durch die Bibliotheken Pillow und Leptonica unterstützt.

Tesseract bietet auch Möglichkeiten zur Konfiguration, darunter „Orientation and script detection“ (OSD), mit der Eigenschaften wie die Textausrichtung und das Layout analysiert werden. Bezogen auf den Kontext dieser Arbeit, ist dies ein wichtiger Aspekt, da kein einheitliches Layout gegeben ist.

Durch sogenannte „Wrapper“ kann Tesseract an eine Schnittstelle angebunden und sich dadurch in einige gängige Programmiersprachen einbetten lassen. Für diese

Arbeit wurde der Python Wrapper *pytesseract* verwendet. Außerdem ist dieser plattformübergreifend und unterstützt gängige Betriebssysteme wie Windows, MAC und Linux.

Einige andere OCR-Tools, die ebenfalls gute Ergebnisse erzielen, sind oft kostenpflichtig. Andere können wiederum nicht immer mit der Schnelligkeit und Genauigkeit von Tesseract (vormals HP Labs OCR) mithalten [21, 5]. Damit hat sich die Engine als sinnvolle Wahl zum Auslesen der wissenschaftlichen Arbeiten herausgestellt.

2.3 Natural Language Processing (NLP)

Um anschließend den extrahierten Text zu interpretieren, wird ein zusätzliches Tool benötigt. „Natural Language Processing“ (NLP) kombiniert verschiedene Techniken zur Analyse natürlich vorkommenden Textes, um eine menschenähnliche Sprachverarbeitung zu erreichen [6]. Zum natürlich vorkommenden Text zählt die schriftliche und mündliche Kommunikation zwischen Menschen. Dieser algorithmische Prozess ist meist Teil eines größeren Prozesses, um gegebenen Text zu interpretieren und weiterzuverarbeiten [4].

Im Bereich der Biomedizin ist die Beschreibung von Mechanismen in natürlicher Sprache gängig, um Modelle effizienter zu entwickeln und sowohl deren Transparenz als auch die Zusammenarbeit der Menschen zu unterstützen [4]. Da NLP auf genau diesen Fall ausgelegt ist, eignet es sich als Tool, um Texte aus wissenschaftlichen Arbeiten zu interpretieren und enthaltene Informationen zu filtern.

2.3.1 REACH

Im Kontext dieser Arbeit wurde das „REACH Reading System“ als NLP-Software gewählt. REACH steht für „Reading and Assembling Contextual and Holistic Mechanims from Text“ und ist damit ein Informationsextraktionssystem. Die Open Source Software ist auf biomedizinische Literatur ausgelegt und kann somit entsprechende molekulare Einheiten und deren Mechanismus extrahieren. Der Zugriff erfolgt über die REACH API über standardmäßige HTTP-GET- und POST-Methoden [3]. Der

Vorteil gegenüber anderen Systemen wie z.B. „TRIPS“ ist die Breite an Nutzungsmöglichkeiten. So kann der Zugriff beispielsweise nicht nur über einen externen Server, sondern auch über einen konfigurierbaren, lokal aufgesetzten Server erfolgen.

2.4 INDRA

„Integrated Network and Dynamical Reasoning Assembler“ (INDRA) ist eine in Python geschriebene Software, die NLP-Systeme sowie andere Software bereitstellt und mit biomedizinischen Datenbanken verknüpft, um aus natürlicher Sprache sogenannte INDRA-Statements zu bauen.

Diese Statements repräsentieren als standardisiertes Format zur Informationsdarstellung molekulare Einheiten und deren biochemische Mechanismen und können anschließend in ein Modell, ein Netzwerk oder andere Ausgabeformate konstruiert werden [4]. Dazu dienen verschiedene Assembler. Mit der Installation von INDRA können die genannten Systeme direkt importiert und genutzt werden. Somit lässt sich INDRA vielseitig einsetzen, um verschiedene Netzwerke oder Modelle aus natürlicher Sprachbeschreibung aufzubauen.

3 Anforderungsanalyse

Die Anforderungsanalyse beschreibt laut Definition „was“ ein Softwaresystem leisten soll, aber nicht „wie“ [10]. Das Ziel dieser Arbeit ist es, einen Prototyp zu entwickeln, der mit Hilfe von OCR und NLP den Text von PDF-Dateien ausliest, auswertet und daraus Netzwerkregeln generiert. Um diesen Ablauf effizient und reibungslos auszuführen, sind jedoch weitere Anwendungsfunktionen sinnvoll, die sich aus verschiedenen Szenarien ableiten lassen. Daran werden die funktionalen und nichtfunktionalen Anforderungen festgelegt, die später als Grundlage für die Konzeption und Implementierung dienen.

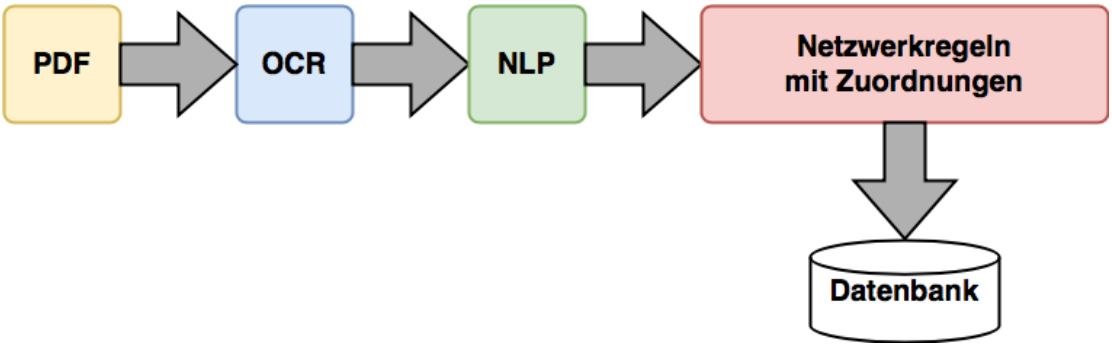

Abbildung 3.1: Das grundlegende Konzept, um Netzwerkregeln aus PDF-Dateien zu extrahieren. Dies geschieht mit Hilfe von OCR und NLP. Die Regeln sollen anschließend mit Zuordnungen (Titel, Autoren usw.) in einer Datenbank gespeichert werden.

3.1 Anwendungsfälle

An dem grundlegenden Konzept in Abbildung 3.1 lassen sich zunächst die wichtigsten Komponenten erkennen, die der Prototyp erfüllen soll. Um dieses Konzept mit sinnvollen Funktionen zu erweitern, wurden Use Cases bzw. Anwendungsfälle aufgestellt. Ein Anwendungsfall besteht aus einer Reihe von Aktionen, die Interaktionen zwischen Nutzer und System zeigen.

Abbildung 3.2 zeigt das daraus resultierende Anwendungsfalldiagramm. Dazu wurden die Komponenten in Abbildung 3.1 in Use Cases überführt und um Fälle erweitert, die nicht nur für das reibungslose Generieren von Netzwerkregeln sinnvoll sind, sondern auch für die Wiederverwendbarkeit und Vertraulichkeit der Daten.

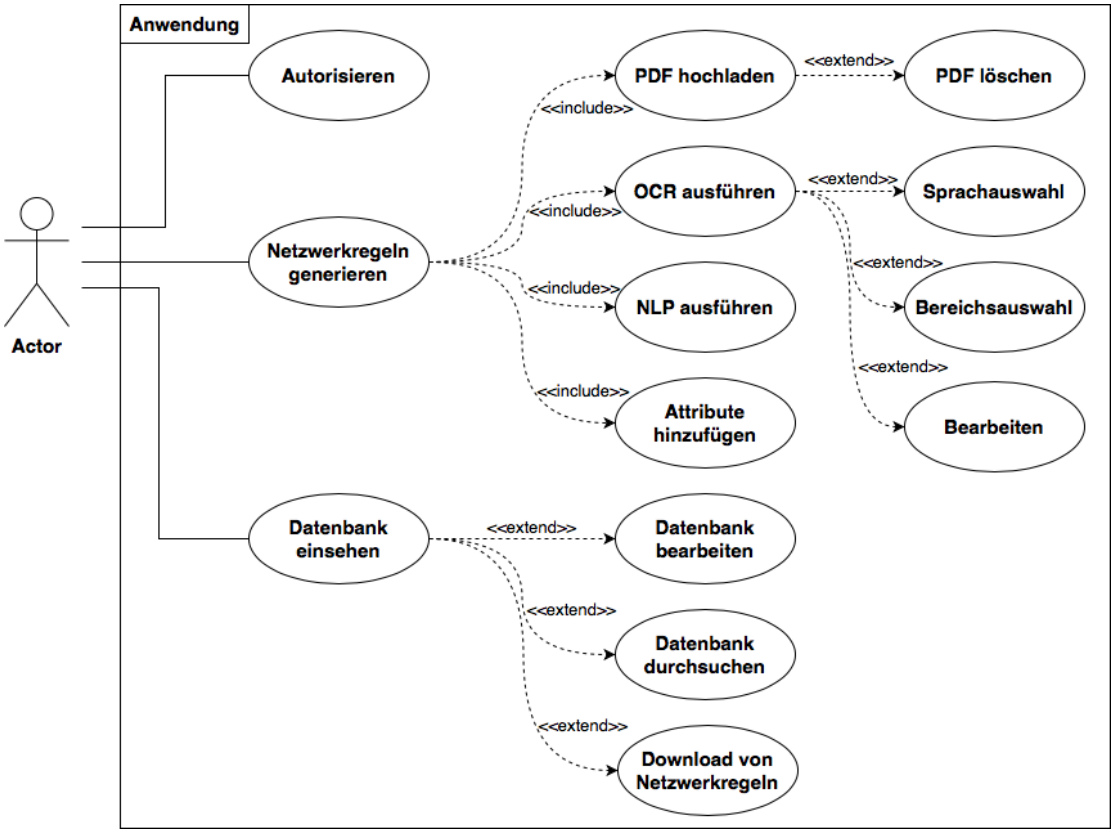

Abbildung 3.2: Ein Anwendungsfalldiagramm, das die Interaktion zwischen Nutzer und Anwendung zeigt. Die Ellipsen stellen die Anwendungsfälle dar, während die Pfeile die Beziehungen untereinander zeigen, die durch „extend“ oder „include“ beschrieben werden.

3.2 Funktionale Anforderungen

In diesem Abschnitt werden die Anwendungsfälle aus Abbildung 3.2 zu funktionalen Anforderungen überführt, die der Prototyp nach der Implementierung erfüllen soll. Zur späteren Zuordnung werden diese mit einer ID versehen. Der Titel soll die Anforderung prägnant wiedergeben, welche in der Beschreibung nochmals genauer ausgeführt wird.

Dabei ergeben sich folgende funktionalen Anforderungen:

ID	FA01
Titel	Autorisierung
Beschreibung	Der Nutzer kann sich im System autorisieren, um Zugang zu sensiblen Daten zu erhalten und die Datenbank zu bearbeiten.

ID	FA02
Titel	Hochladen von PDF-Dateien
Beschreibung	Dem Nutzer soll es möglich sein, eine oder mehrere Dateien im PDF-Format aus dem lokalen Speicher hochzuladen.

ID	FA03
Titel	Löschen von PDF-Dateien
Beschreibung	Der Nutzer soll die Möglichkeit haben, hochgeladene PDF-Dateien wieder zu entfernen.

ID	FA04
Titel	Sprachauswahl
Beschreibung	Der Nutzer soll die Möglichkeit haben, die Sprache der hochgeladenen Dateien auszuwählen, um die Ergebnisse des OCR-Vorgangs zu optimieren. Als Standard soll Englisch festgelegt sein.

ID	FA05
Titel	Bereichsauswahl
Beschreibung	Der Nutzer soll nach Hochladen der Dateien den Bereich in Seitenzahlen festlegen können, auf dem OCR und NLP ausgeführt werden. Als Standard soll die komplette Datei ausgewählt sein.

ID	FA06
Titel	Optische Texterkennung (OCR)
Beschreibung	Die hochgeladenen Dateien soll nach einer Bestätigung automatisch die Texterkennung durchlaufen.

ID	FA07
Titel	Bearbeiten der Texterkennung
Beschreibung	Der Nutzer hat die Möglichkeit, die Ergebnisse des OCR-Vorgangs einzusehen und zu bearbeiten.

ID	FA08
Titel	Natural Language Processing (NLP)
Beschreibung	Die NLP soll den erkannten Text nach Bestätigung durchlaufen und entsprechend untersuchen. Sind Netzwerkregeln vorhanden, werden diese anschließend aus den gegebenen Informationen generiert.

ID	FA09
Titel	Datenbank
Beschreibung	Vom Nutzer erstellte und im Prozess generierte Daten sollen dauerhaft in einer Datenbank gesichert werden.

ID	FA10
Titel	Festlegen von Attributen
Beschreibung	Mit den Ergebnissen sollen auch weitere Attribute festgelegt werden können, um die generierten Netzwerkregeln zuzuordnen: Titel, Autoren, Literaturtyp, Erscheinungsjahr, Nutzer, Erstellungsdatum, Bemerkung.

ID	FA11
Titel	Bearbeiten der Datenbankeinträge
Beschreibung	Der Nutzer soll die Möglichkeit haben, die Datenbank zu bearbeiten.

ID	FA12
Titel	Durchsuchen der Datenbank
Beschreibung	Die Datenbank soll nach den verschiedenen Attributen gefiltert werden können.

ID	FA13
Titel	Exportieren von Netzwerkregeln
Beschreibung	Der Nutzer soll die Möglichkeit haben die generierten Netzwerkregeln als Datei im CSV-Format herunterzuladen.

3.3 Nicht-Funktionale Anforderungen

In diesem Abschnitt werden die Qualitätsanforderungen an den Prototyp spezifiziert. Eine nicht-funktionale Anforderung beschreibt eine Charakteristik, die eine Software vorweisen bzw. eine Rahmenbedingung, die das System einhalten muss [14]. Damit wird die nötige Qualität des zu implementierenden Systems sichergestellt, da sich Entscheidungen in der Konzeption und Implementierung an diesen Aspekten orientieren.

Die Anforderungen werden jeweils wieder mit einer ID, einem Titel und einer Beschreibung aufgelistet. Aus der Analyse ergaben sich folgende Qualitätsanforderungen, welche der Prototyp nach der Implementierung aufweisen sollte:

ID	QA01
Titel	Nutzerfreundlichkeit (Usability)
Beschreibung	Das System muss dem Nutzer eine reibungslose Verwendung ermöglichen, um sein Vorhaben effizient und frustfrei durchzuführen.

ID	QA02
Titel	Intuitive Bedienung
Beschreibung	Die Interaktions- und Navigationskonzepte sollen anhand einer strukturierten Nutzeroberfläche intuitiv bedienbar und übersichtlich sein.

ID	QA03
Titel	Zuverlässigkeit
Beschreibung	Die Anwendung soll bei Verwendung zuverlässig und fehlerfrei arbeiten, um Systemabstürze zu verhindern.

ID	QA04
Titel	Effizienz
Beschreibung	Die Anwendung soll ein angemessenes Verhältnis zwischen der Dauer für die Funktionserfüllung und der Leistung des Systems bereitstellen.

4 Konzeption und Implementierung

Dieses Kapitel befasst sich mit der Konzeption des Prototyps und der Realisierung des Prozesses zur Regelgenerierung. Zunächst wird der Prototyp also anhand eines Diagramms und Mockups konzipiert, basierend auf den festgelegten Anforderungen des letzten Kapitels. Anschließend wird auf die Implementierung eingegangen, wobei die Realisierung der wichtigsten Komponenten aufgezeigt wird, sowie Screenshots dieser Bereiche.

4.1 Konzeption

In diesem Abschnitt werden Konzepte für den Prototyps entwickelt, auf dem die darauffolgende Implementierung basieren soll. Dazu eignet sich die Erstellung von Diagrammen und Mockups, um anhand von Konzepten einen geeigneten Ablauf und ein nutzerfokussiertes User Interface (UI) zu kreieren. Diese wurden mit Hilfe der Webseite „diagrams.net“ [17] erstellt.

4.1.1 Ablauf

Zuerst soll der Ablauf der Regelgenerierung konzipiert werden, um einen geeigneten Bedienungsprozess aufzustellen. Dazu wird ein Zustandsdiagramm modelliert, das mit seinen wenigen aber klar definierten Elementen das Verhalten eines Systems veranschaulichen kann.

Zunächst werden also die benötigten Elemente bestimmt. Die Anwendung kann sich stets nur in einem einzelnen Zustand befinden und diesen erst durch ein bestimmtes Ereignis verlassen. Die funktionalen Anforderungen aus Kapitel 3 bilden

dabei die benötigten Zustände und Ereignisse. Diese lassen sich wie folgt unterteilen: Aus den Anforderungen FA01, FA02, FA06, FA07, FA08, FA09 und FA10 lassen sich geeignete Zustände bestimmen, während die restlichen Anforderungen FA03, FA04, FA05, FA11, FA12 und FA13 der Ereignis-Kategorie zugeordnet werden. Da die Regelgenerierung auf verschiedenen Komponenten aufbaut, die in einer bestimmten Reihenfolge stattfinden müssen, konnte das Konzept in Abbildung 3.1 übernommen und erweitert werden.

