
Bachelorarbeit

im Studiengang Computerlinguistik

an der Ludwig- Maximilians- Universität München

Fakultät für Sprach- und Literaturwissenschaften

# Regelbasierte Extraktion von Informationen zu Filmen aus der Wikipedia

vorgelegt von
Thomas Grieb

| | |
|---|---|
| Betreuer: | Prof. Dr. Klaus Schulz |
| Prüfer: | Prof. Dr. Klaus Schulz |
| Bearbeitungszeitraum: | 23. September - 02. Dezember 2019 |

**Selbstständigkeitserklärung**

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, den 02. Dezember 2019

.....
Thomas Grieb

# Abstract

Die vorgelegte Arbeit aus dem Feld der Computerlinguistik untersucht Methoden zur regelbasierten Verarbeitung eines Wikipedia-Textkorpus zum Zweck der Extraktion von relationalen Angaben aus enthaltenen Artikeln zu Filmen. Als Relationale Angaben gelten dabei die Informationen Filmtitel, Genre, Filmtyp, Erscheinungsjahr sowie Produktionsland. Dazu kommen Methoden aus den Bereichen der Textklassifikation sowie Informationsextraktion zum Einsatz, bei denen im Voraus festgelegte Regeln die Grundlage der Verarbeitung bilden. Diese Regeln werden mit Hilfe von manuell gesammeltem Fachwissen zum Themenbereich Film erstellt und anhand der Betrachtung von bekannten Film-Artikeln weiterentwickelt. Die Ergebnisse eines Vergleichs von Relationsmengen unbetrachteter Artikel mit einem unabhängigen Goldstandard ergeben F1-Scores zwischen 75 und 99%, je nach Relationskategorie. Damit wird gezeigt, dass regelbasierte Verfahren im Bereich der Informationsextraktion auch heute noch eine gute Alternative zu machine-learning-Algrithmen darstellen können. Zusätzlich werden am Ende noch Schwachstellen sowie Möglichkeiten zur Verbesserung der verwendeten Methoden genannt und ein Ausblick auf mögliche weitere Arbeit am Thema gegeben.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Daten waren schon immer ein wichtiges Werkzeug der Menschheit - Von der primitiven Höhlenmalerei bis hin zu den hochkomplexen Personendaten moderner Regierungen und Großkonzerne - Alle haben den Zweck, eine bestimmte Information zu speichern. Je größer die Menge an gesammelten Daten jedoch wird, desto schwieriger gestaltet sich auch die Differenzierung zwischen relevanter und irrelevanter Information. Folglich werden Lösungen zur Sichtung und Strukturierung der enormen Datenmengen benötigt, um an die Kerninformationen zu gelangen und deren Nutzen voll auszuschöpfen.

Besonders relevant werden solche Lösungen bei der Suche nach Informationen im Internet. Zwar lässt sich davon ausgehen, dass irgendwo innerhalb der Gesamtheit an Daten, die das Internet bilden, die gesuchte Angabe enthalten ist, die Suche per Hand würde für einen Menschen jedoch ein praktisch unmögliches Unterfangen darstellen. An dieser Stelle kommen automatische Prozesse ins Spiel, beispielsweise Suchmaschinen, die Webseiten für den Nutzer nach der gewünschten Information durchsuchen. Vorgänge dieser Art fallen in den computerlinguistischen Bereich *Information Retrieval* und sind für viele Menschen längst Teil des Alltags.

Möchten Nutzer, oder weitergedacht Anwendungen, nun jedoch an große Datenmengen zu spezifischen Themenbereichen gelangen, steigt der benötigte Zeitaufwand schnell an. Hier hilft der Bereich der *Information Extraction*: 'Information Extraction (IE) is the name given to any process which selectively structures and combines data which is found, explicitly stated or implied, in one or more texts.' [8] Verfahren dieser Art helfen bei der automatisierten Verarbeitung großer Datenmengen, meist mit dem Ziel der Reduzierung auf einige enthaltene Kerndaten.

Bei der Implementierung solcher Verfahren wird klassisch zwischen zwei Varianten unterschieden: modernes *machine learning*, zu deutsch auch maschinelles Lernen, oder klassische regelbasierte Verfahren. Während maschinelles Lernen heutzutage sehr vielversprechende Ergebnisse erbringt, hängt die erreichte Qualität stark von der Menge der verwendeten Trainingsdaten ab. Da die manuelle Annotation solcher Trainingsdaten viel Zeit in Anspruch nimmt, eignet sich dieser Ansatz nur selten für kleinere Projekte. Hier kommen regelbasierte Systeme ins Spiel: Um die Ergebnisse, die auf manuell erstellen Regeln basierende Systeme erbringen, mit einem machine-learning-System zu erreichen, wird deutlich mehr Zeit benötigt. Im Falle der Informationsextraktion bildet das benötigte Fachwissen die Grundlage solcher Regeln. So argumentieren Waltl u. a. (2018): 'Even if rule-based IE systems require the manual implementation of rules, the manual labour implied directly translates into the quality of the rules, while ML techniques require very deep theoretical knowledge and experimentation to maximize their efficiency.'[7]

Nun werden noch Rohdaten benötigt, meist ein Textkorpus, auf den die gewählte Technik angewendet werden soll. Zwar bestünde die Möglichkeit, Methoden der Information Retrieval und Information Extraktion zu verbinden und als ersten Schritt im Internet automatisiert nach Daten zu suchen (*Webcrawling*), jedoch entstehen dabei diverse neue Probleme, wie zum Beispiel die Frage nach der Zuverlässigkeit der verwendeten Webseiten. Eine einfache Lösung hierfür stellt die Online-Enzyklopädie Wikipedia dar. Wikipedia bietet enorme Datenmengen in maschinenlesbarem Format, mit zumindest abschätzbarer Zuverlässigkeit der Informationen und einer groben Struktur, die die Position einer spezifischen Angaben innerhalb eines Texts erahnen lässt.

Da somit bekannt ist, woher die Daten kommen und mit welcher Methode sie verarbeitet

werden sollen, bleibt noch die Klärung der letzten Frage: Welche Daten, das heißt welches Thema ist von Interesse? Hier hat sich der Autor für den Themenbereich Filme entschieden, da Filme durch ihre enorme Varianz an Titeln und Details eine besondere Herausforderung bei der Annotation und Extraktion mit sich bringen. Weiterhin besitzt der Autor aufgrund von persönlichem Interesse am Medium spezielles Fachwissen zum Thema Film und verfügt damit über die Fähigkeit, eigenhändig sinnvolle Regeln zu erstellen und die richtigen Daten dafür auszuwählen.

## 1.2 Themendefinition

Die vorliegende Arbeit behandelt aus den genannten Gründen die regelbasierte Extraktion und Verarbeitung von relationalen Informationen zum Entitätentyp „Film“ aus den Einleitungen von zugehörigen Wikipedia-Artikeln. Als Einleitungen werden dabei im Folgenden die ersten 300 Zeichen eines Artikels angesehen. Das Ziel ist die Bildung von sogenannten relationalen Angaben, das heißt einheitlich strukturierte Relationsmengen zu jeder gefundenen Entität. Es wird für solche im Weiteren als Relation bezeichneten Elemente festgelegt, dass jede Relationsmenge aus genau fünf Angaben/Relationen besteht: Titel (des Films), Filmtyp, Genre(s), Herkunftsland/Nationalität und Erscheinungsjahr. Genauere Definitionen sind zu finden in 3.2.

Zu diesem Zweck sollen verschiedene regelbasierte Verfahren genutzt werden, um die gewünschten Informationen in den Texten zu suchen und zu extrahieren. Dabei liegt der Fokus nicht auf einer besonders allgemeinen Methode zur Informationsextraktion, sondern auf der spezifische Anwendung dieser Verfahren zur Extraktion von Filmdetails aus der Wikipedia.

Regelbasiert bedeutet im Kontext der Arbeit, dass vom Autor gesammelte Daten, sowohl aus dem Internet als auch eigenes Vorwissen, zur Umsetzung der präsentierten und teilweise im Rahmen dieser Arbeit zur Verfügung gestellten Werkzeuge genutzt werden. Die Sammlung und Strukturierung der Daten ist dabei ein wichtiger Teil des Arbeitsprozesses.

Die einzelnen Arbeitsschritte und Lösungsansätze orientieren sich dabei an den vom Betreuer bereitgestellten Bearbeitungshinweisen [20]. Diese enthalten auch eine Kurzdokumentation zur Erstellung und Anwendung der zur Verfügung gestellten Werkzeuge.

## 1.3 Übersicht

Die folgende Arbeit gliedert sich in mehrere aufeinander aufbauende Arbeitsschritte, zu denen im ersten Teil einige Voraussetzungen, theoretische Grundlagen und Erläuterungen gegeben werden. Darauf folgt der praktische Teil, also die Durchführung der geplanten Schritte und eine Evaluation der Ergebnisse mit folgender Fehleranalyse.

Voraussetzung für die Arbeit mit dem Wikipedia-Korpus ist die vorherige Kürzung der vollständigen Artikel auf die genannten 300 Zeichen. Darauf folgt die Sammlung von Daten, die als Regeln für die folgende Klassifikation und teilweise als Grundlage für einige Lexika genutzt werden. Die Klassifikation erfolgt durch ein vom Autor entwickeltes Programm zur Filterung von Nicht-Film-Artikeln. Dabei erfolgt weiterhin die Generierung von Daten zur Erstellung eines zusätzlichen Lexikons zur Annotation.

Als nächster Schritt werden weitere Daten akquiriert und, ebenso wie die bereits Vorhandenen, in die geforderte Struktur gebracht, um zur Annotation mit den zur Verfügung gestellten Werkzeugen nutzbar gemacht zu werden. Weiterhin erfolgt die Vorannotation der Texte mittels des ersten Werkzeugs, sowie die Korrektur der Annotation durch vom Autor verfasste Grammatiken. Mit der Extraktion und Strukturierung der annotieren Daten erfolgt der Abschluss des praktischen Arbeitsprozesses.

Der letzter Schritt ist eine ausführliche Auswertung der Ergebnisse anhand vorher festgelegter Kriterien und eine Fehleranalyse sowie Problembehandlung. Zuletzt werden auf

Basis der gewonnenen Erkenntnisse mögliche weitere Arbeiten am Thema vorgeschlagen.

Die Arbeit schließt mit einer Zusammenfassung der durchgeführten Arbeitsschritte, gefolgt von einem Fazit.

# 2 Forschung

## 2.1 Literatur

In der vorliegenden Arbeit mussten verschiedene computerlinguistische Probleme nacheinander gelöst werden, um in jedem Bereich weiterverarbeitbare Ergebnisse zu erhalten. Hierbei fällt besonderes Augenmerk auf die Klassifikation der Artikel, die Sammlung und Strukturierung der hierfür und zur Lexikonerstellung benötigten Daten und die Annotation sowie Extraktion der Relationen. Dies erforderte die Kombination und Abwandlung verschiedener bekannter Methoden zur Verwendung in dieser Arbeit. Zusätzlich wurden Methoden zur Evaluation der entstandenen Ergebnisse ausgewählt.

### 2.1.1 Bag-of-Words-Ansatz

*Bag-of-Words* ist ein klassisches Prinzip der Computerlinguistik und wird bereits seit geraumer Zeit zur Feature-Erstellung und Klassifikation verwendet. Es basiert auf dem Konzept des Feature-Vektors, den Joachims (1998) folgendermaßen beschreibt: 'Each distinct word $w_i$ corresponds to a feature with $TF(w_i, d)$, the number of times word $w_i$ occurs in the document $d$ , as its value.'[14] Es entsteht also, vereinfacht ausgedrückt, eine Liste aller in den betrachteten Dokumenten vorkommender einzigartiger Wörter und deren Anzahl an Vorkommen.

Bereits in dieser simplen Form ließe sich damit ein einfacher Klassifikator erstellen: Auf einige Texte eines Fachbereichs angewendet, entsteht ein Bag-of-Words-Modell mit allen Wörtern. Bereinigt man den Vektor von Stopwörtern und extrahiert die $x$ Häufigsten, erhält man ein Lexikon von für den gewählten Fachbereich besonders relevanten Wörtern und deren Häufigkeit in den untersuchten Texten, interpretierbar als Gewicht. Eine Überprüfung neuer Artikel auf die im Lexikon enthaltenen Wörter kann nun zur Klassifikation dieser genutzt werden. [18]

So verwendet Salam (2019) in seinem Paper hierzu einen *bag of concepts...*, ein Lexikon bestehend aus zwei Dateien: 'One file contains positive words, and the other file contains negative words. We assumed that each positive word has a polarity of 1, and each negative word has a polarity of -1. The general idea is to search for these concepts in the review and to calculate the sum of their polarities to determine the overall polarity of the review.'[6]

### 2.1.2 Reguläre Sprachen und Automaten

Reguläre Sprachen und endliche Automaten bilden Kernelemente der theoretischen Informatik und sind eng miteinander verbunden. So ist eine Reguläre Sprache definitiert als jede Sprache, die durch einen sogenannten endlichen Automaten beschrieben werden kann. Umgekehrt akzeptiert jeder korrekte endliche Automat nur reguläre Sprachen als Eingabe. [13]

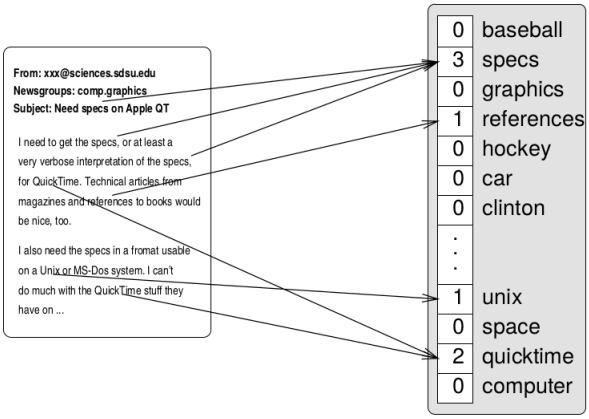


Abbildung 2.1: Beispiel für Feature Vector aus Joachims (1998)

Endliche Automaten, im Englischen *Finite-State Automata*, sind mathematisch wie folgt definiert (nach Roche, 1996):
'A finite-state automaton A is a 5-tuple $(\Sigma, Q, i, F, E)$ where

$\Sigma$ is a finite set called the alphabet,

$Q$ is a finite set of states,

$i \in Q$ is the initial state,

$F \subset Q$ is the set of final states and

$E \subset Q \times (\Sigma \cup \{\epsilon\} \times Q$ is the set of edges.'[9]

Möchte man den Automaten generalisieren, kann das Prinzip des monoidalen endlichen Automaten betrachtet werden. Die Definition ähnelt stark der oben Genannten, mit dem größten Unterschied, dass das Alphabet einem Monoiden entspricht (nach Mihov/Schulz, 2018):
'A monoidal finite-state automaton (MSA) is a tuple of the form
$\langle A = M, Q, I, F, \Delta[= E]\rangle$, where

- $\langle M = M, \circ, e\rangle$ is a monoid' [17]

Mit einem endlichen Automaten können nun reguläre Sprachen über dem Alphabet $\Sigma$ eingelesen werden. Dabei ist die Menge der Regulären Sprachen über das Alphabet $\Sigma$ folgendermaßen rekursiv definiert (nach Kaplan/Kay, 1994; Anmerkung: i kennzeichnet i-fach wiederholte Konkatenation, $\Sigma^\epsilon$ kürzt $\Sigma \cup \{\epsilon\}$ ab):

'1. The empty set and $\{a\}$ for all $a$ in $\Sigma^\epsilon$ are regular languages.

2. If $L_1$, $L_2$ and $L$ are regular languages, then so are
   $L_1 \cdot L_2 = \{xy \| x \in L_1, y \in L_2\}$ (concatenation)
   $L_1 \cup L_2$ (union)
   $L^* = \bigcup_{i=0}^{\infty} L^i$ (Kleene closure)

3. There are no other regular languages.'[15]

Theoretisch sind somit alle endlichen Sprachen auch reguläre Sprachen und können wiederum mittels eines endlichen Automaten eingelesen werden. Soll diese Theorie nun genutzt werden, um eine Zeichenfolge in einem Text zu finden, könnte manuell ein entsprechender endlicher Automat geschrieben werden. Hierbei bestünde die Möglichkeit, eine

besondere Klasse zu verwenden, beispielsweise einen deterministischen endlichen Automaten, auch DEA, oder nicht-deterministischen endlichen Automaten, auch NDEA. Den Unterschied beschreiben Hopcroft u.A (2007) folgendermaßen: '[...] „deterministic “, meaning that the automaton cannot be in more than one state at any one time, or „nondeterministic “ meaning that it may be in several states at once.'[13] Stattdessen kann jedoch auch das informatische Konzept des Regulären Ausdrucks verwendet werden.

### 2.1.3 Reguläre Ausdrücke und Finite-State Transducers

Ein regulärer Ausdruck ist einfach gesagt eine Art Programmiersprache für reguläre Sprachen, auch interpretierbar als benutzerfreundlichere Alternative zur NDEA-Notation[13]. Die hierfür gültigen Regeln und Operatoren sind genau definiert und unendlich kombinierbar, somit lässt sich mit diesem Prinzip ebenso jeder theoretisch denkbare endliche Ausdruck oder sogar Text beschreiben, sofern die verwendeten Zeichen unterstützt werden.

Reguläre Ausdrücke stellen ein wichtiges Werkzeug zur Textverarbeitung dar, so schreiben Wang u. a. (2019): 'Regular expressions are employed in data validation, classification, and extraction [...] A regular expression is a language to describe a set of strings it can match [...]'[23]. Heute unterstützen sowohl viele höhere Programmiersprachen, wie etwa Python, die Verwendung von regulären Ausdrücken, als auch eine Vielzahl an gewöhnlichen Texteditoren, beispielsweise bei Such- und Ersetzungsoperationen.

Da reguläre Ausdrücke als programmiersprachliche Umsetzung von regulären Sprachen gelten, können sie ebenso wie diese in eine Form von endlichem Automat umgewandelt werden, wie von Karttunen u. a. (1997) erläutert: 'Languages are represented by simple automata, relations by transducers. Any regular expression can be compiled into a network that represents the corresponding language or relation.'[16] Bei dieser Form von endlichem Automat handelt es sich um die sogenannten *finite-state Transducer*, oder endliche Transduktoren/Transducer, mathematisch folgendermaßen definiert (nach Roche, 1996):

'[FST] A Finite-State Transducer is a 6-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ where:

- $\Sigma_1$ is a finite alphabet, called the input alphabet.
- $\Sigma_2$ is a finite alphabet, called the output alphabet.
- $Q$ is a finite set of states,
- $i \in Q$ is the initial state,
- $F \subset Q$ is the set of final states and
- $E \subset Q \times \Sigma_1^* \times \Sigma_2^* \times Q$ is the set of edges.' [9]

Im Unterschied zum normalen Automaten erzeugt ein so definierter Transducer jedoch auch eine Ausgabe, benötigt also ein zusätzliches Ausgabealphabet. Des Weiteren setzt der Transducer nicht reguläre Sprachen, sondern die unten definierten regulären Relationen um (nach Kaplan/Kay, 1994):

'1. The empty set and $\{a\}$ for all $a$ in $\Sigma^\epsilon \times ... \times \Sigma^\epsilon$ are regular n-relations.

2. If $R_1$, $R_2$ and $R$ are regular n-relations, then so are
   $R_1 \cdot R_2 = \{xy \| x \in R_1, y \in R_2\}$ (n-way concatenation)
   $R_1 \cup R_2$ (union)
   $R^* = \bigcup_{i=0}^{\infty} R^i$ (n-way Kleene closure)

3. There are no other regular n-relations.'[15]

Solche regulären Relationen bilden Tupel von Zeichen(folgen) der Form „{<$\epsilon$ :$\epsilon$>, <a:aa>, <aa:aaaa>“ (Transducer in Abbildung 2.2) ab, die mittels sogenannter *rewriting rules*, zu deutsch Ersetzungsregeln, festgelegt werden können. Ein spezieller Algorithmus (siehe 2.2.6) kann die Ersetzungsregeln schlussendlich in fertige Transducer umwandeln [1]. Alternativ besteht die Möglichkeit, in Handarbeit eigene Transducer zu entwickeln, um sehr spezifische Aufgaben auszuführen. Ein solcher Transducer (siehe auch 2.1) kann nun genutzt werden, um analog zum regulären Ausdruck eine Zeichenfolge zu erkennen und zu ersetzen.


Abbildung 2.2: Beispiel für Transducer [1]

### 2.1.4 Auswertungsmethoden

'Empirical evaluation plays a central role in estimating the performance of natural language processing (NLP) or information retrieval (IR) systems.' [11] Zur Auswertung der gewonnenen Daten werden in dieser Arbeit die klassischen *Performance Measures Precision*, *Recall* und *F1-Score* in an die Spezifika der Arbeit angepasster Form verwendet. Die im Folgenden demonstrierten und später genutzen Formeln und Definitionen hierzu sind Goutte/Gaussier (2005)[11] und Esuli/Sebastiani (2010)[10] entnommen.

**Confusion Matrix** Zur Berechnung der Performance Measures wird zunächst eine Einordnung der Ergebnis-Objekte in eine *Confusion Matrix* benötigt. Die Confusion Matrix, zu Deutsch auch Wahrheitstabelle, stellt die Ergebnismengen in 4 Klassen unterteilt grafisch dar: *True Positive* (TP), *False Positive* (FP), *False Negative* (FN) und *True Negative* (TN). In welchen Bereich ein Objekt fällt, hängt davon ab, ob der Algorithmus das binäre Label des Objekts richtig geraten hat. [11]

| | | Assignment $z$ | |
|---|---|---|---|
| | | + | - |
| Label $\ell$ | + | $TP$ | $FN$ |
| | - | $FP$ | $TN$ |

Abbildung 2.3: Confusion Matrix aus Goutte/Gaussier (2005)[11]

**Precision und Recall** Unter Verwendung der erhaltenen Werte lassen sich nun Precision (p) und Recall (r) folgendermaßen berechnen (nach Goutte/Gaussier, 2005)[11]

$$p = \frac{TP}{TP + FP}$$
$$r = \frac{TP}{TP + FN}$$

**F1-Score** Klassisch erfolgt die Berechnung des F-Scores aus dem '(weighted) harmonic average of precision and recall' [11], wenn ß das Gewicht beschreibt (nach Goutte/Gaussier, 2005)[11]

$$F_ß = (1 + ß^2)\frac{p \cdot r}{r + ß^2 \cdot p} = \frac{(1 + ß^2) \cdot TP}{(1 + ß^2) \cdot TP + ß^2 \cdot FN + FP}$$

Wird nun das Gewicht ß = 1 gesetzt, erhält man den F1-Score (nach Esuli/Sebastiani, 2010)[10]:

$$F_1 = \frac{2 \cdot p \cdot r}{r + p} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$$

**Anpassung** Um die Festlegung der Evaluierungsmethoden abzuschließen, bleibt noch die Anpassung der beschriebenen Formeln auf die zu erwartenden Ergebnisse. Besonders zu beachten ist dabei die Einordnung von den Ergebnismengen in die Confusion Matrix. Zunächst ist offensichtlich, dass inhaltlich korrekt erkannte Elemente in den Relationen die *True Positives* darstellen, ebenso wie korrekt als leer markierte Felder in den Relationen als *True Negatives* zu interpretieren sind. Nun wäre der nächste naheliegende Schluss, dass fälschlich leere Felder den *False Negatives* und fälschlich befüllte Felder den *False Positives* entsprechen.

Eine solche Einordnung wirft jedoch ein Problem auf: Durch die gesonderte Behandlung der Gruppe *False Element* lassen sich die klassischen Performance Measures aufgrund der Beschränkung auf binäre Labels nicht mehr auf die Ergebnisse anwenden. Eine Möglichkeit, damit umzugehen, könnte durch die seperate Analyse aller einzelnen Begriffe, wie in Cowie/Wilks (2000)[8] erläutert, realisiert werden. Da die Implementierung eines solchen Verfahrens jedoch erneut deutlich zu viel Zeit in Anspruch nähme und die resultierenden Scores in Cowie/Wilks (2000)[8] kritisch betrachtet werden, wird darauf verzichtet und stattdessen mit zwei alternativen Methoden gearbeitet, die dem Rahmen dieser Bachelorarbeit angemessen sind.