Abbildung 4.1 zeigt das daraus resultierende Zustandsdiagramm. Der Zustand zur `Autorisierung` muss als erstes erfolgen, um Zugang zum System zu erhalten. Danach beginnt der eigentliche Prozess zur Regelgenerierung, begonnen mit dem `Hochladen von PDF-Dateien`. Nach Festlegen der Konfigurationen zum Bereich und der Sprache, führt das Ereignis `OCR starten` in den Zustand der OCR Ergebnisansicht. Über einen zusätzlichen Zustand `Bearbeiten` soll die Möglichkeit zum Einsehen und Editieren gegeben sein. Über das Ereignis `NLP starten` gelangt man zum letzten Teil des Prozesses zur Regelgenerierung. Die Ansicht der generierten Netzwerkregeln wird als Zustand `NLP Ergebnis` bezeichnet. Das Hinzufügen der Attribute erfolgt über einen eigenen Zustand, der nach Beendigung zur Regelansicht zurückführt. Mit dem Ereignis `Speichern` wird ein passender Datenbankeintrag angelegt, der im Zustand `Datenbank` eingesehen und bearbeitet werden kann. Der Prozess ist somit beendet und kann beliebig oft wiederholt werden, indem man zum Zustand `Hochladen von PDF-Dateien` zurückkehrt.

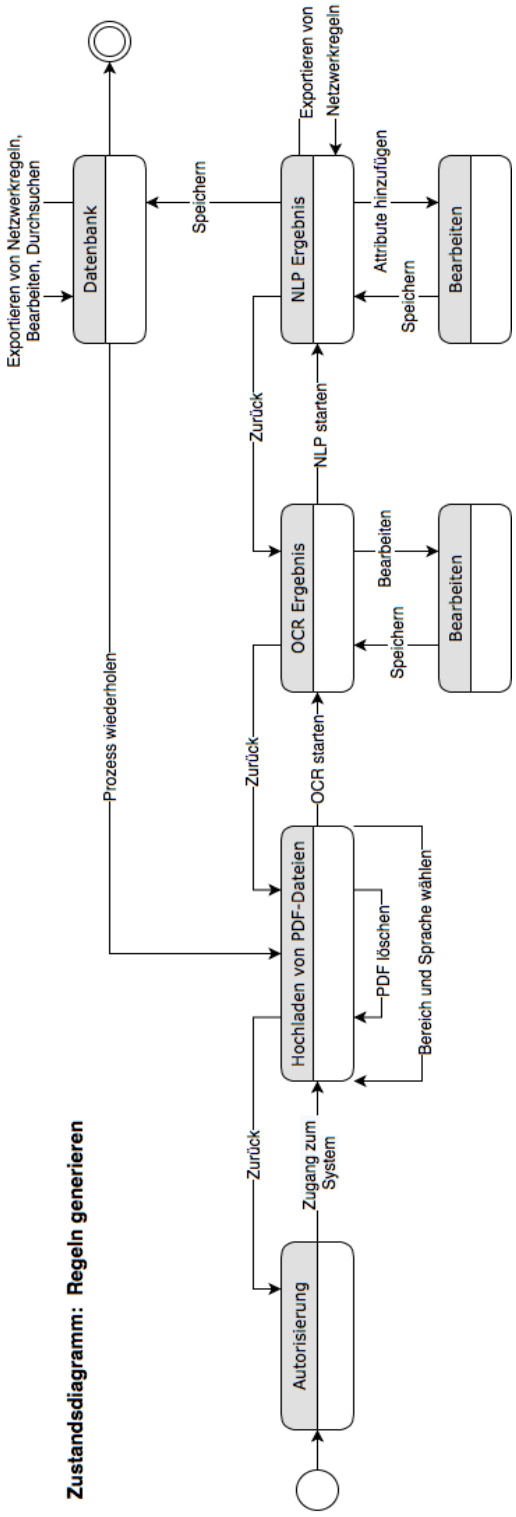

Abbildung 4.1: Ein Zustandsdiagramm zum Ablauf der Regelgenerierung, um den Bedienungsprozess zu konzipieren. Die Zustände und Ereignisse basieren auf den in Kapitel 3 aufgestellten Anforderungen.

4.1.2 Mockups

Um den implementierten Prozess anwenden zu können, wird ein geeignetes User Interface (UI) benötigt. Dazu sollen in diesem Abschnitt Entwürfe in Form von Mockups aufgestellt werden, basierend auf nutzerfokussierten Konzepten. Mockups geben einen ersten Überblick über die Nutzeroberfläche bzw. das User Interface (UI) und helfen dabei Aspekte wie User Experience (UX) umzusetzen, sowie die aufgestellten Qualitätsanforderungen QA01 und QA02. Dazu wird der konzipierte Ablauf des letzten Abschnitts übernommen.

Bei User Experience (UX) geht es darum, dem Nutzer eine zufriedenstellende Interaktion mit dem System zu ermöglichen. Dies kann durch verschiedene Kriterien [23] ermöglicht werden, die im Folgenden erwähnt und angewendet werden:

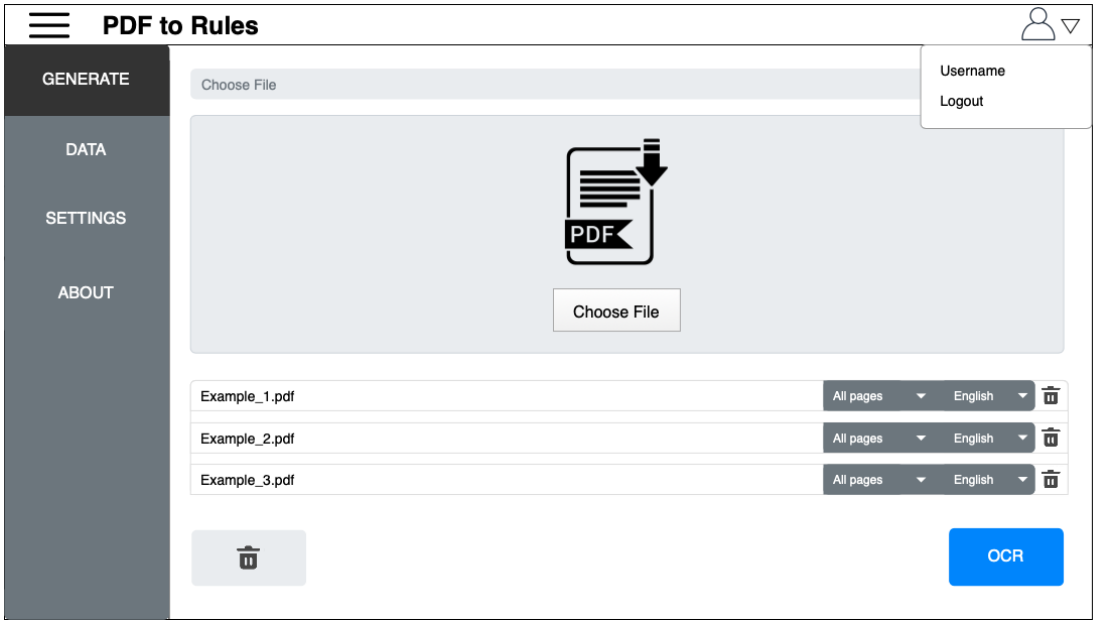

Abbildung 4.2: Mockup zum Hochladen von PDF-Dateien, in dem die gewählte Struktur den intuitiven Bedienungsprozess (QA02) unterstützen soll. In diesem Bereich kann der Nutzer die gewünschten PDF-Dateien hochladen, konfigurieren und löschen (FA02, FA03, FA04, FA05).

Der **Inhalt** der Anwendung muss so strukturiert sein, dass der Nutzer sich schnell zurecht findet und die **Navigationskonzepte** intuitiv bedienen kann. Dazu werden

geeignete UI-Elemente genutzt und entsprechend positioniert. In den Mockups wurde dies umgesetzt, indem eine seitliche Navigationsleiste platziert wurde, um einen Bereichswechsel schnell und übersichtlich zu ermöglichen. Damit bleibt der restliche Bereich für die eigentliche Verwendung. Die Elemente des Hauptbereichs sind stets untereinander angeordnet und unterstützen durch diesen Verlauf auch den Anwendungsprozess, der von oben nach unten erfolgt. Dadurch soll die **intuitive Bedienung** (QA02) unterstützt werden. Da auch **Konsistenz** ein wichtiges Prinzip für eine gute UI darstellt, wird diese Grundstruktur in jeder Ansicht beibehalten.

Abbildung 4.2 zeigt das Mockup, welches für das Hochladen von PDF-Dateien erstellt wurde. Eine **Navigationshilfe**, hier z.B. eine Pfadangabe, und eine Hauptkomponente sollen dem Nutzer vermitteln, in welchem Bereich er sich befindet und welche Aktivität erwartet oder ausgeführt wird. Für die hochgeladenen PDFs eignet sich eine übersichtliche Listenansicht. Durch farblich hervorgehobene Buttons können **visuelle Hinweise** gegeben werden. Dies wird hier genutzt, um den Nutzer durch den Hauptprozess, also das Generieren von Netzwerkregeln, zu führen.

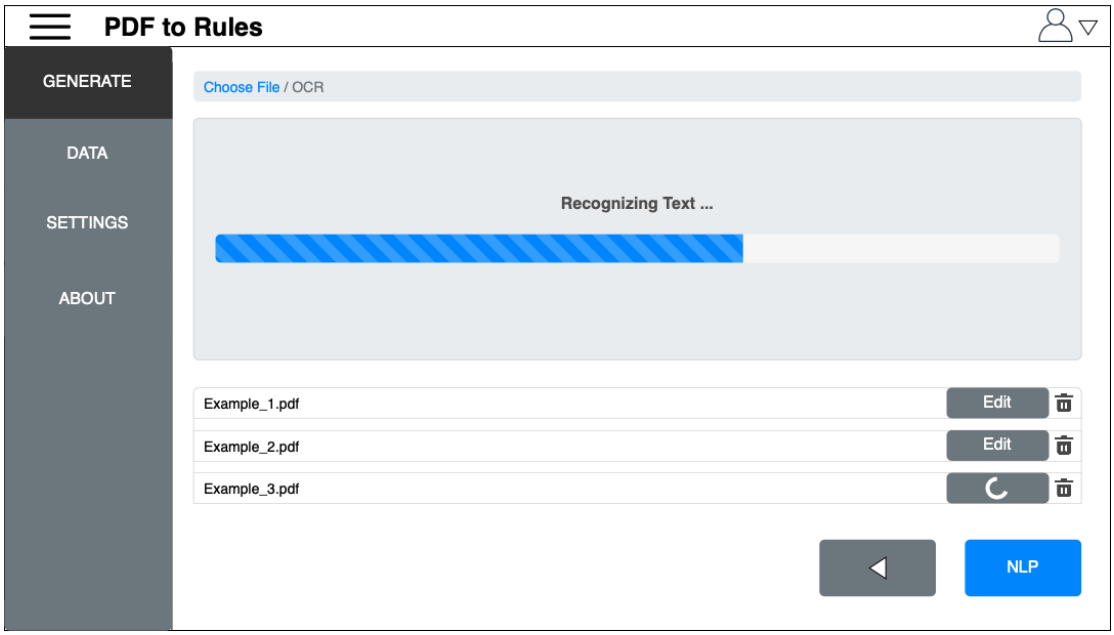

Abbildung 4.3: Mockup zum OCR-Vorgang. Der Prozess zur Texterkennung (FA06) startet in dieser Ansicht und lässt sich visuell verfolgen, um User Feedback zu vermitteln. Auch das Bearbeiten des erkannten Textes (FA07) ist anschließend möglich.

Zudem soll dem Nutzer über geeignetes visuelles **User Feedback** veranschaulicht werden, inwieweit das System auf seine Eingaben und Aktionen reagiert. Dies kann beispielsweise mit Animationen und Alerts erfolgen. Dies wurde in Abbildung 4.3, dem Mockup zum OCR-Vorgang, mit einer Processbar verwirklicht, die den Fortschritt des Vorgangs wiedergeben soll. Auch in der Listenansicht können Elemente wie ein Spinner dem Nutzer vermitteln, ob die Texterkennung noch im Gange ist. Die Möglichkeit zur vorherigen Ansicht zurückzukehren, sowie das Löschen einzelner PDFs soll die **Nutzerfreundlichkeit (Usability)** (QA01) zusätzlich unterstützen und eine reibungslose Nutzung ermöglichen.

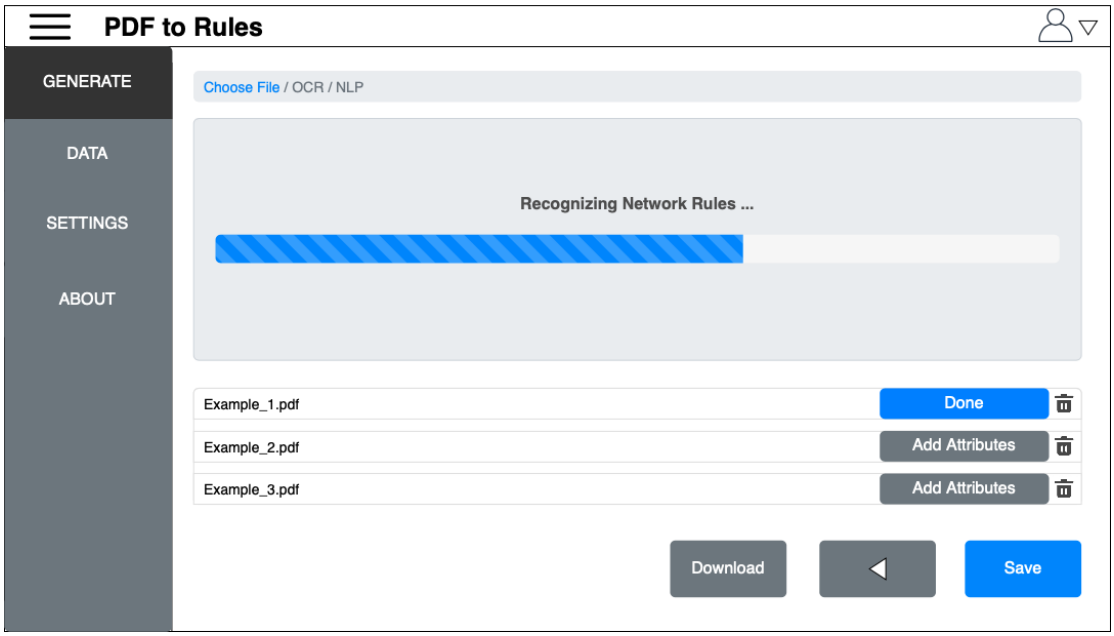

Abbildung 4.4: Mockup zum NLP-Vorgang. Der Prozess zur Sprachverarbeitung (FA08) startet in dieser Ansicht und lässt sich ebenfalls visuell verfolgen. Nach dem Festlegen von Datenbank Attributen (FA10) ist der Prozess zur Regelgenerierung beendet, was durch visuelles Feedback vermittelt wird.

Auch im Mockup zum NLP-Vorgang, zu sehen in Abbildung 4.4, wurden dieselben Konzepte umgesetzt. Auch hier werden mit Farben Komponenten hervorgehoben, die den Nutzer zum Ende des Prozesses zur Regelgenerierung führen. Nach Hinzufügen der Attribute, die zum Auffinden der Datenbankeinträge dienen, wird dem Nutzer visuell vermittelt, dass der Vorgang für dieses PDF beendet ist.

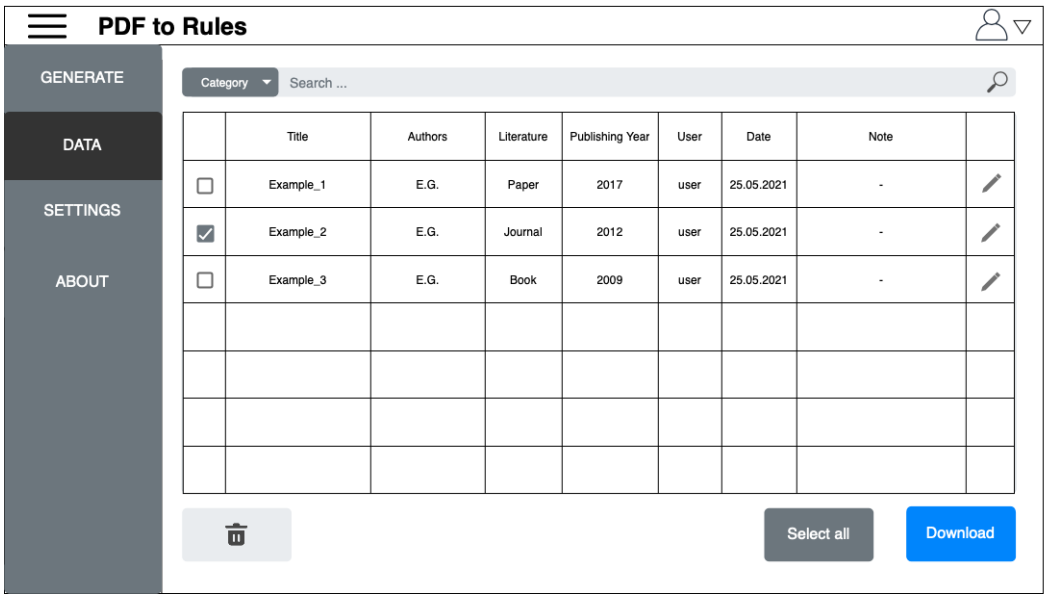

Abbildung 4.5: Mockup zur Einsicht in die Datenbank (FA09). Die Hauptkomponente bildet eine Tabelle, die durch wenige Funktionen (FA11, FA12, FA13) in ihrer Nutzerfreundlichkeit (QA01) unterstützt wird. Das schlichte Design soll den Cognitive Load verringern und positiv zur UX beitragen.