**Reguläre Berechnung ohne FE** Der erste Teil der Evaluation untersucht mittels der klassischen Formeln lediglich, wie häufig fälschlich Werte fehlen oder nicht fehlen. Hierbei werden die *False Elements* vollständig ignoriert.

**Formel zur POS-Evaluation** Um Precision, Recall und F-Score bei Part-of-Speech-Tagging zu errechnen, kommen veränderte Formeln zum Einsatz (nach Hager, 2011[12] und Paroubek, 2007[19]), welche sich analog folgendermaßen auf die Ergebnisse anwenden lassen:

$$p = \frac{\text{Number of correct Relations in extracted data}}{\text{Number of total Relations in extracted data}} = \frac{TP}{TP + FP + FE}$$

$$r = \frac{\text{Number of correct Relations in extracted data}}{\text{Number of correct Relations in gold data}} = \frac{TP}{TP + FN + FE}$$

$$F_1 = \frac{2 \cdot p \cdot r}{p + r}$$

Hier werden zur Kalkulation der Precision die *True Positives* durch die gesamte Anzahl an automatisch extrahierten nicht-leeren Relationen geteilt. Die automatisch extrahierten nicht-leeren Relationen lassen sich durch die Summe alle *True Positives*, *False Positives* und *False Elements* errechnen. Dahingegen entsteht der Recall aus den *True Positives* geteilt durch die Summe aller nicht-leeren Relationen im Goldstandard-Datensatz. Diese errechnen sich aus der Summe der *True Positives*, *False Negatives* und *False Elements*. Die Berechnung des F1-Scores erfolgt wie bekannt.

## 2.2 Technische Voraussetzungen

In diesem Abschnitt werden die technischen Voraussetzungen genauer beschrieben, unter denen die Ergebnisse erarbeitet wurden. Dies beinhaltet das vorrangige Arbeitssystem, verwendete Programmiersprachen und genutzte Tools.

### 2.2.1 Arbeitsumgebung

Vorrangiges Arbeitssystem war ein AMD Ryzen 5 1600 basierter Desktop-PC, auf dem Ubuntu 18.04 (auf Solid State Drive) zum Einsatz kommt. Der Prozessor verfügt über 6 Kerne mit einem maximalen Takt von je 3.20GHz und an Arbeitsspeicher stehen 16GB DDR4 zur Verfügung. Für alle Tests wurde ein eigens mit dem in Python integrierten Tool „venv“ erstelltes *virtual environment* genutzt. Im Weiteren beziehen sich etwaige Hinweise auf Laufzeiten der Prozesse auf dieses System.

### 2.2.2 Python

Bei Python handelt es sich um eine Interpreter-Programmiersprache, die 1991 von Guido van Rossum am *Centrum Wiskunde & Informatica* (Amsterdam) entwickelt wurde. Version 3.0 erschien am 3. Dezember 2008.

Aufgrund der allgemein großen Popularität für linguistische Anwendungen als auch wegen persönlichen Vorkenntnissen des Autors wurde Python in der Version 3.7 als primäre Programmiersprache, sofern anwendbar, verwendet. Des Weiteren bietet Python eine einzigartige Bibliothek an Tools zur Sprachverarbeitung.

Aus *NLP with Python*: 'Python is a simple yet powerful programming language with excellent functionality for processing linguistic data.' [21]

**NLTK** Das *Natural Language Toolkit* für Python ist ein open-source Projekt unter der Apache Lizenz, dessen Entwicklung 2001 unter Edward Loper und Steven Bird an der *University of Pennsylvania* begann. Ursprünglich als Hilfswerkzeug für Lehrveranstaltungen vorgesehen, findet die ständig weiterentwickelte Sammlung von Bibliotheken auch häufig in der Forschung Einsatz.

In diesem Projekt kommt NLTK in der Version 3.4.5 hauptsächlich zur Tokenisierung als Zwischenschritt bei Parsing und Klassifikation zum Einsatz.

### 2.2.3 C(M)-Sprache

Bei der C(M)-Sprache handelt es sich um die Implementierung der Transducer-Techniken als funktionale, deklarative Programmiersprache. Während die Sprache an sich Gemeinsamkeiten mit Haskell hat, kommt eine standardmäßige mathematische Notation ähnlich der in SETL genutzten zur Verwendung. Mihov und Schulz (2018) beschreiben den Kerngedanken hinter der Funktionsweise als: 'In practice, we just formally describe the kind of mathematical object we want to obtain.' Der zugehörige C(M)-Compiler übersetzt ein gültiges Programm in C-Code, das nach der Kompilierung ausgeführt werden kann. [17]

In der vorliegenden Arbeit wurde die Sprache zur Entwicklung zweier Transducer verwendet, die die bestehende Vorannotierung zur endgültigen Annotation umwandeln. Genaueres hierzu in 3.3.

### 2.2.4 Versionierung

Der Autor entschied sich für Git als Versionierungstool, da hierfür der GitLab2-Server des Instituts für Informatik, erreichbar unter `https://gitlab2.cip.ifi.lmu.de/`, genutzt werden konnte. Damit ist ein besserer Datenschutz sichergestellt als dies beispielsweise mit dem in privater Hand befindlichen GitHub der Fall wäre.

### 2.2.5 Dateiformate

**Plaintext** Sowohl zur Speicherung der Originaldaten als auch für mehrere Zwischenstadien der Datenverarbeitung und der Ergebnisdatei kam *plaintext* („.txt“) zum Einsatz, da dieses Format einfach und universell verarbeitbar und für simples zeilenweises Einlesen völlig ausreichend ist.

**JSON** *JSON* ist ein auf JavaScript basierendes Format zum Datenaustausch, das häufig in Webanwendungen, aber auch einer Vielzahl an anderen Projekten in diversen Programmiersprachen Verwendung findet.

Wenn die Speicherung komplexerer Daten in einer bestimmten Struktur gefragt war, wurde das JSON-Format verwendet, so zum Beispiel bei den Genredaten, Schlagwort-Score Paaren und Artikel-Dateien. Dies ermöglicht gute Lesbarkeit für Menschen als auch verlässliche maschinelle Verarbeitbarkeit von verschachtelten Datenkonstrukten.

### 2.2.6 Verwendete Tools

Es wurde eine Reihe an Tools und Python-Libraries verwendet, die Wichtigsten werden im Folgenden kurz beschrieben.

**plmlm-Technik** Das Programm, das die plmlm-Technik anwendet, wandelt ein beliebiges Ersetzungsregel-Wörterbuch in einen fertigen Transducer um, der einen beliebigen Text als Eingabe annimmt. Hierbei ersetzt der Transducer jedes vollständige Vorkommen des auf der linken Seite stehenden Ausdrucks durch die Entsprechung auf der rechten Seite. Das Verfahren wurde genutzt, um die Ersetzung der so im Text vorfindbaren Zeichenkette durch die markierte und gegebenenfalls normalisierte Version des Ausdrucks zu erreichen (siehe 3.2). Das Programm wurde dem Autor für diese Arbeit zur Verfügung gestellt.

**REGEX** Reguläre Ausdrücke sind ein wichtiger Bestandteil der vorliegenden Arbeit und wurden mit Hilfe der in Python integrierten *re*-Library innerhalb des Programmcodes und an mehreren Stellen außerhalb zur Vorformatierung von Daten aus Drittquellen verwendet.

**TreeTagger** Der von Prof. Schmid entwickelte TreeTagger (`https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`) in seiner deutschen Version sollte in Verbindung mit dem von Laurent Pointal entwickelten TreeTaggerWrapper (`https://treetaggerwrapper.readthedocs.io/en/latest/`) beim Parsing genutzt werden, um POS-Tagging zu realisieren. Dieser Ansatz wurde allerdings aufgrund der hohen Laufzeit und des geringen Mehrwerts wieder verworfen. Genaueres dazu im Abschnitt 2.3.2.

## 2.3 Wikipedia-Korpus

Im Folgenden wird der verwendete Korpus und dessen Anpassung genauer beschrieben.

### 2.3.1 Allgemeine Informationen

Als Grundlage für die Arbeit wurde dem Autor von Prof. Schulz ein bereits vorformatierter Wikipedia-Dump (Jahr unbekannt) zur Verfügung gestellt. Sämtliche Formatierungen waren bereits bereinigt und die gesamten Artikeltexte liegen im plaintext-Format vor. Artikelanfänge und -enden werden durch spezielle Markups gekennzeichnet:

```
<doc id="1" url="http://de.wikipedia.org/wiki?curid=1" title="Alan Smithee">
[Artikeltitel]
[Artikeltext]
</doc>
```

Des Weiteren existieren einige inhaltliche Markups, für die jedoch leider keine Erklärungen vorliegen. Ein vermutlich ungewolltes Markup besteht in Form der Ersetzung des &-Zeichens durch die HTML-Formatierung „&“.

Der Korpus ist auf 163 einzelne .txt-Dateien von etwa 25MB Größe aufgeteilt und insgesamt etwa vier Gigabyte groß.

### 2.3.2 Korpusvorbereitung

Da für die folgende Arbeit nur die Artikeltitel und Einleitungstexte relevant sind, stellte der erste Schritt die Bereinigung und Kürzung des vorliegenden Text-Korpus dar. Um verbliebene HTML-Mark-Ups und überflüssigen Text zu entfernen, sollte ein Parser zum Einsatz kommen, der zeilenweise über jede Datei im Zielverzeichnis iteriert. Wird ein Artikelstart-Mark-Up gefunden, wird die nächste Zeile als Titel gespeichert und weitergelesen. Enthält der Artikel lediglich ein „_NOTOC_“-Mark-Up, wird der Artikel ignoriert, da kein relevanter Inhalt existiert.

An diesem Punkt hätte für jede Zeile überprüft werden sollen, ob sie eine Überschrift darstellt, indem die Anzahl an Token und die Existenz einer Verbalphrase kontrolliert wird. Hierzu erfolgt die Tokenisierung und Übergabe des ersten Satzes an einen durch den TreeTagger realisierten POS-Tagger. Bei über 10 Token in einer Zeile oder dem Fehlen einer Verbalphrase liegt eine Überschrift vor und alle weiteren Zeilen werden übersprungen, bis es zum Auftreten des nächsten Artikel-Start-Mark-Up kommt.

Zwar erbringt dieses Verfahren sehr vielversprechende Ergebnisse, die Laufzeit ist jedoch durch die Analyse jedes Satzes inakzeptabel hoch. In einer Stunde schafft der Parser mit dem beschriebenen Verfahren nur etwa zwei bis drei Dateien, für den gesamten Datensatz stünde also eine Laufzeit von über 50 Stunden zu erwarten. Des Weiteren fallen die geschrumpften Dateien erneut sehr groß aus, da die Einleitungstexte mancher Artikel 1000 Zeichen überschreiten.

Als Konsequenz aus den beschriebenen Problemen kommt deshalb eine simplere Methode zum Einsatz. Statt einer Überprüfung jeder einzelnen Zeile werden lediglich die ersten maximal 400 Zeichen des Artikels nach dem Titel extrahiert. Für die spätere Betrachtung werden zwar nur die ersten 300 relevant sein, damit ist aber für ausreichend Überschuss gesorgt, sollten weitere Bereinigungen notwendig sein.

Die Speicherung der gewonnenen Daten erfolgt in einer JSON-Datei mit dem Titel als Key und dem Text als zugehörigem Value.

```
"Alan Smithee": "Alan Smithee steht als Pseudonym für [...]
```

Mit dem zweiten Verfahren liegt die Laufzeit bei unter fünf Minuten und die Größe der meisten erhaltenen Dateien bei nicht mehr als fünf Megabyte (Reduktion um etwa 80%). Der mittels eines Shell-Scripts ermittelte Umfang beläuft sich auf 1.536.552 gekürzte, nicht-leere Artikel.

## 2.4 Datensammlung

Als weitere Vorbereitung für die Arbeit mit dem Korpus und den darin enthaltenen Film-Artikeln war die Sammlung und Strukturierung von zusätzlichen Daten aus dem Themenbereich Film nötig.

### 2.4.1 Genrebezeichner-Datensatz

Sowohl für die Klassifikation als auch die Annotation werden möglichst umfassende Informationen zu existierenden deutschen Filmgenre-Bezeichnungen benötigt. Diese sind zum Teil aus der aktuellen Wikipedia [4] [5] entnommen, zum Teil bei Auftreten nachgetragen und manuell als JSON-Datei nach dem unten zu sehenden Schema strukturiert worden. Die übergeordneten Hauptgenres dienen dabei als Key, während die zugehörigen Values ein Dictionary enthalten, in dem entsprechend benannte Keys auf Listen mit den zugehörigen Subgenres und möglichen Alternativbezeichnern zeigen.

```
"Abenteuerfilm": {
    "Subgenres": [
      "Sandalenfilm",
      "Ritterfilm",
      "Piratenfilm",
      "Mantel- und Degenfilm"
    ],
    "Alt": [
      "Abenteuer"
    ]
  }
```

Zwar ist es nahezu unmöglich, jedes existente Genre zu finden, aber mit etwa 150 verschiedenen Genres und zusätzlichen Alternativ-Bezeichnungen ist ein ausreichend breites Feld abgedeckt.

### 2.4.2 Schlagwort-Lexikon

Da die Klassifikation nur mit Hilfe von Genrebezeichnern keine zufriedenstellenden Ergebnisse erbrachte, bot sich die Erstellung eines Schlagwort-Lexikons an. Diese Methode orientiert sich am bereits erläuterten „bag-of-words“-Prinzip. Da dem Autor jedoch keine vorannotierten Texte aus dem Bereich Film zur Verfügung standen und die benötigten Ressourcen fehlten, eine solche Annotation zu beauftragen, musste die Erstellung eines vereinfachten bag-of-words-Modells manuell erfolgen.

Dazu wurde mittels eigenem Wissen und stichprobenartigem Lesen entsprechender Artikel eine JSON-Datei erstellt, die diverse deutschsprachige, für Film-Artikel auf Wikipedia besonders typische oder untypische Ausdrücke enthält. Dabei erhält jedes Wort einen Score zwischen $-2.0$ und $+2.0$, der die geschätzte Relevanz widerspiegelt. Auch hier stellt die Struktur eine Gruppierung in Kategorien dar: Die Keys benennen die Kategorie und verweisen auf Dictionaries, welche das Schlagwort als Key und den Score als Value speichern.

```
"Orte": {
    "Kino": 0.8,
    "Kinos": 0.8
}
```

Durch die Aufnahme einer solchen Approximation des bag-of-words-Modells konnten die Ergebnisse des Klassifikators deutlich verbessert werden.

# 3 Praktische Arbeit

Die praktische Arbeit umfasst die Klassifikation der Rohdaten, Erstellung der Annotationslexika sowie Annotation der Daten, die Informationsextraktion/Relationsbildung und zuletzt die Evaluation sowie Fehleranalyse und mögliche zukünftige Arbeiten.

## 3.1 Klassifikation nach Relevanz

Die erste Aufgabe nach der Kürzung und Strukturierung des Korpus ist das Finden von für die Arbeit relevanten Einträgen, also aller Artikel, die einen spezifischen Film beschreiben. Dazu wurde ein vereinfachter regelbasierter Klassifikator geschrieben, der die im voraus gesammelten Daten zu Schlagwörtern und Genres zur Klassifikation nutzt.

### 3.1.1 Klassifikator

Die Entwicklung und Anwendung eines machine-learning-Algorithmus zur Klassifikation würde den Umfang dieser Arbeit deutlich überschreiten, deshalb entschied sich der Autor für die Erstellung eines eigenen regelbasierten Klassifikatormodells, das vier Komponenten eines jeden Artikels überprüft und jeweils einen Wert (default = 0) zurückgibt:

**Inhalt des Artikeltitels:**

- Enthält der Titel den Film-Kennzeichner "(Film)"? → Positiver Wert
- Entspricht der Titel einem bekannten Genre-Bezeichner? Wenn ja, weist dies auf einen Genre-Artikel hin. → Sehr stark negativer Wert

**Schlagwörter im Text:**

- Welche bekannten Schlagwörter sind wie oft im Text vertreten? Schlagwörter sind für Filmartikel besonders typische oder untypische Wörter, die über einen geschätzten Likelihood-Score verfügen. → Summe aller Scores

**Genrebezeichner im Text:**

- Ist ein bekannter Genrebezeichner im Text vertreten? Die meisten Filmartikel enthalten einen Solchen. Gültige Genrebezeichner wären etwa „Komödie“ oder „Abenteuer“ → Positiver Wert
- Ist der Genrebezeichner eindeutig? Ein eindeutiger Bezeichner enthält den String „Film“/„film“ (=„Abenteuerfilm“), da gewisse Genrebezeichner auch zum Beispiel für Bücher gelten.
  → Stark positiver Wert

**Jahreszahlen im Text:**

- Ist ein Geburtsjahr im Text vertreten? Schauspieler-Artikel sind nach Anwendung der bisherigen Kriterien noch die häufigste Fehlklassifikation, enthalten aber im Gegensatz zu Film-Artikeln sehr oft ein Geburtsdatum in der Form „(*[date] [...])“.
  → Negativer Wert

- Ist eine allein stehende Jahreszahl ab 1800 vorhanden? Filme enthalten häufig ein Erscheinungsdatum in der Einleitung, das nicht vor dem 19. Jahrhundert sein kann. → Leicht positiver Wert

Diese Überprüfungen sind alle als Funktionen realisiert, die mit Titel oder Text eines Artikels aufgerufen werden und einen Wert zurückgeben, der für die schlussendliche Score-Berechnung genutzt wird.
**Formel:**

$$[titlescore] + [genrescore] + [schlagwortscore] + ([actorscore] * 2) = score$$

Abschließend iteriert das Programm über alle Einleitungsdateien und wendet das beschriebene Verfahren für jeden Artikel an. Wird ein Text als Film-Artikel klassifiziert, kopiert der Klassifikator das gesamte Objekt in eine neue JSON-Datei und den Titel zusätzlich in eine separate Text-Datei, die später zur Lexikonerstellung benötigt wird.

### 3.1.2 Ergebnis & Manuelle Korrekturen

Das Ergebnis der Klassifikation ist eine JSON-Datei mit allen Film-Artikeln, gegliedert, analog zu den Einleitungsdateien, nach folgendem Schema: Der Artikeltitel ist der Key, der Text stellt den Value dar. Zusätzlich entsteht, wie bereits beschrieben, eine separate Titelliste. Die entstandenen Dateien enthalten 21.327 Artikel, wohingegen der ursprüngliche Korpus 1.536.552 Artikel enthielt. Folglich klassifiziert das Programm 1,3880% aller Artikel als Film-Artikel.

Auch wenn der entwickelte Klassifikator für die folgenden Arbeitsschritte ausreichende Ergebnisse bietet, bot sich trotzdem noch zusätzliches manuelles Säubern der entstandenen Titelliste an, um später ein besseres Titellexikon zu erhalten. Hierzu kam wieder Suchen und Ersetzten mit diversen Regulären Ausdrücken zum Einsatz.

Welche regulären Ausdrücke hierbei genau verwendet worden sind, lässt sich einer zusätzlichen Datei („cleanupregex.txt“) entnehmen. Weitestgehend lag der Fokus auf dem Entfernen verschiedener Filmpreise und deren Verleihungen, Filmfestivals, Kinos, Studios und mit abweichenden Tags, beispielsweise „(Roman)“, markierten Artikeln. Hier bestünde die Möglichkeit, im Klassifikator-Programm noch genauer auf die genannten Ausnahmen einzugehen, um eine höhere Automatisierung des Vorgangs zu erreichen. Auf eine solche Programmverbesserung wird an dieser Stelle jedoch verzichtet, da die Arbeit den gesammten Verarbeitungsvorgang untersucht, nicht die Entwicklung eines möglichst guten Klassifikators.

Nach diesem nachträglichen Säubern enthielt die Titelliste noch 20.430 Einträge. Hieraus lässt sich der Schluss ableiten, dass mindestens 4,2059% der vom Klassifikator als Film eingeordneten Artikel *False Positives* darstellen.

In der weiteren Arbeit werden die ersten 998 Einträge der automatisch klassifizierten JSON-Datei als Entwicklungsdatensatz angesehen, während die Einträge 999 und aufwärts soweit wie möglich unbetrachtet bleiben, um teilweise als Testdatensatz genutzt zu werden. Genaueres hierzu in Abschnitt 3.5. Zwar beziehen sich die Korrekturen der Titelliste auf alle Einträge, dies war aber zur Erstellung des Titellexikons notwendig und wurde größtenteils durch die Verwendung der regulären Ausdrücke generalisiert, ohne über die Entwicklungsdaten hinaus per Hand einzelne Einträge zu entfernen.

## 3.2 Annotationslexika

Zur Annotation der gekürzten Daten finden sogenannte Rewrite-Wörterbücher Verwendung. Dabei handelt es sich um eine Art Lexikon, bei dem auf der linken Seite der zu ersetzende Begriff, auf der rechten Seite durch einen Tab getrennt die passende Entsprechung steht, in diesem Fall der Begriff oder eine Normalisierung mit Mark-Ups. Alle Lexika verwenden nach Vorgabe das Dateiformat „.dic“.

Da logischerweise nur so viel Information extrahiert werden kann, wie vorher erfolgreich annotiert worden ist, steht und fällt die in dieser Arbeit untersuchte Methode zur Informationsextraktion mit der Qualität der verwendeten Annotationslexika. Aus diesem Grund wurde hier besonderes Augenmerk auf eine hohe Vollständigkeit gelegt.

**Beispiele:**

```
englischer <NAT>England</NAT>
```

```
Dokumentarfilm <GENRE>Dokumentarfilm</GENRE>
```

Im Folgenden wird eine kurze Erklärung zu jeder Relation, dem zugehörigen Lexikon und dessen Erstellung gegeben.

### 3.2.1 Titellexikon

Die erste und wohl wichtigste Relation stellt der Titel eines Films dar.

Filmtitel und andere Bezeichnungen für Kunst stellen die automatische Informationsextraktion vor ein besonderes Problem: Sie sind nicht an sprachliche Regeln gebunden, bieten häufig keine Herleitungsmöglichkeiten und können sämtlichen Sprachen entstammen, unter Umständen sogar fiktionalen. So ist der Titel „Gattaca“ ein Neologismus, „Jackie Brown“ ein Eigenname und „Wag the Dog – Wenn der Schwanz mit dem Hund wedelt“ eine Satzkombination aus verschiedenen Sprachen. Da die automatisierte Erkennung folglich ausgesprochen komplex ist und somit für diese Arbeit nicht in Frage kommt, musste eine alternative Lösung gefunden werden.

Eine weitere Möglichkeit, die vor allem zur Anwendung auf unstrukturierte Texte zu empfehlen ist, wäre die Analyse von Satzmustern. So lässt sich annehmen, dass in einem Satz mit Grundstruktur nach dem Schema „-Satzbeginn- x [optionale Objekte] ist ein [optionale Objekte] [beliebiger Filmtyp] [optionale Objekte] -Satzende-“ die Struktur an Position x den Filmtitel darstellt. Leider ist die mögliche Varianz eines solchen Satzes enorm. So wären auch die Satzschemas „x ist eine Produktion ...“ (vereinfacht) oder „Die Fortsetzung zu x, x2, gewann 4 Oscars.“ denkbar. Genaueres hierzu in 3.8.

Die dritte Option ist die Verwendung eines vordefinierten Titellexikons, das diverse Filmtitel enthält. Das Format der vorliegenden Daten ermöglichte die Anfertigung einer Liste aller enthaltenen Filme, sofern korrekt erkannt, im Zuge der Klassifikation, deshalb bot sich diese Methode an. Das gewünschte Rewrite-Wörterbuch wird automatisiert auf Basis der manuell bereinigten Artikelliste (Siehe 3.1.2) von einem Python-Programm generiert. Wikipedia-eigene Mark-Ups, wie die Markierung von ambigen Titeln mit „(Film)“ oder „([Jahreszahl])“, sowie dadurch entstehende Dopplungen wurden entfernt.