Um den Nutzer eine übersichtliche Einsicht in die Datenbank zu ermöglichen, wurde in Abbildung 4.5, dem Mockup zur Datenbankeinsicht, eine Tabelle erstellt, welche auch die Hauptkomponente der Ansicht bildet und nur durch wenige Buttons unterstützt wird. Damit soll der **Cognitive Load** reduziert werden. Dies ist der mentale Aufwand, den es benötigt, um eine Aufgabe zu erledigen. Besonders bei umfangreichen Datenbanken sollte eine UI schlicht bleiben, damit der Nutzer eine angenehme UX erlebt. Auch Funktionen wie das Durchsuchen der Datenbank oder das Selektieren aller Einträge stärken die Usability, da die Anzahl der nötigen **Nutzeraktionen** verringert wird. Auch für die restlichen Bereiche und Ansichten werden dieselben Konzepte verwendet, weshalb nur die Mockups der wichtigsten Zustände vorgestellt wurden.

4.2 Implementierung

In diesem Abschnitt wird die Realisierung des Prozesses zur Regelgenerierung aufgezeigt, für den mehrere Teilprozessschritte notwendig sind. Diese werden durch Unterabschnitte unterteilt. Auch das realisierte User Interface wird anhand von Screenshots gezeigt. Die genutzten Konzepte zur Erstellung der Mockups wurden übernommen, mit Änderungen, die sich durch die Implementierung ergaben.

4.2.1 Inputverarbeitung

Zunächst soll die Verarbeitung des User Inputs, sowie das Vorbereiten der PDFs aufgezeigt werden. Dazu wurde in Abbildung 4.6 ein Flussdiagramm erstellt, welches das Vorgehen dieses ersten Teilprozessschritts zeigt.

Die Regelgenerierung beginnt mit dem Hochladen einer gewünschten PDF-Datei, die eine wissenschaftliche Arbeit oder ähnliches sein kann. Die Voraussetzung für das Generieren von Netzwerkregeln ist jedoch, dass genregulatorische Aussagen in Form von natürlicher Sprachbeschreibung enthalten sind. Der File Upload erfolgt dabei durch einen standardmäßigen File Explorer. Im Gegensatz zur Konzeption werden auch die Datenbank Attribute bereits hier vom Nutzer als User Input entgegengenommen. Als Voraussetzung wurde festgelegt, dass mindestens ein Titel festgelegt, sowie eine PDF-Datei hochgeladen werden muss, bevor der Ablauf fortgeführt wird. Alle anderen Attribute gelten dabei als optional. Sind diese Bedingungen erfüllt, wird der Datenbankeintrag angelegt. Aus der Implementierungssicht erleichtert dies den späteren Zugriff. Anschließend folgt das Vorbereiten der PDF-Dateien auf den Prozess zur Texterkennung. Dazu wird zunächst der Seitenbereich gewählt, auf dem der restliche Prozess stattfinden soll. Entsprechend dieser Einstellung, werden die Seiten der PDF darauffolgend in JPEGs konvertiert. Dies ist notwendig, da die OCR-Engine nur Bildformate entgegennehmen kann. Die daraus resultierenden Bilder werden nach Ausführung des OCR-Vorgangs nicht mehr benötigt und gelöscht, um den Speicherbedarf zu minimieren. Damit endet das Verarbeiten des User Inputs und der Prozess zur Texterkennung kann folgend ausgeführt werden.

Abbildung 4.7 zeigt den Screenshot des Prototyp in der **Upload File** Ansicht.

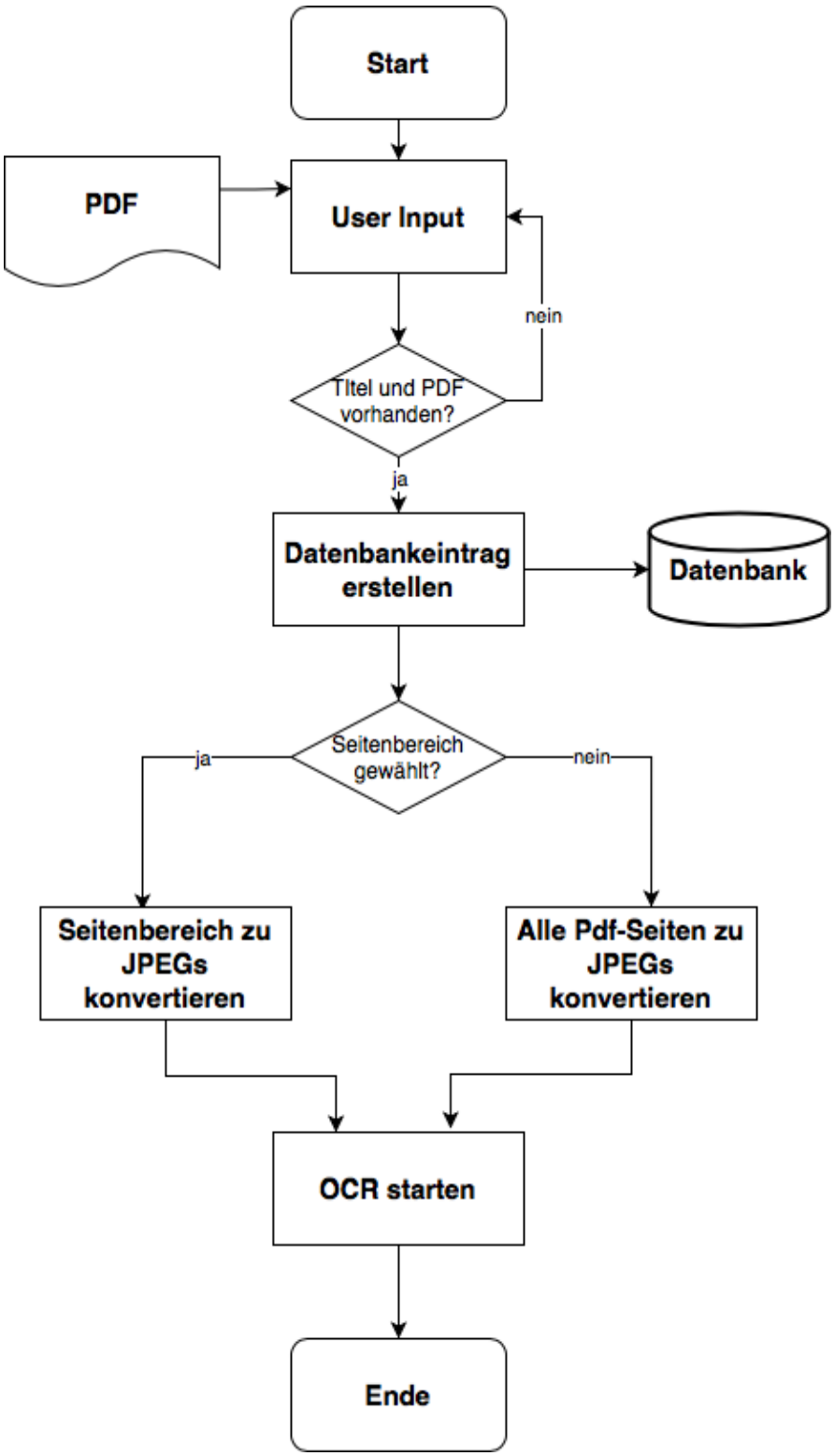

Abbildung 4.6: Ein Flussdiagramm, das die Verarbeitung des User Inputs zeigt, was den ersten Teilprozessschritt bildet, auf den anschließend der OCR-Vorgang folgt.

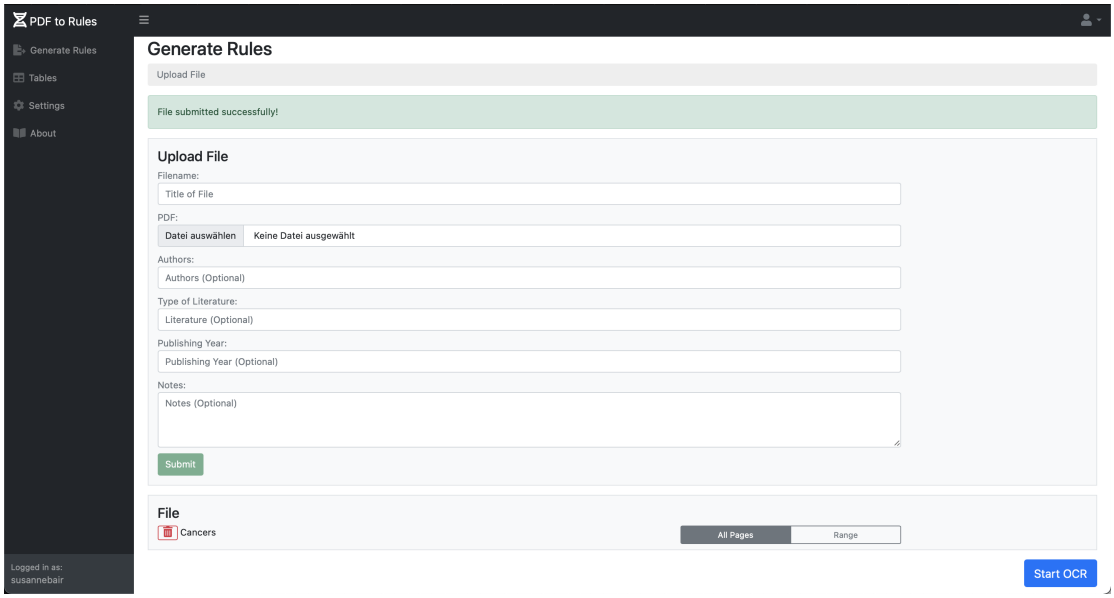

Abbildung 4.7: Ein Screenshot des implementierten Prototyps nach Hochladen einer PDF-Datei, das mittels standardmäßigem File Explorer geschieht. Zusammen mit den festgelegten Attributen wird nach erfolgreichem Upload ein Datenbankeintrag angelegt.

4.2.2 OCR

In diesem Abschnitt wird aufgezeigt, wie der Vorgang zur Texterkennung erfolgt, um den enthaltenen Text der PDF zu extrahieren. Das Ziel ist hierbei, möglichst korrekte Ergebnisse zu erhalten, um daraus wiederum korrekte Netzwerkregeln generieren zu können. Um dies zu erreichen, müssen sowohl die Voraussetzungen für die Bilder als auch die Konfigurationen der Engine passend festgelegt werden. Diese beiden Bedingungen werden im Folgenden festgelegt.

Voraussetzungen für Bilder

Die erste Bedingung, um möglichst korrekte Ergebnisse zu erhalten, ist die Qualität der überreichten Bilder. Auswertungen verschiedener OCR-Engines [21, 5, 22] haben gezeigt, das folgende Faktoren besonders wichtig sind, um die Ergebnisse zu optimieren:

- Auflösung
- Textgröße
- Farbpalette
- Hintergrundfarbe

Außerdem wurde gezeigt [21], dass die Genauigkeit der Ergebnisse stark abfiel, wenn die Bildauflösung von 300 dpi auf 200 dpi herabgesetzt wurde. Die Erhöhung auf 400 dpi führte dabei zu keinen oder minimalen Verbesserungen. Anhand dieser Ergebnisse wird festgelegt, dass die Bilder eine minimale Auflösung von 300 dpi haben sollten, um für eine ausreichende Qualität zu sorgen. Für den Prototyp wurde daher eine Auflösung von 400 dpi gewählt. Außerdem konnte festgestellt werden, dass eine Farbauflösung in Graustufen zu weniger Erkennungsfehlern führte, da die OCR-Engine den Text besser vom Hintergrund separieren kann [21, 5], weshalb auch diese Änderung für die Bilder vorgenommen wird.

Da im Kontext dieser Arbeit nur wissenschaftliche Publikationen oder vergleichbares verarbeitet werden, kann von einer standardmäßigem Hintergrundfarbe Weiß ausgegangen werden, weshalb dieser Faktor nicht verändert wird. Das gleiche gilt für die Textgröße, die bei den betrachteten PDFs ausreichend groß ist.

Konfiguration der Engine

Nach der Installation von Tesseract und PyTesseract kann es im eigenen Projekt importiert und angewendet werden. Die zu erkennende Sprache kann dabei angegeben werden, im Prototyp ist Deutsch und Englisch vom Nutzer einstellbar.

Dabei mussten noch weitere Konfigurationen festgelegt werden, um die Ergebnisse zu optimieren. Tesseract bietet verschiedene Modi, um die Engine auf den eigenen Fall anzupassen. Dabei wurden folgende Modi verwendet:

- **oem** legt den „OCR-Engine Mode“ fest, wobei die Optionen 0-3 wählbar sind. Mit diesem Modus wird die genutzte Engine festgelegt, wobei die gewählte Option 3 für den Modus „Legacy + LSTM engines“ steht. Seit Tesseract 4 wurde eine Engine hinzugefügt, die „Long Short-Term Memory“ (LSTM) Neural Network verwendet und dadurch die Genauigkeit der Ergebnisse verbessert [29].

- **psm** beinhaltet die Optionen 0-13 und beeinflusst die Seitensegmentierung. Die gewählte Option 1 steht für „Automatic page segmentation with OSD“, mit der Eigenschaften wie die Textausrichtung und das Layout analysiert werden. Da die PDFs keine einheitlichen Textlayouts haben, ist eine automatische Erkennung sinnvoll. Dies führt jedoch zu Performanceeinbußen [29].
- **c** steht für „configVar“. Mit diesem Befehl können Default Werte in der `config` verändert werden. Das Setzen von 0 (false) für die Variablen `load_punc_dawg`, `load_system_dawg` und `load_freq_dawg` deaktiviert das Laden von Wörterbüchern im OCR-Prozess. Da Gene oft aus Buchstaben und Zahlen bestehen und nicht in Wörterbüchern vorkommen, stellt Tesseract die enthaltenen Zahlen oft als Buchstaben dar, was zu falschen oder nicht erkannten Ergebnissen führt. Dieser Fehler soll durch Deaktivieren der Wörterbücher vermindert werden.

Mit Berücksichtigung dieser Bedingungen kann der OCR-Vorgang mit den konvertierten und angepassten Bildern angewendet werden. Das Resultat wird anschließend in der Datenbank gespeichert und dem Nutzer wie im Screenshot 4.8 des Prototyps zurückgegeben.

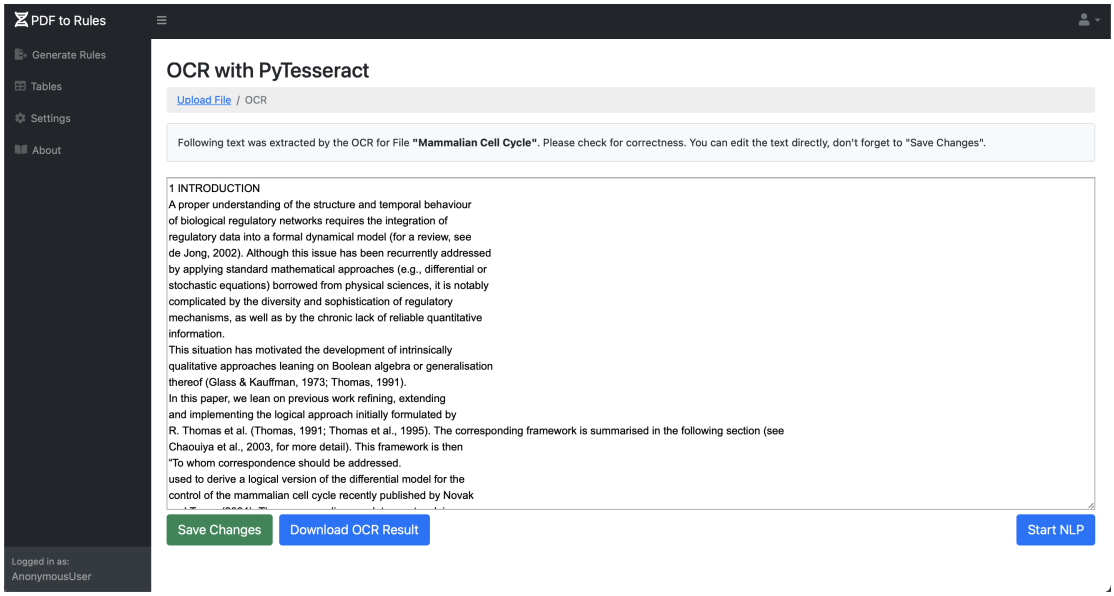

Abbildung 4.8: Ein Screenshot des Prototyps nach Extrahieren des Textes mit der OCR-Engine Tesseract, die mit verschiedenen Modi auf den Kontext dieser Arbeit konfiguriert wurde. Der Text stammt aus der Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8].