**Mark-Up:** <TITLE>[Begriff]</TITLE>
**Beispiel:** Tron (Film) → Tron <TITLE>Tron</TITLE>

### 3.2.2 Filmtypenlexikon

Bei der zweiten Relation handelt es sich um eine Ergänzung, die besonders dann wichtig wird, wenn kein Genre-Begriff gegeben ist. So ließe in Absenz eines Genres im Artikel beispielsweise der Typ „Kurzfilm“ wenigstens eine wage Ahnung darüber zu, um was es sich bei dem Film handeln könnte.

Zur Differenzierung zwischen verschiedenen Filmtypen, wie zum Beispiel „Kurzfilm“ und „Spielfilm“, erfolgte die manuelle Erstellung einer Liste mit derartigen Bezeichnern. Zugleich kann das zusätzliche Tagging von Filmtypen zur Bereinigung von bereits getaggten Genres genutzt werden (Siehe 3.3.3). Jeder der Filmtypen liegt sowohl mit erstem Buchstaben großgeschrieben, als auch kleingeschrieben, vor.

**Mark-Up:** <FILMTYPE>[Begriff]</FILMTYPE>

### 3.2.3 Genrelexikon

Wenn auch die nach dem Titel vermutlich entscheidendste Relation, kommen Genrebezeichner in einer Vielzahl von Artikeln gar nicht vor. Dennoch erlaubt das Genre häufig die beste Schätzung über die Handlung eines Films und dient in vielen Fällen als Hauptkategorie zur Sortierung. Hier sind des Weiteren immer bis zu drei separate Genrebezeichner erlaubt.

Für die Erstellung des Genrelexikons wurden die bereits zur Klassifikation gesammelten und strukturierten Daten weiterverarbeitet. Ein Python-Programm extrahiert alle Genre-Bezeichner und befüllt das Lexikon mit allen Einträgen, die keinem Filmtyp entsprechen, um an dieser Stelle die vorher nicht erwünschte Trennung zwischen den beiden Kategorien zu realisieren. Das resultierende Lexikon enthält etwa 200 Genre-Einträge, wovon allerdings einige alternative Bezeichnungen realisieren.

**Mark-Up:** <GENRE>[Begriff]</GENRE>

### 3.2.4 Nationenlexikon

Die Nationalität gibt bei Filmen meist das Produktionsland/Herkunftsland an.

Die ursprünglichen Daten [2] enthielten nach dem Säubern nur die jeweiligen Staatsbezeichnungen und zugehörige Adjektivwurzeln in dieser Reihenfolge, mussten also als erster Schritt mittels eines kurzen Python-Programms getauscht und flektiert werden. Danach wurden mit Hilfe von regulären Ausdrücken per Suchen-und-Ersetzen die Tags an die zweite Spalte angefügt. Damit dient das Ersetzungswörterbuch nicht nur zur Annotation von Herkunftsbezeichnern, sondern auch zur Normalisierung derselben. Insgesamt erkennt das Lexikon damit über 1200 Flexionen von etwa 200 verschiedenen Länderadjektiven.

**Mark-Up:** <NAT>[Begriff]</NAT>
**Beispiel:** spanischer <NAT>Spanien</NAT>

### 3.2.5 Jahreszahlenlexikon

Bei der Jahreszahl ist die Intention die Repräsentation des Erscheinungsjahrs, allerdings wird bei manchen Filmen stattdessen das Produktionsjahr angegeben. In solchen Fällen wird ein suboptimaler Eintrag akzeptiert, da Produktionsjahr und Erscheinungsjahr meist entweder den Gleichen oder zumindest einen ähnlichen Zeitraum angeben.

Das Jahreszahlenlexikon wurde mit einem Python-Programm automatisch erstellt und getaggt. Es enthält zusätzlich zu allen regulären Jahreszahlen zwischen 1880 und 2029 auch alle vollen Jahrzehnte in „Xer“-Form. Die Kennzeichnung von Jahreszahlen wird benötigt, um das Produktions- beziehungsweise Erscheinungsjahr eines Films zu identifizieren.

**Mark-Up:** <YEAR>[Begriff]</YEAR>

## 3.3 Transducer

Im Folgenden bezieht sich die Bezeichnung Transducer auf die praktische Umsetzung der bereits im Grundlegenden erläuterten Theorien zum endlichen Transducer mittels der C(M)-Sprache, genauer gesagt auf die fertigen Programme. Selbiges gilt für weitere vorher bereits verwendete Begrifflichkeiten wie beispielsweise „Alphabet“. Außerdem werden lediglich die verwendeten Schritte genannt und kurz erläutert, nicht jedoch noch über die bereits gegebenen Erklärungen hinaus auf deren zugrundeliegende Mathematik und Implementierung eingegangen. Nähere Informationen zu den verwendeten Schritten sind dem Buch Finite-State Techniques von Mihov/Schulz zu entnehmen

Konzeptuell ist jede Grammatik unterteilt in 2 Schritte: Festlegung von lokalen Ersetzungsfunktionen, die einzelne Transducer-Bausteine miteinander verbinden, unter Verwendung im Voraus definierter Grundfunktionen und darauf folgend die Anwendung dieser auf den gesamten Text mittels einer globalen Ersetzungsfunktion (*global transducer*).

Hierzu liest das Programm einen Text ein und legt ihn gleichzeitig als zu verwendendes Alphabet fest, um eventuelle Sonderzeichen, beispielsweise asiatische Schriftzeichen, zu erhalten. Darauf folgt die Anwendung der ersten Transducer-Funktion mittels globaler Funktion auf den Ursprungstext. Die zweite Transducer-Funktion erhält im Anschluss die produzierte Ausgabe der vorangegangenen Funktion als Eingabe. Schlussendlich folgt die Rückgabe des gesäuberten Texts. Nur wenn eine Menge von Zeichen gefunden wird, die genau dem geforderten Muster entspricht, wird eine Transducer-Funktion erfolgreich angewandt.

Alle Transducer solcher Art sind in jedem Fall kontextfrei. Das vereinfacht zwar die Anwendung, da nicht eine große Menge von Fällen beachtet oder gar erst erlernt werden muss, und sorgt für eine deutlich schnellere Annotation von Texten, als dies beispielsweise mit Hilfe von Part-of-Speech-Tagging basierten Methoden der Fall wäre, lässt aber dadurch auch keinen Blick auf benachbarte Wörter und Ausdrücke zu. Somit ist es deutlich schwieriger, mit Problemen wie Ambiguität umzugehen.

Um eine Grammatik anwenden zu können, muss sie zunächst mit einigen Terminal-Befehlen kompiliert werden:

```
./cm genre_grammar.cm > genre_grammar.c
gcc -o genre_grammar genre_grammar.c
```

### 3.3.1 Vorannotation durch plmlm-Technik

Mittels der bereits unter 2.2.6 erläuterten plmlm-Technik wurden aus den beschriebenen Annotationslexika Transducer zur Vorannotation der Artikel generiert, die daraufhin auf den verbleibenden Korpus angewendet werden können. Hierbei muss jedoch auf die richtige Reihenfolge geachtet werden.

**Anwendungsschema** Den Anfang macht der Titel-Transducer, da die korrekte Erkennung aller Filmtitel Vorrang hat. Etwaige Fehl-Taggings innerhalb von Titel-Objekten werden später bereinigt. Als Zweites muss der Genre-Transducer folgen, um zuverlässige Markierung von Genre-Bezeichnern, die die Zeichenfolgen „Film“/„film“ beinhalten, zu gewährleisten. Auch hier können Fehl-Taggings innerhalb des Objekts im Anschluss bereinigt werden. Die Reihenfolge der übrigen Transducer kann prinzipiell beliebig gewählt werden, aber unter keinen Umständen darf einer der Transducer doppelt zur Anwendung kommen, da sonst eine Doppelung von Markierungen entsteht.

Die Erstellung und darauf folgende Ausführung eines plmlm-Transducers erfolgt gewöhnlich über das Terminal mit folgenden Befehlen:

```
./plmlm build genre.dic genre.tr
./plmlm apply genre.tr text_to_annotate.txt annotated_text.txt
```

**Automatisierung und erste Bereinigung** Zur weiteren Automatisierung des Annotationsprozesses erfolgte die Integration der Transducer in ein Python-Programm. Dabei sorgt das Programm jedoch nicht nur für die Anwendung in der korrekten Reihenfolge, sondern bereinigt zusätzlich vorher den eingegebenen Datensatz, indem Einträge mit der gesäuberten Titelliste abgeglichen und gegebenenfalls entfernt werden, sollten sie dort nicht vorhanden sein. Des Weiteren erfolgt im Anschluss an den Titel-Transducer direkt eine Löschung aller Titel-Objekte nach der ersten Titel-Endmarkierung. Dies hat eine Reduzierung von Informationsverlust durch Fehltaggings zum Ziel, da solche inkorrekten Markierungen andernfalls weiteres Tagging verhindern. Beispielsweise wäre vorstellbar, dass durch die hypothetische Existenz des Filmtitels „Buddy“ der Genrebezeichner „Buddy-Movie“ teilweise markiert und folglich für den Genre-Transducer unkenntlich gemacht wird.

Eine Anwendung aller fünf Transducer sorgt für einen komplett annotierten Text, allerdings kommt es bei Titel- und Genre-Objekten, wie bereits angedeutet, manchmal zu unerwünschten Markierungen innerhalb der Objekte. Hier kommen die speziell zu diesem Zweck vom Autor entwickelten Transducer ins Spiel.

### 3.3.2 Titel-Grammatik

Bei Titel-Attributen ergeben sich zwei besonders häufige Fehl-Taggings innerhalb des bereits getaggten Titel-Objekts:

1. Tagging von enthaltenen Filmtypen, wie etwa „Film“ als Filmtyp-Objekt
2. Tagging von vierstelligen Zahlen zwischen 1880 und 2029 als Erscheinungsjahr-Objekt

Beide Fälle sind normalerweise zwar nicht falsch, aber innerhalb des Titels nicht erwünscht, da dies schlussendlich zu unsauberen Ergebnis-Relationen führt. Folglich werden zwei Transducer-Funktionen benötigt, die Fehl-Taggings innerhalb des Titels finden und entfernen.

Erster Schritt beider Transducer ist das Finden von Titel-Anfangstags der Form <TITLE>und Kopieren eventuell folgender Zeichenketten. Der zweite Schritt ist jeweils das Finden und Löschen der Markierungen <FILMTYPE>, beziehungsweise <YEAR>, und ein erneutes Kopieren der folgenden Zeichenketten, also eines Filmtyps oder einer Jahreszahl. Darauf folgt eine weitere Finden-und-Löschen-Operation auf das jeweilige Endtag, Kopieren um eventuell folgende Leerzeichen oder weiteren Titeltext zu erkennen und schlussendlich die Suche nach einem Titel-Endtag.

**Beispiel zum ersten Transducer:**

```
Input = "<TITLE>Tron (<FILMTYPE>Film</FILMTYPE>)</TITLE>"
Finde "<TITEL>"
Kopiere Zeichen
Lösche "<FILMTYPE>"
Kopiere Zeichen
Lösche "</FILMTYPE>"
Kopiere Zeichen
Finde "</TITEL>"
Output = "<TITLE>Tron (Film)</TITLE>"
```

### 3.3.3 Genre-Grammatik

Im Gegensatz zu den gänzlich unerwünschten Annotationen innerhalb des Titels besitzen die zusätzlichen Tags innerhalb eines Genre-Objekts durchaus Informationsgehalt und

sollen deshalb zwar behalten, aber vom Genre abgespalten werden. Hierbei handelt es sich um Filmtyp-Taggings, die durch Vorkommen von „Film“/„film“ innerhalb eines Genre-Bezeichners entstehen, wie bereits in Absatz 3.3.1 angesprochen. Da es zwei Arten von Vorkommen gibt, entweder am Anfang („Filmdrama“) oder am Ende („Dokumentarfilm“) des Begriffs, besteht erneut Bedarf für zwei Transducer-Funktionen.

Beide arbeiten hierbei zunächst nach dem selben Prinzip wie die Titel-Transducer, erkennen also ein Genre-Starttag und folgende Zeichen. Wird nun eine Filmtyp-Startmarkierung erkannt, erfolgt eine Ersetzungsoperation durch ein Genre-Endtag kombiniert mit einem Filmtyp-Starttag. Zweck der Ersetzung ist die Abspaltung des Filmtyp-Objekts. Zur Vervollständigung der Trennung folgt das Kopieren weiterer Zeichen und eine letzte Ersetzung von gedoppelten Filmtyp- und Genre-Endtags durch ein einzelnes Filmtyp-Endtag. Dies führt zur Isolation von am Begriffsende stehenden Filmtyp-Objekten.

Am Wortbeginn auftauchende Filmtypen werden mit dem zweiten Transducer getrennt, indem die gedoppelten Starttags durch ein entsprechendes Filmtyp-Tag ersetzt werden. Auf eine Kopieroperation folgt die Ersetzung eines allein stehenden Filmtyp-Endtags durch ein Filmtyp-Endtag verbunden mit einem Genre-Starttag.

**Beispiel zum ersten Transducer:**

```
Input = "<GENRE>Abenteuer<FILMTYPE>Film</FILMTYPE></GENRE>"
Finde "<GENRE>"
Kopiere Zeichen
Ersetze "<FILMTYPE>" durch "</GENRE><FILMTYPE>"
Kopiere Zeichen
Ersetze "</FILMTYPE></GENRE>" durch "</FILMTYPE>"
Output = "<GENRE>Abenteuer</GENRE><FILMTYPE>Film</FILMTYPE>"
```

### 3.3.4 Implementierung

Ein generelles Problem der Anwendung von Transducer-Grammatiken auf größere Texte ist der hohe Rechenaufwand und, daraus resultierend, eine exponentiell ansteigende Programmlaufzeit. Deshalb sollte zumindest jede Grammatik nur auf einzelne Artikel angewandt werden, um die eingelesenen Texte kurz zu halten.

Die Implementierung der beiden Grammatiken erfolgt aus diesen Gründen durch ein Python-Programm, das als ersten Schritt die JSON-Datei mit den automatisch klassifizierten, vorannotierten Artikeln lädt.

Für jeden Artikel entsetzt das Programm überflüssige Anführungszeichen durch „**“ und fügt neue an Textanfang und -ende an. Zusätzlich wird an dieser Stelle der Key, vorher der mitannotierte Artikeltitel, durch eine ID-Nummer ersetzt. Dies geschieht aus zwei Gründen: Erstens ist das Ziel der Arbeit, die gesuchten Informationen aus dem Einleitungstext zu extrahieren, nicht dem Titel. Zweitens reduziert eine Verkürzung der Daten auch weitere Verarbeitungszeiten. Die vorherige Formatierung des Einleitungstexts ist erforderlich, um den Anforderungen der Shell-Syntax zu entsprechen, da nun der Text an die mittels der subprocess-Bibliothek eingebunden kompilierten Programme übergeben wird.

Als letzter Schritt werden die verarbeiteten Einträge in ein Dictionary mit der ID als Key abgelegt und am Ende des Programmdurchlaufs wieder in eine JSON-Datei geschrieben.

Ein Test anhand der ersten 998 Artikel der vollen Film-Daten, von denen etwa 900 verarbeitet worden sind, ergab eine Laufzeit von etwa 44 Minuten, also etwas weniger als fünf Minuten pro 100 verarbeiteter Artikel. Für den gesamten Datensatz mit um die 20.000 Einträgen lässt sich damit eine Laufzeit von etwa 16 Stunden abschätzen.

## 3.4 Relationen

Als letzter Schritt des Prozesses folgt die Extraktion der annotierten Daten und die Bildung der finalen Relationsmengen. Zusätzlich soll kurz eine Information über die Einordnung der Ergebnisrelationen gegeben werden.

### 3.4.1 Relationsextraktion

Zur Extraktion der annotierten Daten und Bildung der Relationen kommt ein weiteres Pythonprogramm zum Einsatz. Zu jeder Relation werden zunächst einfache reguläre Ausdrücke der Form „Variable_Starttag+'(.+?)'+Variable_Endtag“ (als String) gebildet, die das komplette Objekt matchen, dabei aber die Zeichenfolge zwischen den Markierungen in einer extra Gruppe abspeichern. Dies ermöglicht die einfache Extraktion über den Aufruf des korrekten *group*-Objekts innerhalb des erhaltenen *re.match*-Objekts.

Für jede Kategorie wird der erste Match verwendet, da im Allgemeinen davon auszugehen ist, dass das erste Vorkommen einer Information entweder korrekt oder das einzige seiner Art ist. Im Falle des Titels sind ohnehin alle weiteren Markierungen entfernt worden (siehe 3.3.1). Die Genre-Relation stellt dabei eine Ausnahme dar, hier sind bis zu drei Matches gestattet. Sollte eine Relation gar nicht auftreten, wird stattdessen „N/A“ für „not available“ eingetragen.

Die gefundenen Werte werden konkatiniert und in ein Dictionary eingetragen, welches wiederum bei Programmende zeilenweise in eine Textdatei geschrieben wird.

### 3.4.2 Relationsdatei

Die damit entstandene Datei enthält nun Relationen zu allen Wikipedia-Artikeln, die zuerst vom Klassifizierungsprogramm als Film eingeordnet worden und danach bei der manuellen Nachkorrektur in der Liste verblieben sind. Dabei entspricht jede Relation der Form:

```
<Titel|Filmtyp|Genre1(/Genre2/Genre3)|Herkunft|Erscheinungsjahr>

<Prinzessin Mononoke|Film|Zeichentrick/Anime|Japan|1997>
<Tron|Spielfilm|N/A|USA|1982>
```

### 3.4.3 Relationsbewertung

An dieser Stelle soll zusätzlich noch festgelegt werden, wie die extrahierten Relationen bei der Auswertung eingestuft werden:

**Inakzeptable Relationen:** Alle Relationen mit zwei oder mehr falsch eingetragenen Begriffen sind als unzureichendes Ergebnis zu werten. Hier sind die zugrunde liegenden Informationen zu stark verzerrt, um noch als wertvoll angesehen zu werden.

**Akzeptable Relationen:** Als akzeptable Relationen werden hier alle Relationen bezeichnet, die nur eine falsche Eintragung enthalten. Zwar ist der Informationsgehalt dieses Objekts geschwächt, aber nicht in zu starkem Maße.

**Perfekte Relationen:** Perfekte Relationen entsprechen in ihrem Inhalt dem Goldstandard und sind damit optimal getaggt und extrahiert. Syntaktische Wohlgeformtheit spielt dabei keine Rolle, lediglich die inhaltliche Aussage. Diese Relationen sind das Ziel, da sie die gesuchten Kerninformationen des ursprünglichen Textes bestmöglich wiedergeben.

## 3.5 Evaluation

Im Weiteren wird untersucht, wie präzise die Filmerkennung ist, welche Relationen korrekt zugeordnet sind und wie die Ergebnisse bei den für diese Arbeit gewählten Performance Measures abschneiden.

### 3.5.1 Datensätze und Goldstandard

Um einen möglichen Bias und ein daraus entstehendes Overfitting zu vermeiden, wurden für die bisherige praktische Arbeit, so weit wie möglich, nur die ersten 998 automatisch als Filmartikel klassifizierten Einträge betrachtet, sie stellen somit die Entwicklungsdaten dar (voll annotierte Datei: „movie_0_998_clean.json“).

Anmerkung: Die Korrektur eines Fehlers in den Lexika gegen Ende der Arbeit hatte eine Verbesserung des Klassifikators zur Folge, weshalb mehr Artikel korrekt klassifiziert werden konnten und es zu Verschiebungen innerhalb der Filmliste und des Datensatzes kam. Aus diesem Grund sind einige Grenzen etwas verschoben und stellen keine „glatten Zahlen“, wie etwa 100, mehr dar.

Zur Beurteilung der Arbeit benötigte der Autor des Weiteren einen möglichst unabhängigen Goldstandard für zumindest eine kleine Menge an Filmrelationen. Die Realisierung eines solchen Goldstandards übernahm zur Sicherstellung einer unvoreingenommenen Annotation eine Kommilitonin des Autors anhand von übergebenen Annotation-Guidelines.

Ziel war die manuelle Extraktion der ersten 102 von der Erstellerin gefundenen Film-Artikel in der Datei „movie_999_1999_clean.json“. Bei dieser Datei handelt es sich um die Testdaten, die die vollständig annotierten Einträge 999 - 1999 aus dem kompletten Filmartikel-Datensatz umfassen. Insgesamt wurden damit 115 Einträge von der Erstellerin betrachtet. Alle betrachteten Artikel, die keinem Film entsprachen, haben die Markierung # am Anfang ihrer Zeile erhalten. Artikel, die vom Klassifikator nicht gefunden wurden, sind für die Evaluation der Informationsextraktion nicht relevant, da sie auf Unzulänglichkeiten des Klassifikators zurück zu führen sind.

**Annotation-Guidelines:** Die Betrachtung beschränkt sich je Artikel auf die gegebenen 300 Zeichen. Sämtliche abgeschnittene Informationen gelten als verloren und werden nicht mit einbezogen. Damit gilt das selbe Prinzip wie bereits bei dem automatisierten Verfahren.

**Nationalität:** Sofern im Artikel keine Nennung eines Ursprungslands erfolgt, wird keine Nationalität eingetragen, selbst wenn dem Ersteller des Goldstandards die Nationalitäten des Regisseurs, der Mitwirkenden und/oder des Produktionsstudios bekannt sind, oder die Nationalität durch den Originaltitel erahnbar ist. Sofern mehrere Ursprungsländer genannt werden, wird nur das Erste eingetragen.

**Filmtyp:** Beim Auftreten von mehreren unterschiedlichen Filmtypen wird lediglich der Erste beachtet. Bei Verbindungen von zwei Nomen (zum Beispiel „Spielfilm-Zweiteiler“), wird das Erste als semantischer Fokus betrachtet und als korrekte Relation aufgefasst. Bei Verbindungen von Adjektiv und Nomen (z.B. „zweiteiliger Fernsehfilm“) wird das Nomen als semantischer Fokus betrachtet und somit zur Eintragung verwendet.

**Genre(s):** Bis zu drei Genre-Einträge sind gestattet, die Reihenfolge der Eintragung entspricht der Reihenfolge des Vorkommens im Artikel.

Die bestehenden Annotationswörterbücher gelten als Anhaltspunkte für die Eintragung in die Relations-Tabelle. Sollten Ursprungsländer, Genres oder Filmtypen hier je-

doch gefehlt haben, werden diese dennoch als Relation zum Film eingetragen, um die Vollständigkeit der verwendeten Lexika zu testen.

### 3.5.2 Auswertung

Zur Beurteilung der Ergebnisse kommt ein weiteres Python-Programm zur Verwendung, das die automatisch erstellten Relationen mit dem Goldstandard abgleicht und die Einstufung jeder einzelnen Relation mittels einer Matrix darstellt, die aus fünf Spalten (eine pro Relationskategorie) und 102 Zeilen (eine pro Film-Eintrag im Goldstandard) besteht. Die gewonnenen Daten können genutzt werden, um Informationen über die Leistung der verwendeten Verfahren zu berechnen.

**Klassifikator** Wie bereits angedeutet, gehören von 115 betrachteten Relationen genau 13 nicht zu Film-Artikeln und sind damit als Fehlklassifikationen zu werten. Als solche sind sie auf den Klassifikator und die manuelle Nachkorrektur der Titelliste zurückzuführen. Leider ist dem Autor nicht bekannt, wie viele Artikel vom Klassifikator fälschlich nicht als Film eingeordnet worden sind (*False Negatives* des Klassifikators), somit kann hierfür keine Berechnung von Recall oder F1-Score durchgeführt werden. Die hierfür notwendige manuelle Sichtung der Rohdaten überstiege den für diese Arbeit vertretbaren Zeitaufwand. Die erreichte Precision (p) lässt sich jedoch berechnen:

$$TP = TruePositives = 102$$

$$FP = FalsePositves = 13$$

$$p = \frac{TP}{TP + FP} = \frac{102}{102 + 13} = 0,887$$

Zusätzlich ergibt ein Durchlauf ohne das manuell nachkorrigierte Titellexikon insgesamt 6 weitere unerwünschte Relationen, bis 102 Film-Relationen gefunden sind. Somit ergibt sich für den Klassifikator ohne Nachkorrektur:

$$FP = 19$$

$$p = \frac{102}{102 + 19} = 0,843$$

Dies entspricht einer Klassifikator-Precision von 84,3% auf dem sehr kleinen Testdatensatz. Im Beispiel verbessert sich die Precision durch die Verwendung des Titellexikons also um etwa 4 Prozentpunkte. Damit bewegt sich das Ergebnis hier in einem ähnlichen Bereich wie die vorherige Schätzung (siehe 3.1.2) Im Folgenden werden jedoch lediglich die 102 korrekt klassifizierten Artikel betrachtet, um die Effektivität der genutzen Methoden unabhängig davon betrachten zu können.