4.2.3 INDRA

Nach Betätigen der Schaltfläche **Start NLP** in Abbildung 4.8 beginnt der vielschichtige Prozess zur Regelextraktion mit INDRA. Das Ergebnis der OCR-Engine wird mit Hilfe von INDRA-Modulen und externen Werkzeugen von Text in natürlicher Sprache zu einem bool'schen Netzwerk umgewandelt.

Dazu sind im Grunde drei Schritte notwendig [4]: Durch die natürliche Sprachverarbeitung (NLP) wird der Text auf genregulatorische Aussagen untersucht, indem den Einheiten sowie deren Mechanismen eindeutige Bezeichner zugewiesen werden, die auf verknüpften Datenbanken beruhen. Um diese Informationen in ein vielseitig nutzbares Format zu überführen, werden im nächsten Schritt sogenannte INDRA-Statements erstellt. Im dritten und letzten Schritt werden diese INDRA-Statements mit Hilfe von integrierten Assemblern in ausführbare Modelle oder Netzwerke transformiert.

Dieser komplexe Vorgang wurde in Abbildung 4.13 veranschaulicht, angelehnt an „Figure 1“ aus Quelle [4], jedoch angepasst an den Kontext dieser Arbeit. Dort werden die drei Schichten der INDRA-Architektur gezeigt, die den Informationsfluss zwischen Nutzer, INDRA und den externen Tools veranschaulicht [4]. Im Folgenden werden die auf der Abbildung vermerkten Schritte 1-4 dieser Architektur genauer erklärt. In der Spalte Resultat können die Ergebnisse der Teilschritte anhand eines Beispielsatzes verfolgt werden.

NLP

Zunächst muss also die „Natural Language Processing“ (NLP) stattfinden, der wichtigste Schritt nach Ausführen der OCR. Erst durch diese Verarbeitung können die benötigten Aussagen vom restlichen Text unterschieden werden.

INDRA stellt verschiedene Module zur Sprachverarbeitung bereit, darunter das hierfür gewählte REACH Reading System, das auf Biomedizin ausgelegt ist und zurzeit die meisten Nutzungsmöglichkeiten bietet. Dabei hat sich das lokale Aufsetzen des Servers für den Kontext dieser Arbeit als Effektivste herausgestellt. Dies liegt mitunter daran, dass das Verarbeiten ganzer wissenschaftlicher Arbeiten eine zu große Menge an Text produziert um von einer externen API verarbeitet werden zu können. Der lokale Server konnte auf eine maximale Zeichenanzahl von 256000 Zeichen konfiguriert werden und ist somit fähig, PDFs mit großem Umfang zu verarbeiten.

In **Schritt 1** der Abbildung 4.13 wird der extrahierte Text als natürliche Sprachbeschreibung der REACH-Schnittstelle übergeben. Zur Veranschaulichung an einem Beispiel wurde hierfür der Text *„CycA inactivates Cdh1.“* gewählt, der im Laufe des Prozesses in eine Bool’sche Funktion umgewandelt wird.

NLP erfolgt dabei in zwei Teilen [4]: Bei der sogenannten „Named entity recognition“ (NER) werden spezielle Wörter erkannt. Im Bezug auf Biomedizin sind dies Gene, Proteine und andere zugehörige Begriffe. Auf diese Weise können relevante Aussagen vom restlichen Text unterschieden werden. Beim anschließenden „Grounding“ werden den Begriffen eindeutige Bezeichner zugewiesen, die auf verknüpften Datenbanken beruhen. Das Resultat dieser Vorgänge ist eine von REACH erstellte JSON-Datei, die alle gewonnenen Informationen speichert. Ein Screenshot in der Spalte Resultat der Abbildung 4.13 zeigt einen Auszug dieser JSON, bezogen auf

das gewählte Beispiel.

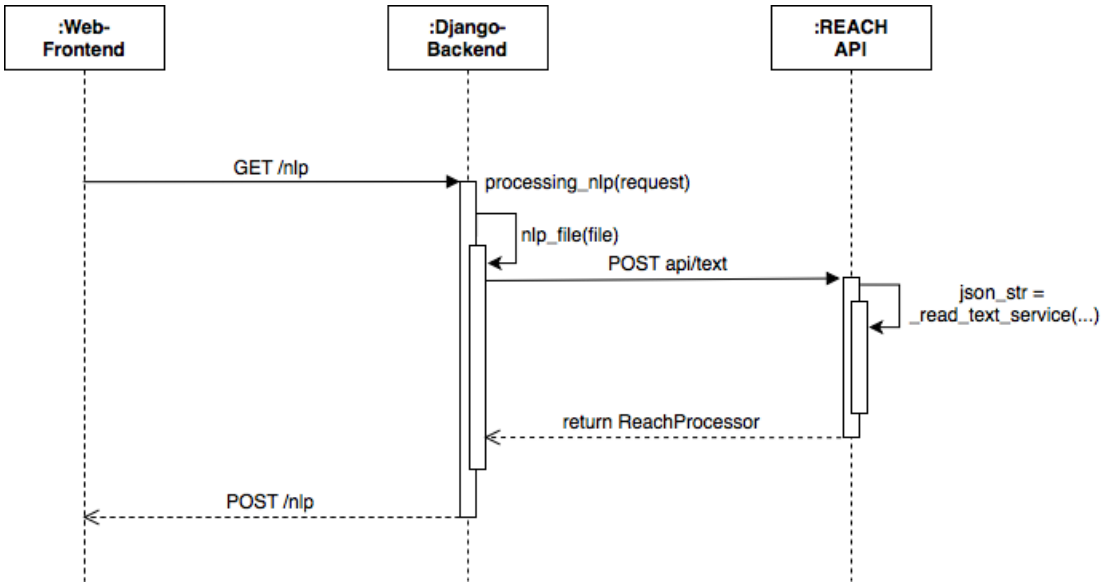

Abbildung 4.9: Der NLP-Prozess, der innerhalb der INDRA-Architektur stattfindet, veranschaulicht anhand eines Sequenzdiagramms, dass die Kommunikation zwischen dem Web-Frontend, dem Django-Backend und der REACH API zeigt.

Der genaue Ablauf innerhalb des implementieren Prototyps wird dabei im Sequenzdiagramm 4.9 veranschaulicht, das die Kommunikation zwischen Frontend und Backend der Anwendung mit der REACH API zeigt. Startet der Nutzer den NLP-Vorgang, wird eine GET-Anfrage an das Backend geschickt, in dem Methodenaufrufe stattfinden, welche die Kommunikation mit dem Server starten. Daraufhin wird der erkannte Text über eine POST-Methode an die REACH API übergeben, bei der Folgendes passiert: Der Text wird verarbeitet und ein JSON-String erstellt. Dieser wird wiederum in einen ReachProcessor überführt, der benötigt wird, um die gewonnenen Informationen in das INDRA-Format zu überführen, was in **Schritt 2** der Abbildung 4.13 geschieht. Dieser ReachProcessor enthält die INDRA-Statements, die im Backend extrahiert und für den weiteren Verlauf genutzt werden.

INDRA-Statements

INDRA-Statements sind allgemeine Wissensdarstellungen für mechanistische Informationen [4]. Sie setzen sich zusammen aus molekularen Einheiten (CycA, CDH1, ...) und deren Mechanismen (Inhibition, Activation, Phosphorylation, ...). Auch ein Beweis, genannt Evidence, liegt als Textauszug vor und kann als Beleg für das Statement eingesehen werden. Somit kann auch die herangezogene Textstelle sowie die Korrektheit der daraus resultierenden Netzwerkregeln zurückverfolgt werden. Für unser Beispiel in der Resultat Spalte in Abbildung 4.13 ergibt sich also das INDRA-Statement *Inhibition(cycA(), CDH1())* mit der beispielhaften Evidence „CycA inactivates Cdh1."

Assembler

Die INDRA-Statements können dann genutzt werden, um Modelle, Netzwerke oder andere Ausgabeformate zu konstruieren [4]. Dies geschieht in **Schritt 3** der Abbildung 4.13 mit Hilfe eines weiteren INDRA-Moduls, dem Assembler. Denn um die gewünschten Netzwerkregeln zu erhalten, muss ein Bool'sches Netzwerk erstellt werden. Dazu wird der SIF Assembler genutzt, der für Netzwerkanalysen, Logikmodellierung und Visualisierung eingesetzt wird [9]. Dieser nimmt die INDRA-Statements als Input entgegen und stellt einen Graphen im Netzwerk Format „NetworkX" zusammen. „NetworkX" ist ein Python Package, ausgelegt auf das Erstellen, Bearbeiten und Analysieren von Netzwerken.

Dieser Graph baut sich aus einem SIF String zusammen. SIF steht dabei für „Simple Interaction Format" und ist im Grunde eine Liste, bestehend aus drei Spalten: Teilnehmer A, einem Interaktionstyp und Teilnehmer B [18]. In unserem Fall bildet der Interaktionstyp stets eine 1 oder -1 ab. Im Beispiel ergibt sich somit *cycA -1 CDH1*. Dieses Format kann von Netzwerkvisualisierungssoftware und booleschen Netzwerksimulationstools interpretiert werden [4].

Eines dieser Tools ist das Package „Booleannet". Die Software wird eingesetzt, um genregulatorische Netzwerke im bool'schen Formalismus darzustellen [12]. Mit Hilfe des Tools wird aus dem Graphen ein Bool'sches Netzwerk konstruiert, aufgebaut aus den Anfangszuständen jeder Einheit, die nicht verwendet werden, und den generierten Netzwerkregeln. Für das Beispiel der Resultat Spalte ergibt sich

damit *cycA* = not (CDH1)*. In **Schritt 4** der Abbildung 4.13 werden abschließend alle Netzwerkregeln extrahiert und die logischen Operatoren mit „&“, „|“ und „!“ ersetzt. Für das Beispiel in Abbildung 4.13 ergibt sich die Bool'sche Funktion bzw. Netzwerkregel *cycA, ! (CDH1)*.

Im Screenshot 4.10 des Prototyps kann man sehen, dass die generierten Netzwerkregeln abschließend tabellarisch aufgelistet werden. Zur Kontrolle der Regeln wurden auch die INDRA-Statements mit ihrer Evidence als durchsuchbare Tabelle angehängt.

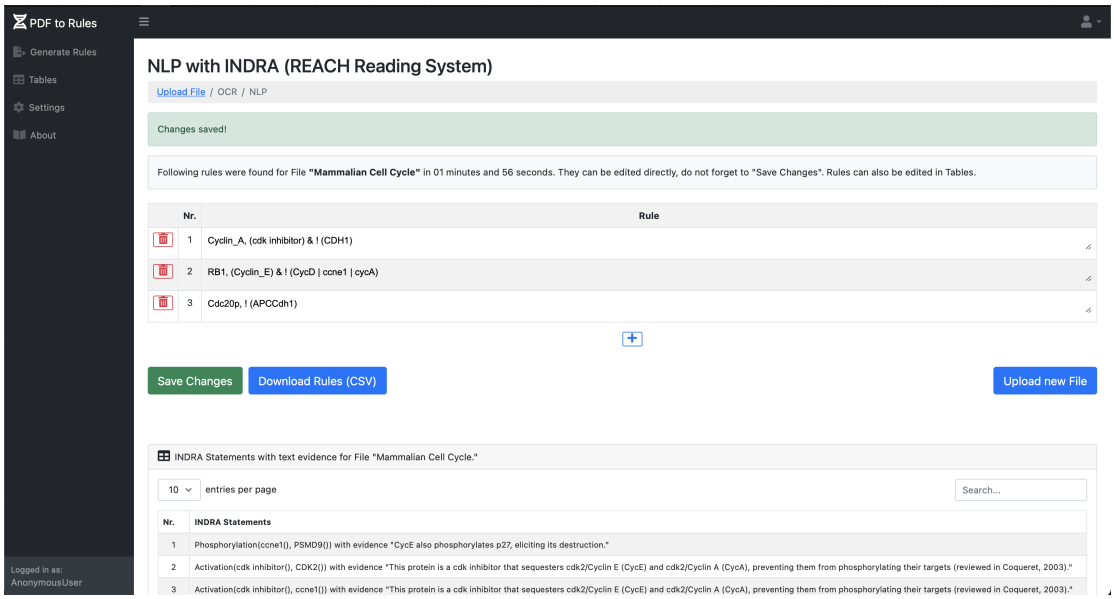

Abbildung 4.10: Screenshot des Prototyps nach Generieren der Netzwerkregeln mit NLP und INDRA. Zur Kontrolle werden die INDRA-Statements mit ihrer Evidence ebenfalls in einer durchsuchbaren Tabelle angezeigt.

4.2.4 Datenbank

Das Datenbankschema in Abbildung 4.11 wurde mit Hilfe von „pgAdmin 4“ generiert, ein Programm zur Entwicklung und Administration von PostgreSQL Datenbanken. Für den Prozess zur Regelgenerierung wurden zwei Datenbanktabellen erstellt und verwendet: In **generated_files** werden alle Daten gespeichert, die vom

Nutzer als Input entgegengenommen werden, sowie die Daten, die im Laufe des Prozesses erstellt werden, darunter der erkannte Text oder das Array der generierten Netzwerkregeln. In **generate_setting** wird lediglich der Defaultwert der OCR-Sprache gespeichert, der vom Nutzer im Bereich `Settings` festgelegt wird. Bei beiden Datenbanktabellen bildet die ID den eindeutigen Primärschlüssel.

Der Screenshot in Abbildung 4.12 zeigt den realisierten Prototyp im Bereich `Tables` und damit die den Inhalt der Datenbank, der sowohl hier als auch im Admin-Bereich bearbeitet werden kann.

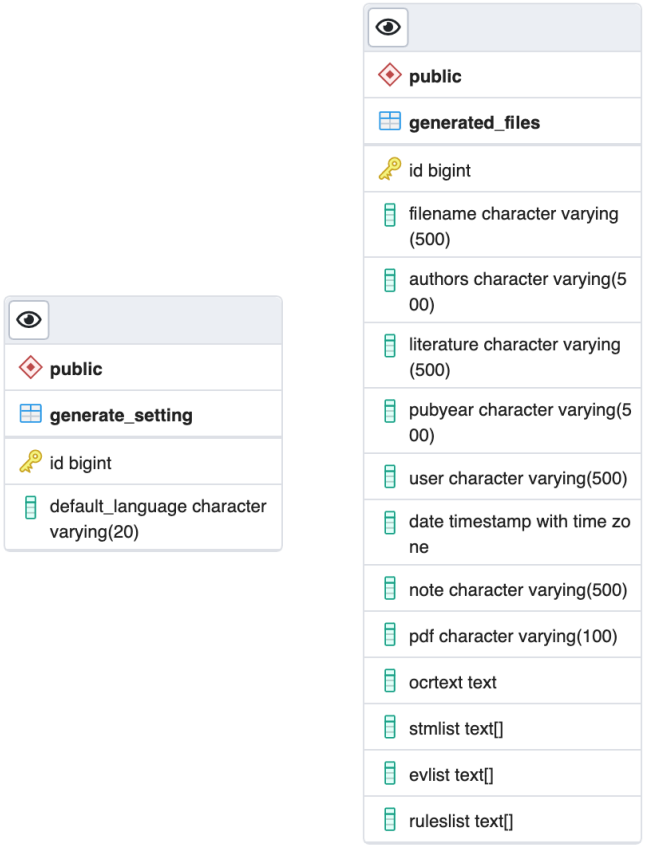

Abbildung 4.11: Das Datenbankschema, das die Datenbanktabellen zeigt, die für den Prozess zur Regelgenerierung erstellt und verwendet wurden. Das Schema wurde mit der Software „pgAdmin 4“ generiert.

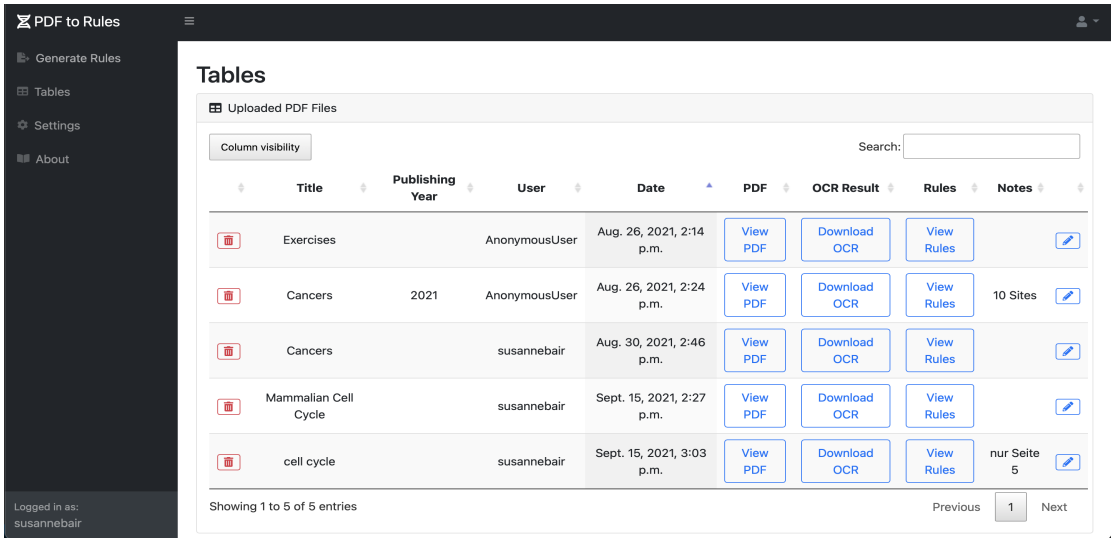

Abbildung 4.12: Screenshot des Prototyps im Tables Bereich, der den Inhalt der PostgreSQL Datenbank ausgibt und Möglichkeiten zur Suche und Bearbeitung bietet.

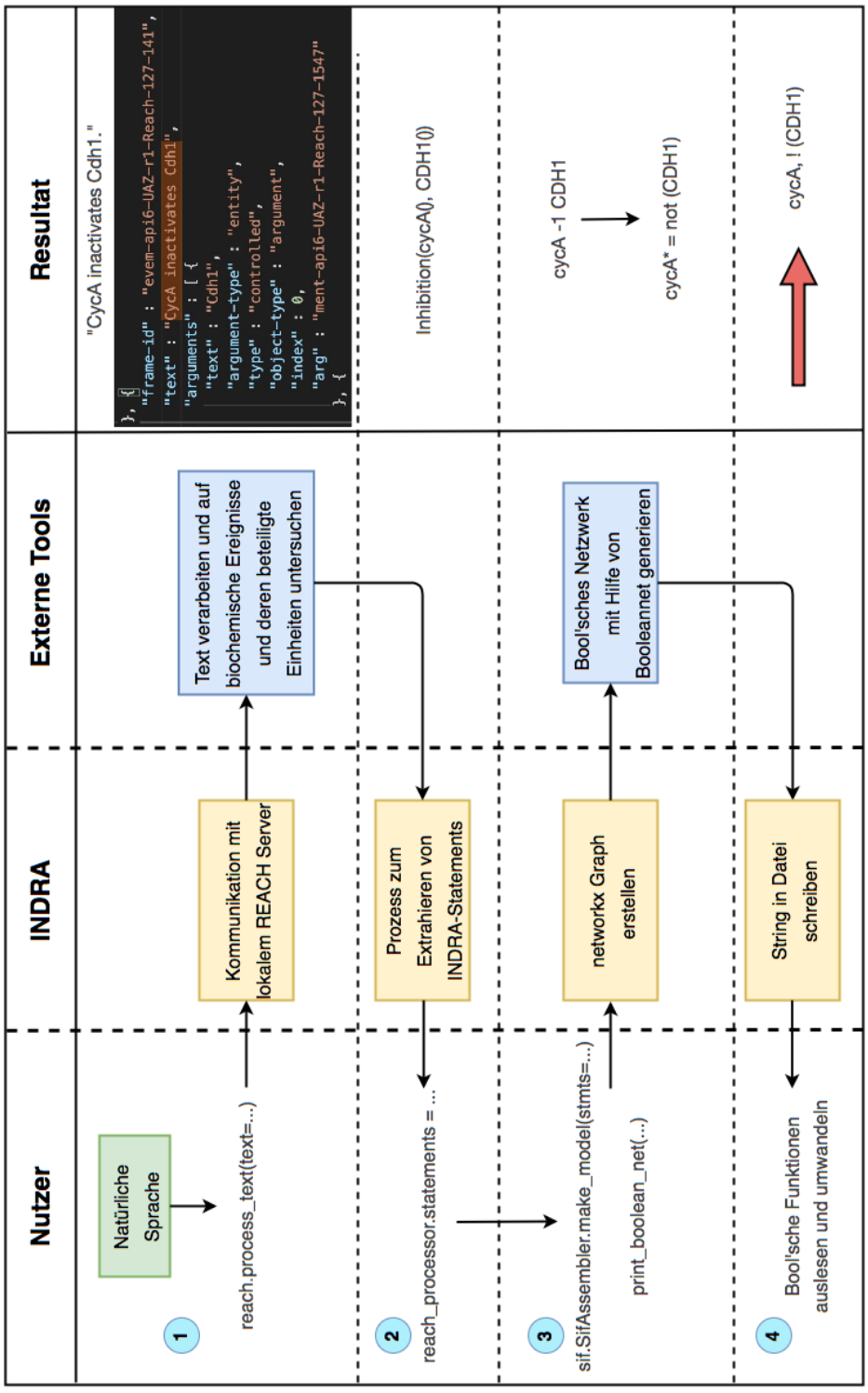

Abbildung 4.13: Die drei Schichten der INDRA-Architektur, veranschaulicht anhand einer Grafik und verfolgbar an einem Beispiel in der Spalte Resultat. Durch die Spalten Nutzer, INDRA und Externe Tools ist zu erkennen, wo welcher Schritt stattfindet.

5 Evaluation

In diesem Abschnitt werden die Ergebnisse des Prototyps analysiert und evaluiert. Genauere Messungen sollen einen Einblick in die Leistung des Systems geben und aufzeigen, inwieweit sich der implementierte Prozess für die Problemstellung eignet. Hierbei wird erwartet, dass sowohl die Texterkennung als auch die generierten Netzwerkregeln so korrekt und vollständig wie möglich sind.

Um Aussagen über den Prozess treffen zu können, werden die Teilbereiche OCR und NLP zusammen mit INDRA einzeln auf verschiedene Parameter untersucht, da die generierten Netzwerkregeln das Resultat der einzelnen Teilschritte sind. Um eine reale Anwendung zu simulieren, wurden hierfür die wissenschaftlichen Publikationen „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8], „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] und „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] als Testobjekte hergenommen, deren korrekte Netzwerkregeln innerhalb der jeweiligen Arbeit vorliegen und zum Vergleich dienen.

5.1 Auswertung der OCR

Zunächst werden die Ergebnisse des OCR-Vorgangs ausgewertet. Dabei muss vor allem geprüft werden, wie weit der erkannte dem korrekten Text gleicht und ob verschiedene Layouts korrekt erkannt werden. Anhand dessen kann die Genauigkeit des OCR-Vorgangs gemessen werden. Anschließend kann beobachtet werden, welche Fehler dabei auftreten und wie viele davon die Generierung von Netzwerkregeln betreffen. Die OCR-Engine wurde nicht auf eigene Testdaten trainiert und der Vorgang erfolgt ohne menschliches Einwirken. Die Konfiguration der Engine wurde in Kapitel 4 erläutert, als Sprache wurde Englisch festgelegt.

5.1.1 Prüfen des erkannten Textes

Zunächst wurde der Text der Publikation „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] vollständig verbessert, um eine korrekte Version des Textes zu erhalten und mit der OCR-Erkennung vergleichen zu können. Mehrfache Überprüfungen haben sichergestellt, dass die Korrektur fehlerfrei ist. Dabei wurde der Abschnitt „References“ weggelassen, da im Literaturverzeichnis keine Aussagen zu Netzwerkregeln auftreten und somit keine Ergebnisse beeinflussen, die für den Kontext dieser Arbeit von Bedeutung sind. Aus den gleichen Gründen wurde der senkrechte Text am rechten Rand, sowie ausgelesener Text aus Bildern nicht in die Bewertung einbezogen.

Ein Fehler lässt sich dabei folgendermaßen definieren: Ein Zeichen, das sich unterscheidet oder komplett fehlt. Wird ein komplettes Wort falsch erkannt, wird der Fehler ebenfalls durch die Anzahl dessen falscher Zeichen aufgefasst. Die Genauigkeit lässt sich dann prozentual berechnen, indem die Anzahl richtig erkannter Zeichen durch die gesamte Anzahl von Zeichen dividiert wird. Fehler wurden dabei mehrfach manuell mit Hilfe der Webseite „60tools“ [20] identifiziert. Diese stellt einen sogenannten „Diff“ bereit, ein Tool zum Vergleichen zweier Texte.

Zuvor wird noch geprüft, ob die Ergebnisse der OCR-Engine konsistent sind. Um dies zu überprüfen, wurden mehrere Durchläufe ausgeführt und verglichen. Dabei ergab sich unter allen Durchläufen stets eine Genauigkeit von 100%. Daraus lässt sich schließen, dass die OCR-Engine für ein PDF gleichbleibende Ergebnisse liefert. Bei den Messungen werden also keine variierenden Ergebnisse berücksichtigt.

Im Folgenden wird der erkannte Text des OCR-Vorgangs für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] (**Test 1**) mit der korrekten Version verglichen. Die Messungen sind in Abbildung 5.1 zu sehen. Bei fast 42000 Zeichen sind lediglich 146 Fehler aufgetreten, was einer Genauigkeit von **99,652%** entspricht. Vergleicht man dieses Ergebnis mit den Auswertungen umfangreicher OCR-Tests [22], zeigt sich, dass eine derart hohe Genauigkeit nur unter besten Voraussetzungen möglich ist.

	OCR Text	Korrekter Text
Wörter	6480	6450
Zeichen	41969	41964
Fehler	**146**	
Genauigkeit	**(41964-146) / 41964 = 99,652%**	

Abbildung 5.1: Das Ergebnis des Vergleichs vom Text des OCR-Vorgangs für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] mit der korrigierten Version. Dabei wurde eine Genauigkeit von 99,652% erreicht.

5.1.2 Weitere Auswertungen

In diesem Abschnitt sollen zusätzlich die Texterkennungen der Arbeiten „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] (**Test 2**) und „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] (**Test 3**) überprüft werden, um die Ergebnisse miteinander vergleichen zu können. Auch hier wurde jeweils auf den Abschnitt „References“ verzichtet, was für die erste Arbeit einem Umfang von 19 Seiten und für die zweite 17 Seiten entspricht. Auch Tesseracts Layoutanalyse (OSD) soll untersucht werden, um Fehlerursachen aufzuzeigen.

Es traten erneut nur wenige Fehler auf, die sich in Tabelle 5.1 zusammenfassen lassen. Die wenigen Fehler, die bei der Texterkennung auftreten, können genregulatorische Aussagen betreffen und somit die Generierung von Netzwerkregeln beeinflussen oder einschränken. Besonders die rot markierten Fehlerursachen haben sich dabei als kritisch herausgestellt. Kleinste Änderung in der Bezeichnung eines Gens können dazu führen, dass es nicht mehr erkannt wird. Die Layouts der Arbeiten werden korrekt und im richtigen Verlauf erkannt, bis auf Ausnahmen, die vor allem Tabellen und Abbildungen betreffen. Dabei werden die Tabelleninhalte miteinander vermischt und Schrift in Abbildungen bei der Texterkennung aufgenommen. Treten dort Einheiten oder Mechanismen auf, können falsche Netzwerkregeln entstehen.

Somit lässt sich feststellen, dass trotz der hohen Genauigkeit der OCR-Engine Fehler auftreten können, welche die Endergebnisse des gesamten Prozesses beeinflussen.

	OCR: Fehlerursachen	**Layout: Fehlerursachen**
Test 1	Kursive Schrift, Buchstaben mit Akut/Akzent, tiefer gestellte Zeichen, Fremdwörter/spezielle Begriffe, mathematische Operatoren, Sonderzeichen, Fehlende oder zusätzliche Leerzeichen, Fremdsprachiger Schriftsatz (z.B. ß), Anhängen von falschen Buchstaben, Erkennen von Buchstaben statt Zahlen und umgekehrt, Erkennen falscher Buchstaben	Tabellen, wechselnde Textausrichtung, Abbildungen
Test 2	Sonderzeichen, Exponenten und tiefer gestellte Zeichen, Fremdsprachiger Schriftsatz (z.B. ß), Einheitenzeichen, Anhängen von falschen Buchstaben, Erkennen von Buchstaben statt Zahlen und umgekehrt, Erkennen falscher Buchstaben	Titelseite, Abbildungen, Tabellen
Test 3	Sonderzeichen, Exponenten, fehlende Leerzeichen, Fremdsprachiger Schriftsatz (z.B. β), Anhängen von falschen Buchstaben, Erkennen von Buchstaben statt Zahlen und umgekehrt, Erkennen falscher Buchstaben	Tabellen, Abbildungen

Tabelle 5.1: Die Auswertungen der Texterkennung (OCR) zusammengefasst in einer Tabelle, mit den Fehlerursachen zur Text- und Layouterkennung, die für die ausgeführten Tests festgestellt wurden.

5.2 Auswertung der generierten Netzwerkregeln

Als nächstes sollen die generierten Netzwerkregeln überprüft werden, welche der Prototyp aus den wissenschaftlichen Publikationen „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8], „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] und „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] extrahiert. Dabei werden verschiedene Parameter berücksichtigt: Zum einen wird die Korrektheit der gefunden Einheiten geprüft, zum anderen die logischen Verknüpfungen. Da auch die Performance dieses aufwendigsten Teilprozessschritts von Bedeutung ist, soll eine kurze Analyse zur Laufzeit einen Einblick in die Dauer des Vorgangs geben. Mit der Auswertung der genannten Faktoren, erhält man einen Überblick über die Fähigkeiten und Grenzen des Vorgangs zur Regelgenerierung, der mit Hilfe von NLP und der Software INDRA erfolgt. Dazu wird das vollständig korrigierte Ergebnis der OCR überreicht, erneut ohne den Abschnitt „References“, um eine Aussage über das bestmögliche Ergebnis treffen zu können. Der Prozess erfolgt nach Start automatisch ohne menschliches Einwirken.

In der folgenden Tabelle 5.2 wurden die korrekten Netzwerkregeln der Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8], die auf Seite 5 der gleichen Publikation aufgelistet sind und zudem vom Institut für Medizinische Systembiologie in betrachteter Form bereitgestellt wurden, in die Spalte **Korrekt** eingefügt. In der Spalte **Generiert** wurden die generierten Netzwerkregeln des Prototyps entsprechend ihrer ersten Einheit den korrekten Regeln zugeordnet. Auf diese Weise kann ein direkter Vergleich gezogen werden. Nach der Analyse verschiedener Faktoren kann ein Prozentsatz aufgestellt und zur Auswertung herangezogen werden.

Für Zellen mit einem * wurden keine passenden Regeln gefunden. Insgesamt hat der Prototyp 9 Netzwerkregeln aus 24 aufgestellten INDRA-Statements generiert. Da mit jedem Statement auch der dazugehörige Satz im Prototyp angezeigt wird, konnte folgendes festgestellt werden: Einige der erkannten Einheiten wurden im Laufe des Prozesses anders bezeichnet. Da es sich dabei um Aliase handelt, kann dies jedoch vernachlässigt werden. Folgende Bezeichnungen können als gleich betrachtet werden: Das Gen *Cyclin E* wird auch *ccne1* genannt. *RB1* und *Rb* meinen dasselbe, sowie *Cyclin_A* und *cycA*. *PSMD9* ist im Kontext des Zellzyklus nicht kor-

rekt, wird jedoch nicht als Fehler betrachtet, da man über das INDRA-Statement zurückverfolgen kann, dass es aus der ursprünglich korrekten Bezeichnung *p27* generiert wurde.

Nr.	**Korrekt**	**Generiert**
1	*CycD, CycD*	*
2	*Rb, (! CycA & ! CycB & ! CycD & ! CycE) | (p27 & ! CycB & ! CycD)*	*RB1, (CycD | Cyclin_E) & ! (ccne1 | cycA)*
3	*E2F, (! Rb & ! CycA & ! CycB) | (p27 & ! Rb & ! CycB)*	*
4	*CycE, (E2F & ! Rb)*	*Cyclin_E, cdk inhibitor*
5	*CycA, (E2F & ! Rb & ! Cdc20 & ! (Cdh1 & UbcH10)) | (CycA & ! Rb & ! Cdc20 & ! (Cdh1 & UbcH10))*	*Cyclin_A, (cdk inhibitor) & ! (CDH1)*
6	*p27, (! CycD & ! CycE & ! CycA & ! CycB) | (p27 & ! (CycE & CycA) & ! CycB &! CycD)*	*PSMD9, ! (ccne1 | CycD | cycA)*
7	*Cdc20, CycB*	*CDC20, (Cyclin | ganS) & ! (CDH1 | RB1)*
8	*Cdh1,(! CycA & ! CycB) | (Cdc20) | (p27 & ! CycB)*	*CDH1, (CDC20) & ! (cycA)*
9	*UbcH10, ! Cdh1 | (Cdh1 & UbcH10 & (Cdc20 | CycA | CycB))*	*
10	*CycB, ! Cdc20 & ! Cdh1*	*

Tabelle 5.2: Ein direkter Vergleich der korrekten Netzwerkregeln, die für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] vorliegen, im Vergleich zu den generierten Netzwerkregeln des implementierten Prototyps.