**Annotation** Um die Effektivität der Annotation abzuschätzen, bietet sich sowohl die Betrachtung ganzer Einträge, als auch der einzelnen Bestandteile an.

| Perfekte Relationen | Akzeptable Relationen | Inakzeptable Relationen |
|---|---|---|
| 60 | 34 | 8 |

Tabelle 3.1: Aufteilung auf die festgelegten Gruppen

Die Tabellen 3.1 und 3.2 zeigen, dass zwar lediglich etwa 58,8% der extrahierten Relationen perfekt sind, die meisten Fehler dabei aber in der Kategorie Genre auftauchen. Titel und Genre weisen jedoch besonders hohe Werte in der Gruppe *False Element* (FE), auf, also Felder, wo der Text die korrekte Information enthält, aber die Falsche eingetragen wird.

| Kat./Gruppe | TP | FP | FN | TN | FE |
|---|---|---|---|---|---|
| Titel | 93 | 0 | 1 | 0 | 8 |
| Filmtyp | 94 | 0 | 2 | 1 | 5 |
| Genre | 42 | 7 | 5 | 40 | 8 |
| Nation | 57 | 6 | 5 | 32 | 2 |
| Jahr | 96 | 2 | 0 | 4 | 0 |
| Total | 382 | 15 | 13 | 77 | 23 |

Tabelle 3.2: Vorkommen der einzelnen Kategorien

Zusätzlich kommt es bei Genre und Nation besonders häufig zu *False Positives* (FP), da hier Informationen, die nicht enthalten sind, fälschlich markiert werden. Ein ähnliches Bild zeigt sich bei den *False Negatives* (FN), es wird also nichts eingetragen, wenn die gesuchte Information vorhanden ist. In den Kategorien True Postive (TP), es ist eine Information vorhanden und sie wurde korrekt eingetragen, und *True Negative* (TN), es ist keine Information vorhanden und das Feld wird als leer markiert, werden zusammengerechnet dennoch allgemein hohe Werte errechnet, was für die Effektivität der Annotation und Extraktion spricht.

| | | Titel | Filmtyp | Genre | Nation | Jahr | Total |
|---|---|---|---|---|---|---|---|
| Ohne FE | Precision | 1.0 | 1.0 | 0.85714 | 0.90476 | 0.97959 | 0.96222 |
| | Recall | 0.98936 | 0.97917 | 0.89362 | 0.91935 | 1.0 | 0.96709 |
| | $F_1$-Score | 0.99465 | 0.98947 | **0.875** | 0.912 | 0.98969 | **0.96465** |
| Mit FE | Precision | 0.92079 | 0.94949 | 0.73684 | 0.87692 | 0.97959 | 0.90952 |
| | Recall | 0.91176 | 0.93069 | 0.76364 | 0.89063 | 1.0 | 0.91388 |
| | $F_1$-Score | 0.91625 | 0.94 | **0.75** | 0.88372 | 0.98969 | **0.91169** |

Tabelle 3.3: Ergebnisse mit/ohne *False Element*, gerundet auf fünfte Nachkommastelle

Die Performance Measures ohne *False Elements* in 3.3 sind, wie in 2.1.4 bereits erklärt, als Indikator für fälschlich leere/nicht-leere Felder zu verstehen. Dabei gibt der Recall an, wie oft ein Feld fälschlich leer bleibt, während die Precision eine Aussage darüber macht, mit welcher Häufigkeit ein Feld fälschlich einen Eintrag erhält. Wie bereits in 3.2 zu sehen, treten vor allem beim Genre Probleme sowohl mit Über- als auch in etwas geringerem Maße Unterbefüllung auf, was sich in einem niedrigeren Recall, oder Precision in Bezug auf die Unterbefüllung, und F-Score niederschlägt. Ein ähnliches Muster ist bei der Kategorie Nation zu beobachten, wenn auch abgeschwächter. Für die hohen Werte in der Kategorie Titel zeigt sich das sehr große Titellexikon verantwortlich, dank dem nur sehr selten kein möglicher Titel gefunden werden kann. Filmtyp und Jahr schneiden perfekt, beziehungsweise nahezu perfekt, ab, da bereits der Klassifikator die Existenz von Filmtypen und Jahreszahlen als positives Indiz für Filmartikel auffasst, in den Artikeln der Datensätze also fast immer mindestens ein Vorkommen beider Kategorien existiert. Hier liegt ein durch den Klassifikator verursachter Bias vor, jedoch ist im Fall der Jahreszahlen das gute Abschneiden auch auf das theoretisch komplette Lexikon zurückzuführen, da es nur eine begrenze Anzahl von möglichen Jahreszahlen gibt. Die hohen Scores beim Filmtyp liegen allerdings wohl eher im kleinen Vergleichsdatensatz begründet, nicht in einem nahezu fehlerlosen Programm oder Lexikon.

Im Vergleich dazu geben die Performance Measures mit *False Elements* zusätzlich Informationen über die Korrektheit aller nicht-leeren Felder, es wird hiermit also die Summe aller falschen Eintragungen angegeben. Hier schneidet die Kategorie Genre erneut in allen Bereichen deutlich schlechter ab als alle Anderen, da etwa ein Viertel aller Eintragungen inkorrekt sind, was als unzureichend einzuordnen ist. In der Kategorie Titel kommt es

ebenso zu einer merklichen Verschlechterung der Ergebnisse, dennoch bleiben die Werte hier auf einem als gut zu bewertenden Niveau. Zwar ist bei der Kategorie Nation nur eine leichte Verschlechterung zu beobachten, in Kombination mit den bereits vorher nicht optimalen Ergebnissen ergibt sich dadurch allerdings eine offensichtliche Schwäche. Jahr schneidet genauso ab, wie ohne Einbezug der *False Elements*, da solche hier nicht vorliegen. Zuletzt fällt zusätzlich eine deutliche Verschlechterung beim Filmtyp auf, der dennoch nach wie vor gute Ergebnisse zeigt.

Anhand der obigen Gesamtwerte (Total in Tabelle 3.2) lässt sich erkennen, dass die korrekte Erkennung, ob Information vorhanden ist, sehr gut funktioniert ($F_1 = 0.96465$). Dies lässt den Schluss zu, dass die verwendeten Annotationslexika über ausreichend Daten verfügen, um die allermeisten relevanten Relationsvorkommen abzudecken. Des Weiteren kann anhand der unteren Gesamtwerte ($F_1 = 0.91169$) geschlussfolgert werden, dass dahingegen bei der korrekten Extraktion und Zuordnung noch kleinere Probleme bestehen. Dennoch kommen auch hier, zumindest im Falle des Tests, zufriedenstellende Daten zustande.

Zuletzt bietet sich noch eine Detailbetrachtung der *False Elements* in der Kategorie Genre an, da dieses Relationselement potenziell aus drei Begriffen bestehen kann und damit rein statistisch das Fehlerrisiko zunimmt.

| Position | TP | FP | FN | Gold | Auto |
|---|---|---|---|---|---|
| 1 | 6 | 1 | 1 | 5 | 2 |
| 2 | 3 | 4 | 1 | 3 | 4 |
| 3 | 0 | 2 | 0 | 0 | 2 |

Tabelle 3.4: Vorkommen von mehrteiligen Genre-Elementen bei FEs

Wie in der Tabelle 3.4 zu sehen ist, hängen alle Fehlzuordnungen mit mehrteiligen Genre-Elementen zusammen. Dabei fällt vor Allem auf, dass die erste Position in 75% der Fälle korrekt zugeordnet worden ist, während sich Fehler an 2. und 3. Stelle häufen. Des Weiteren werden zu häufig mehrere Genres zu Filmen gefunden, bei denen nur ein Genre genannt wird. Leider ist die Menge an *False Elements* mit acht Elementen aufgrund der geringen Größe des Vergleichsdatensatzes (Goldstandard) so klein, dass sich keine sicheren Verallgemeinerungen machen lassen, dennoch wird mit Hilfe dieser Daten im Folgenden noch eine Vermutung angestellt, warum gerade das Genre so schlecht abschneidet

## 3.6 Fehleranalyse

An einigen Stellen eines langwierigen Arbeitsprozesses gilt es, gewisse Entscheidungen über die weitere Vorgehensweise zu treffen. Leider stellt sich zuweilen bei späteren Arbeitsschritten, oder zuletzt bei der Auswertung der Ergebnisse, heraus, dass vorher getroffene Entscheidungen auch Fehler und Probleme zur Folge hatten oder bestimmte Aspekte und Alternativen nicht in Betracht gezogen worden sind. Es folgt eine Aufzählung einiger solcher Fehler und ein kurzer Ausblick auf mögliche Verbesserungen. Dabei ist anzumerken, dass einige Fehler und Einschränkungen bewusst in Kauf genommen werden mussten, um innerhalb des Umfangs einer Bachelorarbeit zu bleiben. Größere Verbesserungen, die aufgrund dieses Umstands nicht möglich waren, werden in 3.8 angesprochen.

### 3.6.1 Annotation

Wie bereits in der Auswertung erläutert, scheinen die verwendeten Annotationslexika ausreichend vollständig zu sein, um den Großteil der Informationen zu annotieren. Ebenso erfolgt die Anwendung mittels der plmlm-Technik mit zufriedenstellenden Ergebnissen. Dennoch sollen in diesem Abschnitt einige durch die Lexika und Annotationsmethode ausgelöste Fehler besprochen werden. Eine etwas genauere Problembehandlung der „wahllosen“ Annotation aufgrund der fehlenden Kontextsensitivität folgt in 3.7.

**Titel** Da das verwendete Titellexikon automatisiert während der Klassifikation erstellt wird, ist garantiert, dass jeder benötigte Filmtitel enthalten ist. Das ist zwar sehr effizient, bringt aber auch Probleme mit sich. So werden Artikeltitel, bei denen fälschlich das Label Film vergeben wird, ohne weitere Kontrolle zum Lexikon hinzugefügt. Durch eine manuelle Säuberung können zwar einige Einträge entfernt werden, trotzdem entstehen bei der Extraktion noch Relationen, die nicht zu Filmen gehören, da der Titel im Lexikon steht.

Ein weiterer Faktor ist damit die begrenzte Übertragbarkeit auf andere Texte außerhalb der Wikipedia. Zum Einen muss immer als erster Schritt eine Klassifikation durch das hier genutzte Programm erfolgen, um die notwendige Liste zu erhalten. Zum Anderen kann die Liste nur erstellt werden, wenn der Titel des Artikels bekannt ist. Somit eignet sich die hier verwendete Methode der Lexikonerstellung nicht für andere Anwendungsbereiche. Bleibt man jedoch bei ähnlichen Rahmenbedingungen, erhält man ein Wörterbuch mit hoher Vollständigkeit.

Diese hohe Vollständigkeit bringt jedoch im Falle des Titellexikons auch einen enormen Nachteil: Je mehr Filme man erkennen möchte, desto mehr Filmtitel benötigt man. Folglich werden umso mehr mögliche Zeichenkombinationen als Filmtitel abgespeichert. Erfolgt nun die Durchsuchung eines Textes nach Vorkommen solcher Kombinationen, wächst mit einem großen Lexikon auch die Wahrscheinlichkeit der fälschlichen Annotation gewöhnlicher Ausdrücke - es entsteht Ambiguität. So existiert beispielsweise ein Film mit dem Titel „Woo“, was zum fälschlichen Teil-Tagging des Namens „Woody Allen“ führt. Zur Eingrenzung solcher Fehler werden im hier verwendeten Lexikon Titel mit weniger als zwei Zeichen übergangen und der damit verbundene Informationsverlust in Kauf genommen. Zusätzlich werden während der Annotation alle Titel-Annotationen nach dem ersten Titel-Objekt wieder entfernt, um eine möglichst saubere Weiterverarbeitung zu gewährleisten (siehe Abschnitt 3.3.1). Dennoch lassen sich solche auf Ambiguität beruhenden Fehler mit den hier verwendeten Methoden nicht ganz vermeiden.