Drei weitere Regeln wurden vom Prototyp generiert, jedoch nicht in die Tabelle eingefügt:

- *Ccne1, cdk inhibitor* konnte nicht zugeordnet werden
- *CDK2, cdk inhibitor* konnte nicht zugeordnet werden
- *CycA, (cdk inhibitor) & ! (Cdh1)* gleicht Nr. 5 in Spalte **Generiert**

Im Folgenden werden die generierten Netzwerkregeln mit den korrekten in zwei Kategorien abgeglichen, um eine Aussage über die Korrektheit und Vollständigkeit treffen zu können.

5.2.1 Prüfen der molekularen Einheiten

In den korrekten Netzwerkregeln kommen insgesamt 11 verschiedene Einheiten (Gene, Proteine usw.) vor, von denen im eigenen Prozess gesamt 9 erkannt und zum Aufbau von Regeln herangezogen wurden. Nicht erkannt wurden *E2F* und *UbcH10*. Dies macht einen Prozentsatz von **82%** erkannter Einheiten.

Weiterhin soll die Korrektheit innerhalb der generierten Regeln geprüft werden. Dabei werden die Reihen der Tabelle 5.2 herangezogen, für die generierte Netzwerkregeln vorliegen. Anhand der Nummer können diese zurückverfolgt werden. Folgende Auswertung ergab sich, wobei eine mehrfach vorkommende Einheit in einer Regel nur einmal gezählt wird:

Nr.	**Korrekt erkannt**	**Nicht erkannt**
2	*Rb, CycD, CycE / ccne1, CycA*	*CycB, p27*
4	*CycE*	*E2F, Rb*
5	*CycA, Cdh1*	*E2F, Rb, Cdc20, UbcH10*
6	*p27 / PSMD9, CycE / ccne1, CycD, CycA*	*CycB*
7	*Cdc20*	*CycB*
8	*Cdh1, Cdc20, CycA*	*p27, CycB*

Tabelle 5.3: Die generierten Netzwerkregeln für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] werden auf die Korrektheit ihrer Einheiten überprüft. In der linken Spalte werden die korrekt erkannten Einheiten je Regel aufgelistet, in der rechten Spalte die nicht erkannten.

Von den insgesamt 27 Einheiten, die sich aus der Summe der Einheiten pro Regel ergeben, werden also 15 (linke Spalte) korrekt erkannt. 12 Einheiten fehlen dabei in den generierten Netzwerkregeln, was letztendlich einem Prozentsatz von ca. **55%** korrekt erkannter Einheiten entspricht.

5.2.2 Prüfen der logischen Operatoren

Da für eine korrekte Netzwerkregel nicht nur die richtigen Einheiten, sondern auch deren logische Verknüpfung wichtig ist, sollen auch diese untersucht und ausgewertet werden.

Zunächst werden die generierten Netzwerkregeln mit Hilfe der mathematischen Gesetze für boolesche Algebra umgeformt, um korrekte Teilabschnitte besser erkennen und vergleichen zu können. Dabei ergaben sich folgende Änderungen:

Nr.	**Korrekt**	**Generiert**
2	*Rb, (! CycA & ! CycB & ! CycD & ! CycE) | (p27 & ! CycB & ! CycD)*	*Rb, (! CycA & ! CycE) & (CycD | Cyclin_E)*
4	*CycE, (E2F & ! Rb)*	*CycE, cdk inhibitor*
5	*CycA, (E2F & ! Rb & ! Cdc20 & ! (Cdh1 & UbcH10)) | (CycA & ! Rb & ! Cdc20 & ! (Cdh1 & UbcH10))*	*CycA, (cdk inhibitor) & (! Cdh1)*
6	*p27, (! CycD & ! CycE & ! CycA & ! CycB) | (p27 & ! (CycE & CycA) & ! CycB &! CycD)*	*p27, (! CycD & ! CycE & ! CycA)*
7	*Cdc20, CycB*	*Cdc20, (Cyclin | ganS) & (! Cdh1 & ! Rb)*
8	*Cdh1,(! CycA & ! CycB) | (Cdc20) | (p27 & ! CycB)*	*Cdh1, (! CycA) & (Cdc20)*

Tabelle 5.4: Ein direkter Vergleich der korrekten Netzwerkregeln, die für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] vorliegen, im Vergleich zu den generierten Netzwerkregeln des implementierten Prototyps, die mathematisch umgeformt wurden.

Wie bereits in Kapitel 2 vorgestellt, gibt es drei verschiedene Operatoren. Darunter das logische „UND“, „ODER“ und „NICHT“, dargestellt als „&“, „|“ und „!“. Durch diese Verknüpfungen kann eine Netzwerkregel komponentenweise aufgebaut werden. Da sich eine „NICHT“ Verknüpfung auf nur eine Einheit und ein „UND / ODER“ auf mindestens zwei Einheiten bezieht, werden diese getrennt voneinander ausgewertet. Die Ergebnisse können in Tabelle 5.5 eingesehen werden. Dabei werden nur die Funktionen berücksichtigt, deren Einheiten im vorherigen Abschnitt erkannt wurden.

Nr.	**NICHT**		**UND / ODER**	
	Korrekt erkannt	Nicht erkannt	Korrekt erkannt	Nicht erkannt
2	***!*** *CycA,* ***!*** *CycE*	***!*** *CycD*	*! CycA* ***&*** *! CycE*	***..*** */ (! CycD* ***&*** ***..****)*
4	-	-	-	-
5	***!*** *Cdh1*	-	-	-
6	***!*** *CycD,* ***!*** *CycE,* ***!*** *CycA*	-	*(CycD* ***&*** *! CycE* ***&*** *! CycA*	-
7	-	-	-	-
8	***!*** *CycA, Cdc20*	-	-	***..*** */ Cdc20*

Tabelle 5.5: Die generierten Netzwerkregeln für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] werden auf die Korrektheit ihrer logischen Operatoren überprüft. In der linken Spalte werden die „NICHT“ Aussagen zugeordnet, in der rechte Spalte die „UND / ODER“ Aussagen.

Insgesamt ergeben sich 9 einstellige Verknüpfungen (vorhandenes oder fehlendes „NICHT“) und 6 „UND / ODER“ Aussagen, wenn man nur die in Tabelle 5.3 erkannten Einheiten betrachtet. Dies ergibt jeweils einen Prozentsatz von ca. **88,8%** und **50%** korrekt erkannter logischer Operatoren. Von den insgesamt 15 logischen Operatoren ergaben sich 4 falsche, was eine gesamte Korrektheit von ca. **73,3%** ergibt.

Dabei konnte, anhand der im Prototyp einsehbaren INDRA-Statements, zurückverfolgt werden, warum falsche Verknüpfungen aufgebaut werden:

1. Für Sätze mit einer einzelnen Aussage werden zwei INDRA-Statements erstellt, die sich widersprechen. Dies tritt bei Regel Nr. 2 auf, da sowohl *Activation(CycD(), RB1())* wie auch *Inhibition(CycD(), RB1())* erstellt wird, ersteres jedoch falsch ist.

2. Für Sätze mit mehreren Aussagen werden wiederum falsche Verknüpfungen erstellt, da ein Mechanismus einem falschen Gen zugeordnet wird. Dies tritt besonders dann auf, wenn eine Einheit nicht erkannt wird. Auch dies tritt bei Regel Nr. 2 auf, da *CycE* sich auf das nicht erkannte *E2F* statt *Rb* beziehen müsste.

3. Aus falsch aufgestellten INDRA-Statements folgen inkorrekte „UND / ODER“ Verknüpfungen, da Einheiten einer Kategorie z.B. *Activation* untereinander

mit einem logischen „ODER“ und über ein logisches „UND“ mit anderen Kategorien verknüpft werden.

4. Kommt die Bezeichnung einer Einheit in einem Satz mehrfach in unterschiedlicher Form vor (Aliase), wird das INDRA-Statement für jedes dieser Bezeichnungen aufgestellt. Deshalb treten Fehler wie in Regel Nr. 2 auf und dieselbe Einheit wird doppelt mit unterschiedlicher Bezeichnung herangezogen.

5.2.3 Weitere Auswertungen

In diesem Abschnitt sollen die Netzwerkregeln von zwei weiteren Publikationen untersucht werden. Dabei bleibt das Auswertungsverfahren gleich, um die Ergebnisse der Arbeiten vergleichen zu können. Die Bool'schen Funktionen der Publikation „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] (**Test 2**) und „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] (**Test 3**) sind tabellarisch enthalten und werden erneut mit den generierten Netzwerkregeln des Prototyps gegenübergestellt, abgebildet in den Tabellen 5.6 und 5.7.

Test 2

In diesem Abschnitt werden die Netzwerkregeln der Arbeit „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27], die tabellarisch auf Seite 10 und 11 der Publikation enthalten sind, mit den Regeln des Prototyps abgeglichen. Der Vergleich ist in Tabelle 5.6 einzusehen.

Nr.	**Korrekt**	**Generiert**
1	*TCF7L2, (! PRKD1)*	*TCF7L2, PRKD1 | RHOA | protein kinase*
2	*AURKA, PAK1*	*AURKA, RAC1*
3	*Phosphorylated-CFL1, CD44 | TCF7L2 | (CFL1 & LIMK & ! SSH1L)*	*CFL1, (RAS | AURKA | doxycycline | Actin | TCF7L2 | CD44) & ! (LIMK1 | CASP14)*
4	*CFL1, (SSH1L & Phosphorylated-CFL1) | (SSH1L & LIMK & Phosphorylated-CFL1)*	-
5	*CD44, TWIST1*	*CD44, TWIST1*
6	*TWIST1, AURKA*	*TWIST1, ! (AURKA)*
7	*LIMK, (RHOA | PAK1 | PAK4) & ! SSH1L*	*LIMK1, PAK4 | ROCK | PAK1*
8	*SSH1L, ((F-actin_new | F-actin_old) & ! PRKD1) | (PI3K & AURKA) | (LIMK & SSH1L)*	*SSH1, F_actin | AURKA*
9	*F-actin_new, (CFL1 & ARP2/3) | (RHOA(-3) & ! CFL1)*	-
10	*F-actin_old, (F-actin_old & ! CFL1) | F-actin_new*	-
11	*ARP2/3, RAC1(-2)*	*Actin, RAS | KRAS | RAC1*
12	*KRAS, 1*	-
13	*PI3K, KRAS | CD44*	-
14	*PRKD1, RHOA*	*PRKD1, RHOA*
15	*RHOA, ! PAK4*	*RHOA, ! (PAK4)*
16	*RAC1, PI3K | Phorsphorylated-CFL1(-3)*	*RAC1, PI3K | CFL1*
17	*PAK1, RAC1*	*PAK1, RAC1*
18	*PAK4, RAC1*	*PAK4, RAC1*
19	*CDH1, ! TWIST1*	*CDH1, ! (TWIST1)*
20	*CTNNB1, PAK1 & ! PRKD1 & ! CDH1*	*CTNNB1, ! (CDH1 | PRKD1)*
21	*GSK3B, ! AKT*	*GSK3B, ! (AKT)*
22	*MYC, (! GSK3B & STAT3) | (CTNNB1 & ! GSK3B)*	*MYC, (STAT3) & ! (GSK3B)*
23	*CCND1, (! GSK3B & MYC & AKT)*	*CCND1, AKT | MYC*
24	*RB, ! CCND1*	*RB1, ! (CCND1)*
25	*E2F, ! RB*	-
26	*CCNE1, E2F*	-

Nr.	**Korrekt**	**Generiert**
27	*S-phase, E2F & CCNE1*	-
28	*AKT, PI3K & STAT3*	*AKT1, STAT3*
29	*STAT3, (Phosphorylated-CFL1 | CFL1) & CD44*	*STAT3, CD44*
30	*Anti-apoptotic proteins, STAT3*	*BCL2L1, STAT und MCL1, STAT*
31	*Pro-apoptotic proteins, ! AKT*	-
32	*CYCS, Pro-apoptotic proteins & ! Anti-apoptotic proteins & CFL1*	-
33	*Caspases, CYCS & ! AKT*	*Caspase, CYCS und CASP9, ! (AKT)*

Tabelle 5.6: Ein direkter Vergleich der korrekten Netzwerkregeln, die für die Arbeit „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] vorliegen, im Vergleich zu den generierten Netzwerkregeln des implementierten Prototyps.

Für den korrigierten Text ergaben sich insgesamt 73 INDRA-Statements und 37 Netzwerkregeln. Anhand des INDRA-Statements und der zugehörigen Evidence (Textbeweis) „*actin-related protein 2/3 complex (ARP2/3)*“ konnte *Actin* der Regel Nr. 11 zugeordnet werden. Aus der Arbeit geht hervor, dass sich *Anti-apoptotic proteins* aus den Einheiten *BCL2L1* und *MCL1* aufbaut. Regel Nr. 2 wird als korrekt betrachtet, da aus der Evidence des INDRA-Statements hervorgeht: „*[...], RAC1 activates aurora kinase A (AURKA)*“ gilt. Das gleiche tritt bei Nr. 7 auf, denn im Text heißt es „*[...] ROCK (downstream of RHOA)*“.

Gesamt wurden also 23 von 33 Netzwerkregeln generiert, die den korrekten zugeordnet werden konnten, was einen Prozentsatz von ca. **70%** macht und somit mehr als im letzten Versuch, dort waren es genau **60%**.

Wie in der zuvor betrachteten Arbeit, wird auch hier *E2F* nicht erkannt. Auch *BAD (pro-apoptotic proteins)*, *CCNE1* und *ARP2/3* treten in keinem INDRA-Statement auf. *Anti-apoptotic proteins* wurde ebenfalls nicht erkannt, jedoch die dazugehörigen Einheiten (siehe Nr. 30 in 5.6), weshalb hier kein Fehler gezählt wird. *CFL1* wird vom Prototyp nicht von *Phosphorylated-CFL1* unterschieden, sowie *F-actin_new* nicht von *F-actin_old*. Von den insgesamt 34 auftretenden Einheiten, wobei *Anti-apoptotic proteins* wegen *BCL2L1* und *MCL1* doppelt gezählt wird, treten 25 in mindestens einem Statement auf. Das macht ca. **74%**, im Vergleich also weniger als

im letzten Test.

Um die Vollständigkeit innerhalb der gefundenen Regeln zu prüfen, werden erneut die unterschiedlichen Einheiten pro Reihe addiert. Für die korrekte Spalte ergeben sich 68 Einheiten, für die Spalte der generierten Netzwerkregeln 56, was gesamt ca. **82%** macht. Damit sind die Regeln deutlich vollständiger als im letzten Versuch, obwohl weniger Einheiten erkannt bzw. unterschieden werden.

Auch die Korrektheit innerhalb der generierten Regeln wird geprüft, indem die logischen Verknüpfungen untersucht werden. Insgesamt ergeben sich 34 einstellige Verknüpfungen (vorhandenes oder fehlendes „NICHT“) und 10 „UND / ODER“ Aussagen, wenn man nur die Operatoren zwischen erkannten Einheiten betrachtet. Es sind 31 korrekte einstellige und 7 korrekte zweistellige Verknüpfungen, was einen Prozentsatz von ca. **91%** und **70%** korrekt erkannter logischer Operatoren macht. Von den insgesamt 44 logischen Operatoren sind also 6 falsch, was eine gesamte Korrektheit von ca. **86%** ergibt und ebenfalls höher ist als im anderen Test.

Folgende Fehlerursachen traten bei diesem Test auf:

- Die Fehlerursachen 3 und 4 aus der letzten Auswertung treten auch hier auf, z.B. bei Regel Nr. 1 in 5.6.
- Modifikationen wie *Phosphorylated-CFL1* werden nicht unterschieden und müssen manuell nachgearbeitet werden. Im INDRA-Statement ist die Modifikation teilweise zu erkennen, z.B. bei *Activation(CFL1(mods: (phosphorylation)), RAC1())*.
- Aktivitätszustände wie in Regel Nr. 12 werden vom Prototyp generell nicht verarbeitet.

Test 3

In diesem Abschnitt werden die Netzwerkregeln der Arbeit „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26], die tabellarisch auf Seite 7 enthalten sind, mit den Regeln des Prototyps abgeglichen. Der Vergleich ist in Tabelle 5.7 einzusehen.

Nr.	**Korrekt**	**Generiert**
1	*Wnt, Wnt*	*Wnt, (RAC | JNK | PKC | IGF1 | TCF_LEF) & ! (KL | AXIN2)*
2	*axin, ERK | ! Wnt*	*AXIN2, ERK*
3	*GSK3β, !(Wnt | ERK | Akt)*	*GSK3β, ! (ERK | AKT)*
4	*DC, axin & GSK3β*	-
5	*β-catenin, !DC*	*β-catenin, AXIN2*
6	*TCF, β-catenin & !(JNK & FoxO)*	*TCF_LEF, ! (DVL)*
7	*FoxO, !Akt & β-catenin*	*FOXO, (JNK) & ! (AKT | DVL | IGF1)*
8	*Rho, (Wnt | PI3K | mTORC2) & !(Rac | PKC)*	*RHO, (Wnt | JNK) & ! (PKC)*
9	*Rac, (Wnt | PI3K | mTORC2) & !Rho*	*RAC, Wnt | JNK*
10	*MEKK1, Rac | Rho*	*MAP3K1, RHO | RAC*
11	*JNK, MEKK1 | Rac*	*JNK, RAC*
12	*PKC, Rho | Wnt | mTORC2*	*PKC, PTPN5 | Wnt | mTORC2*
13	*IGF, IGF*	*IGF1, (Wnt | PI3K | AKT) & ! (IRS JNK)*
14	*IRS, IGF & !(S6K & JNK)*	*IRS, ! (sok | JNK)*
15	*PI3K, (IRS | Ras) & !Rho*	*PI3K, (IRS) & ! (PTEN | RHO)*
16	*Akt, PI3K | mTORC2*	*AKT, IRS | PI3K | mTORC2 | IGF1*
17	*TSC2, !(Akt | ERK) | GSK3β*	*TSC2, (GSK3β) & ! (PI3K Akt | ERK)*
18	*mTORC1, !TSC2*	*mTORC1, ! (TSC2)*
19	*S6K, mTORC1 | GSK3β*	*p70-S6 kinase, mTORC1*
20	*Ras, IGF | Wnt*	*RAS, IGF1 | DVL*
21	*Raf, (Ras | PKC) & !Akt*	*RAF, (RAS | PKC) & ! (AKT)*
22	*ERK, Raf*	*ERK, IGF1*
23	*mTORC2, !(S6K | GSK3β) & (PI3K | TSC2)*	*mTORC2, ! (adenosine 5'-monophosphate | GSK3β)*

Tabelle 5.7: Ein direkter Vergleich der korrekten Netzwerkregeln, die für die Arbeit „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] vorliegen, im Vergleich zu den generierten Netzwerkregeln des implementierten Prototyps.

Für den korrigierten Text ergaben sich insgesamt 97 INDRA-Statements und 46 Netzwerkregeln. Die Zuordnung erfolgt wieder mit Hilfe der INDRA-Statements und der Evidence, wenn die Bezeichnung nicht eindeutig war. Sind Bezeichnungen im Kontext der Arbeit falsch, kann dies ebenfalls zurückverfolgt und korrigiert werden und wird nicht als Fehler gezählt. Da die OCR-Engine kein griechisches Beta erkennt und stattdessen ein B erzeugt, was den NLP-Vorgang jedoch nicht ein-

schränkte, wurde dies in Tabelle 5.7 nachkorrigiert.

Gesamt konnten 22 generierte den 23 korrekten Netzwerkregeln zugeordnet werden, was ca. **96%** macht und damit den höchsten Prozentsatz aller Tests. Nicht erkannt wurde lediglich *DC*, ansonsten treten alle Einheiten auf, wenn auch mit anderer Bezeichnung.

Überprüft man die Vollständigkeit innerhalb der Regeln, fällt auf, dass auch hier oft Einheiten fehlen, vor allem mit steigender Größe der korrekten Netzwerkregeln. Außerdem fällt auf, dass der Prototyp z.B. in Regel Nr. 1, 7, 13 und 16 der Tabelle 5.7 deutlich mehr Einheiten heranzieht als in der korrekten Regel auftreten. Die Korrektheit der Netzwerkregeln muss deshalb vom Nutzer geprüft werden, um falsche Aussagen auszuschließen. Auch für die logischen Operatoren treten wieder vereinzelte falsche Verknüpfungen auf.

Die Fehlerursachen lassen sich wie folgt zusammenfassen:

- Fehlerursache 1 der ersten Auswertung tritt auch hier auf, z.B. mit *Inhibition(KL(), Wnt())* und *Activation(KL(), Wnt())* für denselben Satz.
- Fehlerursache 2 der ersten Auswertung tritt auch hier auf, z.B. in Regel Nr. 5 in Tabelle 5.7 Da *DC* nicht erkannt wird, wird fälschlicherweise *axin* herangezogen.
- Fehlerursache 4 der ersten Auswertung tritt auch hier auf.
- Durch die Fußnoten der Publikation, werden Sätze, die über zwei Seiten ragen, unterbrochen und es können Fehler auftreten.

Vergleichen der Ergebnisse

Nun soll untersucht werden, wodurch die Unterschiede der Ergebnisse zustande kommen. Zunächst fällt auf, dass die Netzwerkregeln des zweiten Tests größtenteils kürzer ausfallen als die des ersten, das heißt pro Regel treten weniger Einheiten und damit auch Operatoren auf. Die in der zweiten Publikation enthaltenen Regeln, welche aus insgesamt zwei Einheiten bestehen, wurden dabei nahezu korrekt generiert, vorausgesetzt die beteiligten Einheiten wurden vom Prototyp erkannt. Bei zunehmender Größe der Regeln fehlen jedoch einige Einheiten zur Vollständigkeit

und es treten falsche logische Verknüpfungen auf. Dies tritt bei allen durchgeführten Versuchen auf. Der dritte Test zeigt außerdem, dass die Verknüpfungen, die zur Vollständigkeit einer Netzwerkregel fehlen, nicht unbedingt mit den insgesamt erkannten Einheiten zusammenhängt. Dort wurden alle Einheiten erkannt, die in langen Netzwerkregeln wiederum fehlen. Das bedeutet, der Prototyp kann die textuell beschriebene Verknüpfung nicht filtern, auch wenn die zugehörigen Einheiten erkannt werden.

Die aufgezählten Fehlerursachen deuten außerdem darauf hin, dass besonders dann Fehler auftreten, wenn in Sätzen mehrere genregulatorische Aussagen oder verschiedene Bezeichnungen auftreten. Dies lässt auch wieder Rückschlüsse zu, dass Sätze mit einzelnen Aussagen korrekter verarbeitet werden.

5.2.4 Prüfen der Laufzeit

Abschließend soll die Zeit gemessen werden, die der NLP-Vorgang und die anschließende Umwandlung mit INDRA in Anspruch nimmt, da auch die Performanz des Prozesses wichtig ist für ein effizientes Arbeiten mit dem implementierten Vorgang. Dazu werden erneut dieselben wissenschaftlichen Arbeiten herangezogen und der Vorgang mehrfach mit unterschiedlicher Seitenzahl ausgeführt. Die Ergebnisse wurden mit Hilfe der Webseite „Diagramm erstellen“ [28] veranschaulicht. Die Analyse wurde auf einem Laptop mit einem 1,4 GHz Quad-Core Intel i5 Prozessor und 16 GB RAM durchgeführt.

In Abbildung 5.2 ist zu erkennen, dass die Zeiten für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] aus drei Durchgängen nur um wenige Sekunden schwanken und einen logarithmischen Verlauf annehmen. In **Durchlauf 3** fällt auf, dass die Zeit für 1 Seite beinahe um das 7-fache erhöht ist. Hier muss angemerkt werden, dass der lokale REACH Server neu gestartet wurde. Dies sollte aufzeigen, ob der Serverneustart die Laufzeit verlängert, was offenbar nur beim ersten Vorgang der Fall ist und der Graph sich anschließend auf die vorherigen Zeitmessungen einpendelt. Die Gesamtdauer für 8 Seiten beträgt hier durchschnittlich 1 Minute und 43 Sekunden.

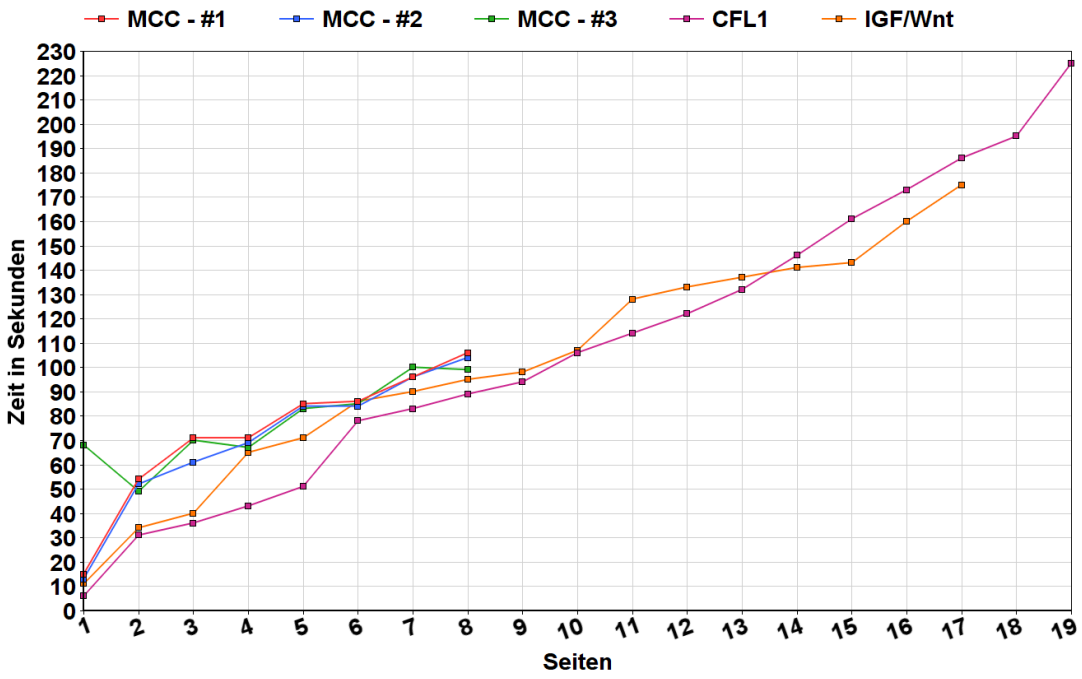

Abbildung 5.2: Die Laufzeitanalyse für den NLP-Vorgang, veranschaulicht mit Graphen. Die x-Achse steht für die Anzahl der Seiten des PDFs, die y-Achse für die Bearbeitungszeit in Sekunden. Dabei wurden für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] drei Durchläufe ausgeführt (rot, blau und grün) und für die Arbeit „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] einer (gekennzeichnet in lila mit „CFL1“), sowie für „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] (gekennzeichnet in orange mit „IGF/Wnt“).

Für die Arbeit „Unraveling the Molecular Tumor-Promoting Regulation of Cofilin-1 in Pancreatic Cancer“ [27] (in der Abbildung lila und mit „CFL1“ bezeichnet) wurden daher nur die Zeiten eines Durchlaufes gemessen. Hier fällt auf, dass die Zeiten pro Seitenanzahl etwas kürzer sind. Dies lässt sich aber darauf zurückführen, dass hier weniger Zeichen pro Seite auftreten. Beispielsweise setzt sich Seite 1 aus 4 578 Zeichen zusammen, während es in der zuerst betrachteten Arbeit 6 187 Zeichen sind. Ab der Seitenspanne 9 bis 18 nimmt der Graph einen nahezu linearen Verlauf an, was bedeutet der Prozess erfolgt langsamer als mit niedrigeren Seitenzahlen. Der Sprung von 18 auf 19 lässt vermuten, dass der Vorgang nach 18 Seiten, was

in in dieser Arbeit 62 243 Zeichen entspricht, nochmals langsamer erfolgt. Die Gesamtdauer für 19 Seiten beträgt hier 3 Minuten und 45 Sekunden.

Für die Arbeit „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“ [26] (in der Abbildung orange und mit „IGF/Wnt“ bezeichnet) wurden die Zeiten eines Durchlaufs gemessen. Hier treten mehr Schwankungen auf, die sich aber damit erklären lassen, dass in der Publikation viele Abbildungen enthalten sind, z.B. bei Seite 5, 9 und 15, wo Gefälle im Graphen auftreten. Ansonsten ließe sich ebenfalls ein logarithmischer Verlauf erkennen, der ab 11 Seiten eventuell linear übergehen würde. Die Gesamtdauer für 17 Seiten beträgt 2 Minuten und 55 Sekunden.

Insgesamt lässt sich gut erkennen, dass die Laufzeiten relativ gleichmäßig verlaufen und keine auffälligen Schwankungen auftreten, außer beim Neustart des REACH Servers. Außerdem konnte festgestellt werden, dass die Zeiten weniger von der Anzahl der enthaltenen Netzwerkregeln, sondern vielmehr von der Anzahl der übergebenen Zeichen abhängt.

6 Diskussion

In dieser Arbeit wurde ein Prototyp entwickelt, der mit Hilfe von OCR, NLP und der Python-Software INDRA sogenannte Netzwerkregeln aus PDF-Dateien generieren kann. Eine Evaluation verschiedener Parameter hat gezeigt, dass der implementierte Prozess korrekte Ergebnisse liefern und somit bei der Extraktion von Netzwerkregeln unterstützen kann.

6.1 Prototyp

Im Vergleich zur Konzeption haben sich für den realisierten Prototyp wenige Änderungen ergeben. Momentan ist der Prozess zur Regelgenerierung nur mit einzelnen PDFs möglich. Dies könnte künftig erweitert werden, indem ein multipler File Upload implementiert wird, was vom Prototyp und dessen verwendeter Software unterstützt wird. Damit könnten auch zusammenhängende Netzwerkregeln aus mehreren PDFs generiert werden, indem man die jeweils aufgestellten INDRA-Statements verknüpft, bevor sie anschließend dem Assembler überreicht werden.

Die UI-Komponente zur Autorisierung wurde eingefügt, jedoch nicht implementiert. Dies kann bei Bedarf erweitert werden, ist für die Generierung von Netzwerkregeln jedoch nicht zwingend notwendig. Der damit verbundene Datenschutz kann auch durch eine netzwerkinterne Verwendung unterstützt werden. Da von einer gemeinsamen Nutzung der Anwendung profitiert wird, kann ein offener Zugang aber auch vorteilhaft sein.

Bis auf die genannten Punkte wurden jedoch alle in Kapitel 3 aufgestellten Anforderungen erfolgreich umgesetzt. Durch die nutzerfokussierte Konzeption in Kapitel 4 konnten auch die Qualitätsanforderungen Q01 und Q02 erfüllt werden. Die nichtfunktionalen Anforderungen Q03 und Q04 wurden in der Implementierung berück-

sichtigt und sind ebenfalls gegeben. Die Evaluation der Laufzeit hat gezeigt, dass die Dauer des aufwendigsten Teilprozesschritts in angemessenem Rahmen liegt, verglichen mit dem Zeitaufwand des manuellen Extrahierens.

Der Prototyp unterstützt folgende Erkennung, die durch REACH gegeben ist: 17 biochemische Interaktionen (z. B. Phosphorylierung, Komplexaufbau, Katalyse), relevante Teilnehmer (Gene, Proteine, einfache Chemikalien, Orte, Mutationen) und Kontextinformationen (Arten, Organe, Zelltypen, Zelllinien) [3].

6.2 Verwendete Software

Um eine allgemeine Nutzung zu ermöglichen, wurde der Prototyp in Form einer Webanwendung implementiert. Nach Bereitstellen der Webseite auf einem Webserver kann die Anwendung von jedem Gerät plattformübergreifend aufgerufen werden. Außerdem können mehrere Clients, also Nutzer, gleichzeitig darauf zugreifen und die generierten Daten gemeinsam nutzen. Auf diese Weise kann zum Beispiel das gesamte Institut für Medizinische Systembiologie von der Anwendung profitieren.

Der Prototyp läuft innerhalb eines Virtual Enviroments, um das Projekt vom restlichen System zu isolieren und Versionskonflikte zu vermeiden. Hierfür diente die venv Library für Python 3. Das gesamte Projekt wurde in der IDE Visual Studio Code (Version 1.59.1) entwickelt, das alle benötigten Implementierungstools bereitstellt.

Zum Erstellen der Webanwendung wurde das Open Source und Full Stack Framework Django genutzt, das wie INDRA auf Python basiert und damit die Kombination der Software erleichtert, weil keine weiteren Wrapper benötigt werden. Verglichen mit anderen Python-Frameworks wie Flask, bietet Django mehr Features, darunter einen vorgefertigten Admin-Bereich, integriertes „Object-relational mapping“ (ORM), Sicherheit gegen übliche Bedrohungen sowie eine umfangreiche Community. Dieser „batteries included “ Ansatz von Django hat eine schnelle Entwicklung der Webanwendung ermöglicht, wodurch nicht nur der Prozess zur Regelgenerierung, sondern auch eine stabile und nutzerfreundliche Anwendung verwirklicht werden konnte. Durch den Fokus auf das „Don’t Repeat Yourself“ (DRY) Prinzip zählt es sogar zu einem der produktivsten Frameworks für Webentwicklung [24]. Bei diesem Prinzip geht es darum Wiederholungen zu vermeiden, das heißt Komponenten

werden nur an einer Stelle im Projekt definiert und nicht mehrfach wiederholt. Damit wurde ebenfalls die Entwicklung beschleunigt, sowie Wiederholungsfehler vermieden.

Bei den fünf Datenbanken, die Django offiziell unterstützt, handelt es sich um relationale Datenbankmanagementsysteme, die mit ihrer Struktur für den Kontext dieser Arbeit passend sind. Für den Prototyp wurde das Open Source Datenbanksystem PostgreSQL als RDBMS verwendet, welches die Datenbanksprache SQL („Structured Query Language“) nutzt. Im Vergleich zu MySQL, das ebenfalls häufig genutzt wird, erhält PostgreSQL mehr Community Unterstützung [7]. Außerdem unterstützt es die Verwendung von Arrays, die für das Speichern der INDRA-Statements und Netzwerkregeln benötigt werden. Damit war PostgreSQL auch eine geeignetere Wahl gegenüber dem standardmäßig integrierten DBMS SQLite, welches für eine spätere Produktionsumgebung auch nicht empfohlen wird [7].

Um die hochgeladenen PDF-Seiten in Bildformate zu konvertieren, wurde PDF2image genutzt. Dies ist ein Python Wrapper für Poppler, um PDF-Dateien in PIL-Bildobjekte zu konvertieren. PIL ist die „Python Image Library“ und bietet dem Python-Interpreter Funktionen zur Bildbearbeitung. Der Vorteil des Tools ist, dass mit der Konvertierung auch alle benötigten Konfigurationen für die Bilder, die in Kapitel 4 erläutert wurden, festgelegt werden. Somit erhält man mit nur einer Methode alle Voraussetzungen, die man für optimale Ergebnisse der OCR-Engine benötigt. Gegenüber anderen Tools wie Ghostscript überzeugte PDF2image vor allem durch seine leichte Anwendung.

Für das User Interface wurde das frei zur Verfügung stehende „Bootstrap SB Admin Template“ verwendet, welches auf der offiziellen Website von Bootstrap zum Download bereitsteht. Die Oberfläche wurde dann entsprechend den Mockups angepasst, mit den Änderungen, die sich im Laufe der Implementierung ergaben. Das darin integrierte Open Source Plug-In „DataTables“ wurde auch für die eigene Tabelle verwendet. Damit ließen sich Funktionen wie das Durchsuchen und Sortieren der Datenbank, sowie das Ausblenden von Spalten ohne umfangreiche Integration und Konfiguration umsetzen.

6.3 Evaluationsergebnisse

Zunächst sollen die Evaluationsergebnisse der Texterkennung (OCR) betrachtet werden. Die vorgenommenen Tests haben vergleichbare Ergebnisse geliefert. Für die Arbeit „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“ [8] konnte eine Genauigkeit von 99,652% festgestellt werden, was zeigt, dass die Konfigurationen der Bilder und der Engine, wie in Kapitel 4 vorgestellt, sehr gute Ergebnisse liefern. Umfangreiche OCR-Tests wie Quelle [22] unterstreichen diese Aussage, da eine derart hohe Genauigkeit nur unter besten Voraussetzungen möglich ist. Die festgestellten Fehlerursachen lassen sich wie folgt zusammenfassen: Die falsche Erkennung von Sonderzeichen, fremdsprachigem Schriftsatz, Fremdwörter und Mathematik bezogene Fehler beeinflussen die Generierung von Netzwerkregeln nicht. Jedoch können falsch erkannte und zusätzliche Buchstaben oder Zahlen zu veränderten Genbezeichnungen führen. Dies kann im schlimmsten Fall dazu führen, dass die Einheiten im nächsten Teilprozessschritt nicht mehr erkannt und deshalb nicht als Bestandteil einer Netzwerkregel aufgenommen werden. Diese Schwäche des OCR-Vorgangs könnte künftig optimiert werden, indem z.B. die Wörterbücher nicht nur ausgeschaltet, sondern stattdessen eine eigene Wortliste mit fehleranfälligen Genbezeichnungen überreicht wird. Dies bezieht sich auf die Verwendung von Tesseract. Momentan können solche Fehler jedoch auch durch den Nutzer korrigiert werden.

Außerdem konnte festgestellt werden, dass die automatische Layouterkennung von Tesseract wechselnde Layouts korrekt erkennt und den Text im richtigen Verlauf wiedergibt. Dadurch ist die Effizienz des Prototyps nicht an ein Layout gebunden und es können verschiedenste Publikationen überreicht werden. Schwierigkeiten bereiten jedoch Abbildungen, da auch dort enthaltener Text analysiert wird. Dadurch könnten zusätzliche Genbezeichnungen zu späteren Fehlern führen. Auch eine Tabellenerkenung ist bisher nicht gegeben, wodurch die Zelleninhalte vermischt oder falsch aufgeführt werden, was wiederum zu falschen Regeln führen kann. Eine künftige Erweiterung um eine „Table Recognition“ könnte diese Schwäche ausgleichen. Durch die Möglichkeit zur Bearbeitung können daraus resultierende Fehler jedoch durch den Nutzer ausgeglichen werden.

Nun sollen die Evaluationsergebnisse der generierten Netzwerkregeln betrachtet werden, die mit NLP und INDRA erfolgen. Wie sich gezeigt hat, wurden in kei-

nem der durchgeführten Tests alle vorkommenden Einheiten erkannt und somit auch nicht als INDRA-Statements aufgenommen. Diese Schwäche führt dazu, dass manche Regeln fehlen oder gefundene nicht vollständig sind. Die Netzwerkregeln, die generiert wurden, weisen jedoch eine hohe Korrektheit auf, wenn man von den fehlenden Einheiten absieht. Die erkannten Fehlerursachen, die zu falschen Verknüpfungen oder nicht erkannten Einheiten führen, können wie folgt zusammengefasst werden: Treten in Sätzen mehrere genregulatorische Aussagen auf, steigert das die Anfälligkeit für Fehler, da falsche Mechanismen zugeordnet oder mehrere INDRA-Statements für Aliase aufgestellt werden. Auch durch unklare Formulierungen können sich widersprechende INDRA-Statements aufgestellt werden. Aus inkorrekten INDRA-Statements folgen dann falsche Verknüpfungen. Weitere Schwächen sind Modifikationen, sowie kontextbezogene Bezeichnungen. Eine Stärke des Prototyps ist jedoch die Möglichkeit, die gefundenen INDRA-Statements mit ihrer Evidence einzusehen. So können auftretende Fehler zurückverfolgt und überarbeitet werden. Insgesamt fällt auf, dass mit steigender Größe von Netzwerkregeln auch mehr Fehler und Unvollständigkeiten auftreten. Dadurch kann festgelegt werden, dass die Stärke des implementierten Prozesses bei der Generierung von Netzwerkregeln liegt, die sich aus wenigen Einheiten aufbauen. Auch dies könnte mit dem Heranziehen mehrerer PDFs optimiert werden, da auf diese Weise auch mehr INDRA-Statements erstellt werden.

Die Laufzeit, die beim letzten Teilprozessschritt in Anspruch genommen wird, hängt dabei von der Anzahl der übergebenen Zeichen ab. Das bedeutet mit wachsender Größe der überreichten Publikationen verlängert sich auch die Dauer des Prozesses. Dadurch konnte festgestellt werden, dass der Prototyp umfangreiche Arbeiten zuverlässig verarbeiten kann, jedoch durch eine Laufzeit bezogene Obergrenze limitiert ist. Durch die Möglichkeit der Bereichsauswahl können umfangreiche PDFs jedoch auch portionsweise verarbeitet werden, wodurch eine zu lange Laufzeit umgangen werden kann.

Mit dem Aufstellen der INDRA-Statements sind auch Erweiterungen denkbar, die über Bool'sche Netzwerke hinausragen. Insgesamt hat der Prototyp die Zielsetzung erfüllt, indem gezeigt wurde, dass er korrekte Netzwerkregeln generieren kann. Durch genannte Erweiterungen könnten die Ergebnisse künftig optimiert werden, um noch bessere Ergebnisse zu erzielen.

7 Zusammenfassung

Mit Hilfe von optischer Texterkennung (OCR), natürlicher Sprachverarbeitung (NLP) und der Software INDRA wurde ein Prototyp implementiert, der in der Lage ist, PDF-Dateien auf genregulatorische Aussagen zu untersuchen und Bool'sche Funktionen bzw. Netzwerkregeln zu generieren. Der Prototyp unterstützt die durch das REACH Reading System ermöglichte Erkennung von biochemischen Interaktionen, damit verbundenen Einheiten und Kontextinformationen. Somit kann die Extraktion von Netzwerkregeln aus biomedizinischer Literatur technisch unterstützt werden, um eine effizientere Simulation und Analyse von Bool'schen Netzwerken zu erreichen. Dazu wurde ein Konzept aufgestellt, das die nötigen Komponenten um nützliche Funktionen erweitert, um eine nutzerfreundliche Anwendung zu erhalten.

Eine anschließende Evaluation hat gezeigt, dass die Texterkennung eine hohe Genauigkeit aufweist und das wechselnde Layout bis auf Abbildungen und Tabellen korrekt wiedergibt. Außerdem konnte festgestellt werden, dass die Netzwerkregeln mit zunehmender Größe nicht vollständig erkannt werden und mehr Fehler auftreten, womit die Stärke des Prototyps momentan auf dem Generieren von Regeln liegt, die aus wenigen Einheiten aufgebaut sind. Eine Erweiterung um das Auslesen mehrerer gleichzeitiger PDF-Dateien und die damit verbundene Kombination von INDRA-Statements verschiedener Arbeiten könnte dies künftig optimieren.

8 Datenabhängigkeit

In diesem Abschnitt werden die Dependencies bzw. Abhängigkeiten mit den entsprechenden Versionen aufgelistet. Einige der benötigten Tools sind bereits in Django und INDRA integriert und werden deshalb nicht gelistet. Folgende Software muss zur Verwendung installiert werden:

- python 3.6+ (3.9.2), `https://www.python.org`
- Django 3.2.4, `https://www.djangoproject.com`
- INDRA 1.20.0, `https://github.com/sorgerlab/indra`
- pdf2image 1.16.0, `https://github.com/Belval/pdf2image`
- poppler 21.09.0 `https://poppler.freedesktop.org`
- pyjnius 1.3.0, `https://github.com/kivy/pyjnius`
- Tesseract 4.1.1, `https://github.com/tesseract-ocr/tesseract`
- pytesseract 0.3.7, `https://github.com/madmaze/pytesseract`
- lokaler REACH Server: sbt 1.5.5, `https://www.scala-sbt.org/index.html` und Java Development Kit (JDK) 8 oder 11
- Datenbank: psycopg2-binary, `https://www.psycopg.org/docs/`

Literatur

[1] Kauffman S. A. u. a. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993. ISBN: 978-0-19-507951-7.

[2] Müller A. „Reverse Engineering Methoden zur Rekonstruktion von Genregulationsnetzwerken aus Genexpressionsdaten“. Diplomarbeit. Universität Leipzig, 2004.

[3] Computational Language Understanding (CLU) Lab at University of Arizona. *Reach*. 2014. URL: http://agathon.sista.arizona.edu:8080/odinweb/ (besucht am 29. 10. 2021).

[4] Gyori B.M. u. a. „From word models to executable models of signaling networks using automated assembly“. In: *Molecular Systems Biology* 13.11 (2017), S. 954. DOI: 10.15252/msb.20177651. URL: http://msb.embopress.org/content/13/11/954.

[5] Patel C., Patel A. und Patel D. „Optical character recognition by open source OCR tool tesseract: A case study“. In: *International Journal of Computer Applications* 55.10 (2012), S. 50–56. DOI: 10.5120/8794-2784.

[6] Liddy E. D. „Natural language processing“. In: *Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc.* (2001).

[7] Marani F. *Introduction to Django. In: Practical Django 2 and Channels 2*. Apress, Berkeley, CA, 2019. DOI: 10.1007/978-1-4842-4099-1_1.

[8] A. Fauré u. a. „Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle“. In: *Bioinformatics* 22.14 (Juli 2006), e124–e131. DOI: 10.1093/bioinformatics/btl210.

[9] B. M. Gyori und J. A. Bachman. *INDRA documentation*. 2021. URL: https://indra.readthedocs.io/en/stable/index.html (besucht am 27. 10. 2021).

[10] Hußmann H. *Zur formalen Beschreibung der funktionalen Anforderungen an ein Informationssystem*. Techn. Ber. Inst. für Informatik an der Technischen Universität München, 1993.

[11] Maucher M. Hopfensitz M. Müssel C. und Kestler H. A. „Attractors in Boolean networks: a tutorial“. In: *Computational Statistics* 28.1 (2013), S. 19–36. DOI: 10.1007/s00180-012-0324-2.

[12] Albert I. *Boolean Network Modeling*. 2014. URL: https://github.com/ialbert/booleannet (besucht am 24. 10. 2021).

[13] N. Ali M. Isheawy und H. Hasan. „Optical Character Recognition (OCR) System“. In: *IOSR Journal of Computer Engineering (IOSR-JCE)* 17 (2 2015), S. 22–26.

[14] Bergsmann J. „Nicht-funktionale Anforderungen“. In: *Quality-Knowledgeletter (Software Quality Lab)* 5.3 (2004).

[15] Schwab J.D. und Kestler H. A. „Automatic Screening for Perturbations in Boolean Networks“. In: *Frontiers in physiology* 9 (2018), S. 431. ISSN: 1664-042X. DOI: 10.3389/fphys.2018.00431. URL: https://www.frontiersin.org/article/10.3389/fphys.2018.00431.

[16] J. Kroos. „Boolesche Netzwerke mit zufälligen Regeln: ein regelbasiertes Modell mit zufälligen Regeln für regulatorische genetische Netzwerke“. Bachelor- und Masterseminar zur WT: Mathematische Biologie. Westfälischen Wilhelms-Universität Münster, 2012.

[17] JGraph Ltd. *diagrams.net*. 2005. URL: www.draw.io (besucht am 21. 10. 2021).

[18] A. Luna u. a. „PaxtoolsR: pathway analysis in R using Pathway Commons“. In: *Bioinformatics* 32.8 (2016), S. 1262–1264. DOI: 10.1093/bioinformatics/btv733.

[19] Smith R. „An overview of the Tesseract OCR engine“. In: *Ninth international conference on document analysis and recognition (ICDAR 2007)*. Bd. 2. IEEE. 2007, S. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[20] Tristan Radtke. *60tools*. 2013. URL: https://60tools.com/de/tool/text-comparison (besucht am 27. 10. 2021).

[21] S. V. Rice, F. R. Jenkins und T. A. Nartker. *The fourth annual test of OCR accuracy*. Technical Report. Information Science Research Institute (ISRI), 1995.

[22] S. V. Rice, J. Kanai und T. A. Nartker. *An evaluation of OCR accuracy*. Techn. Ber. University of Nevada, Las Vegas: Information Science Research Institute (ISRI), 1993.

[23] M. Ritter und C. Winterbottom. *UX for the Web: Build websites for user experience and usability*. Packt Publishing Ltd, 2017. ISBN: 1787128474.

[24] D. Rubio. *Beginning Django: Web Application Development and Deployment with Python*. Springer, 2017. ISBN: 1484227867. DOI: 10.1007/978-1-4842-2787-9.

[25] Julian Schwab u. a. „ViSiBooL—visualization and simulation of Boolean networks with temporal constraints“. In: *Bioinformatics* 33.4 (Nov. 2016), S. 601–604. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw661. eprint: https://academic.oup.com/bioinformatics/article-pdf/33/4/601/25146697/btw661.pdf. URL: https://doi.org/10.1093/bioinformatics/btw661.

[26] L. Siegle u. a. „A Boolean network of the crosstalk between IGF and Wnt signaling in aging satellite cells“. In: *PLoS One* 13.3 (2018), e0195126. DOI: 10.1371/journal.pone.0195126.

[27] S. D. Werle u. a. „Unraveling the molecular tumor-promoting regulation of cofilin-1 in pancreatic cancer“. In: *Cancers* 13.4 (2021). ISSN: 2072-6694. DOI: 10.3390/cancers13040725. URL: https://www.mdpi.com/2072-6694/13/4/725.

[28] Zygomatic. *Diagrammwerkzeug*. 2021. URL: www.diagrammerstellen.de (besucht am 26. 10. 2021).

[29] Smith R. u.a. *Tesseract Open Source OCR Engine*. 2005. URL: https://github.com/tesseract-ocr/tesseract (besucht am 25. 10. 2021).