**Genre** Beim Genre-Lexikon zeigten sich im Laufe der Arbeit ebenso Probleme mit Ambiguität und bereichsübergreifenden Bedeutungen. Zwar konnten diese mitunter durch die Reduzierung der alternativen Bezeichner reduziert werden, ganz eliminieren ließen sie sich jedoch nicht. So wird beispielsweise ein Teil des Wortes „Dramatiker“ entsprechend als Genre getaggt, da es „Drama“ enthält, obwohl es sich hierbei lediglich um eine Person handelt, die etwa eine Rolle in der Handlung spielen könnte.

Zusätzlich enthält das Genre-Lexikon keine kleingeschriebenen Einträge. Sollte ein unbekanntes Kompositum aus zwei Genre-Bezeichnern gebildet werden, etwa „Actiontrash“, erfolgt keine Markierung. Die Behandlung solcher Fälle ließe sich zwar, etwa durch die Aufnahme jedes Genre-Begriffs in kleingeschriebener Form, relativ simpel realisieren, könnte potenziell aber wiederum zu Ambiguität und daraus resultierenden Fehltaggings führen.

Dennoch gibt es in den Texten immer wieder Genres, die nicht erkannt werden, weil kein Eintrag zu ihnen vorhanden ist oder weil die genutzte Bezeichnung nicht bekannt ist. Eine Erweiterung des Genre-Lexikons wäre also eine sinnvolle Verbesserung.

**Nationen und Jahreszahlen** Auch bei der Annotation der Nationalitäten-Adjektive kommt es an mehreren Stellen zu Fehlern. Besonders häufig sind dabei das fälschliche Tagging von solchen Adjektiven in Bezug auf die Originalsprache eines Films und die Erkennung von Nationalitäten, die sich auf die Handlung beziehen. Weiterhin wäre auch die inkorrekte Erkennung der Herkunft eines teilhabenden Schauspielers oder Regisseurs als Produktionsland denkbar.

Ein anderer Aspekt sind fehlende Einträge zu Länder- und Städtenamen im Lexikon. Dies verhindert zwar, dass seltene Vorkommen von Herkunftsbezeichnern wie „... ist ein Film aus Berlin (Deutschland)“ erkannt werden, verhindert aber auch das fälschliche Erkennen von Handlungsorten, die häufig so angegeben werden. Des Weiteren würde die korrekte Zuordnung von Städtenamen zu Ländern ein sehr großes Lexikon erfordern.

Ähnliche Fehler entstehen bei der Jahreszahlen-Annotation. Da der Annotationsmechanismus keinen Unterschied dabei macht, ob die Jahreszahl aus der Handlung des Films stammt oder nicht, geraten solche Zeitangaben hin und wieder in die Relationen.

In beiden Fällen sind die Fehler hier jedoch auf das grundlegende Verfahren zurückzuführen, nicht die Vollständigkeit der Annotationslexika.

### 3.6.2 Transducer-Grammatiken

Die Bereinigung und Verfeinerung der Annotation erfolgt mit den verwendeten Grammatiken zwar bereits zufriedenstellend, doch gerade bei der Titel-Grammatik kommt es hin und wieder noch zu Fehlern. So kann es beispielsweise vorkommen, dass in einem Titel ein Genre-Bezeichner vorhanden ist. Aufgrund fehlender Transducer-Regeln für diesen Fall erfolgt keine Bereinigung der entsprechenden Markierungen. Weiterhin gibt es Vorkommen von Nationalitäten-Adjektiven in Titeln, die somit markiert werden. Da bei der Nationen-Annotation zusätzlich zur Markierung noch eine Normalisierung erfolgt, wäre die Bereinigung solcher Fehler durch den Verlust der Flexion sehr schwierig.

Die Transducer-Grammatiken leiden im Allgemeinen unter einem klaren Problem: Für jeden Fall müssen Regeln geschrieben werden, und jede weitere Regel erhöht die Laufzeit. Diese Limitierung hatte die bewusste Entscheidung zur Folge, im Falle der Titel nur die beiden Fehlerquellen zu untersuchen, die nach dem Ermessen des Autors am Häufigsten zu Komplikationen führen.

### 3.6.3 Extraktionsschema

Bei der Extraktion, wie auch dem vorangegangenen Annotationsvorgang, entstehen die meisten Fehler vermutlich durch das kontext-insensitive Vorgehen. Hier bestünde ein weiteres Mal die Möglichkeit, Markierungen in bestimmten Kontexten zu übergehen, wie etwa im Fall der Jahreszahlen aus der Handlung. Da solche Maßnahmen aufgrund der komplexen Implementierung und damit einhergehendem Rechen- sowie Zeitaufwand nicht erfolgen, werden vorherige ungünstige Annotationen weiter in die endgültigen Relationen mitgenommen.

In diesem Kontext gilt es auch, noch einmal die Kategorie Genre anzusprechen, zumal die Drei-Teilung des Genre-Elements eine weitere Fehlerquelle bei der Extraktion, sowie Relationsbildung, ausmacht. Während das erste erkannte Genre häufig noch korrekt ist,

nimmt die Fehlerquote bei den weiteren gefundenen Elementen stark zu. Grund dafür ist vorrangig, dass die Mehrzahl der Film-Artikel der Wikipedia nur über einen korrekten Genre-Bezeichner verfügt, gleichzeitig aber fast alle Informationen über die Handlung oder etwaiges Ursprungsmaterial geben. Somit kommt es beispielsweise vor, dass ein Film fälschlich als zweites Genre „Musical“ erhält, da es sich um die Verfilmung eines Musicals handelt.

### 3.6.4 Goldstandard

Die Evaluation der Ergebnisse erlaubt zwar eine grobe Einschätzung der allgemeinen Leistung der eingesetzten Verfahren, ist aber aufgrund des sehr kleinen Goldstandard-Datensatzes nicht als vollständiger Nachweis über die Richtigkeit der produzierten Ergebnisse zu werten. Leider stellt die Zusammenstellung eines großen Goldstandards einen enormen Zeitaufwand dar, oder alternativ einen hohen Kostenaufwand, war also im Rahmen dieser Arbeit nicht umsetzbar.

## 3.7 Entstandene Probleme

### 3.7.1 Überfixierung auf Einleitungsextraktion

Gerade zu Anfang der Arbeit wurden verschiedene Ansätze ausprobiert, die sich jedoch als Sackgassen herausstellten. Besonders viel Zeit wurde hier auf die Entwicklung eines Programms zur Kürzung der Artikel auf die Einleitungen verwendet. Das entwickelte Programm nutzt Part-of-Speech-Tagging zum Finden unvollständiger Sätze, genauer gesagt Sätze, die über keine Verbalphrase verfügen. Solche Konstrukte zeichnen sehr häufig eine Unterüberschrift aus, weisen also auf das Ende der Einleitung hin. Zwar sind die Ergebnisse dabei zufriedenstellend, eine hohe Laufzeit (mehrere Stunden für einen geringen Teil des Korpus) und teils übermäßig lange Einleitungstexte disqualifizierten dieses Verfahren jedoch.

### 3.7.2 Ambiguität des Genrelexikons

Gerade die Thematik der Ambiguität von Begriffen führte anfänglich zu größeren Problemen. So enthielten die Genredaten ursprünglich deutlich mehr alternative Bezeichnungen, wie etwa „Tanz“ für das Genre des Tanzfilms. Die Absicht lag in der Erkennung von verketteten Genreangaben, zum Beispiel „Tanz-Dokumentarfilm“. Durch die häufige Verwendung einiger Wörter außerhalb von Genrebezeichnungen musste die Anzahl an Alternativbegriffen jedoch wieder reduziert werden. Zusätzlich werden viele Genres ebenso in Bezug auf andere Medien verwendet, besonders häufig bei Büchern.

## 3.8 Zukünftige Arbeit

An dieser Stelle werden noch kurz einige Einzelheiten angesprochen, die Stoff für weitere Arbeiten an dem untersuchten Themenbereich bieten.

### 3.8.1 Annotationslexika

Wie bereits in der Auswertung erläutert, scheinen die verwendeten Annotationslexika ausreichend vollständig zu sein, um den Großteil der Informationen zu annotieren. Jedoch soll an dieser Stelle auf das Zipf'sche Gesetz verwiesen werden, das in Bezug auf die Vorkommen von verschiedenen Wörtern in Texten besagt (vereinfacht): Das am Meisten vorkommende Wort kommt in etwa doppelt so häufig vor wie das am zweitmeisten Vorkommende, welches wiederum doppelt so häufig vorkommt wie das darauf Folgende. Es kommt also jedes Wort halb so oft vor wie das in der Rangliste Vorhergegangene, sofern vorhanden [22]. Daraus lässt sich wiederum im vorliegenden Fall folgern, dass eine geringe Anzahl von Wörtern einen großen Teil aller Vorkommen von relevanten Begriffen darstellt. Werden lediglich eben diese Begriffe in das Lexikon eingefügt, führt das zwar zu zu einer guten Abdeckung aller Vorkommen, nicht aber aller Begriffe. Folglich gehen also einzigartige Information verloren.

Aus diesen Gründen sind sehr große Lexika, solange sie nicht für zu starke Ambiguität sorgen, für die Informationsextraktion vorteilhaft. Gleichzeitig nimmt ein solches Unterfangen enorm viel Zeit in Anspruch, da die Findung schwieriger wird, je seltener ein Wort ist, während die häufigen Wörter vergleichsweise schnell zusammen getragen werden können. Um nicht den Rahmen dieser Bachelorarbeit zu sprengen, wurde auf eine so intensive Suche verzichtet, zumal die erstellten Lexika ausreichten. Im Folgenden sollen jedoch kurz Wege besprochen werden, wie solche Vergrößerungen in zukünftigen Arbeiten realisiert werden könnten.

**Erweiterung der Lexika zu Genre und Titel** Zur Generierung einer größeren Filmliste käme die Nutzung bereits existenter Ressourcen im Internet in Frage. Besonders interessant wäre hier die Webseite IMDb, die eine große Online-Datenbank zum Thema Film und Fernsehen unterhält. Automatisches Webcrawling oder gar ein offizieller Zugriff auf die Daten könnten als Grundlage für ein Lexikon mit einem Großteil der relevanten Titeleinträge dienen. Hierbei entsteht jedoch auch Problempotenzial - Möchte man alle auf IMDb existenten Filme abdecken, würde man 516.726 Titelbezeichner benötigen. Die Inklusion von Kurzfilmen erhöht diese Zahl auf fast 1.2 Millionen Titel [3]. Bei einer solchen Liste käme es zu starker Ambiguität, was eine deutliche Häufung von falschen Taggings zur Folge haben könnte.

Eine alternative Variante zur Schaffung des Titellexikon stellt die Erkennung von Titeln mittels verschiedener Satzanalyse-Methoden dar. So könnten bestehende Tokenizer genutzt werden, um einen Text in einzelne Sätze und Tokens zu zerlegen. Darauf erfolgt die Nutzung eines Part-of-Speech-Taggers zur Zerlegung jedes Satzes in seine Satzbausteine. Da viele Einleitungssätze zu Wikipedia-Artikeln dem Schema „X ist ... Y“ folgen, wo X der gesuchte Titel ist und Y dem Bereich entspricht, also beispielsweise „Tron ist ... Spiel-[Film]“, ist anzunehmen, dass das Subjekt des Satzes dem Titel entspricht, wenn das Objekt einem Filmtyp oder Ähnlichem entspricht.

Beide Methoden ließen sich auch zur Vergrößerung des Genre-Lexikons verwenden. So sind Wörter vor dem Wort Film in einem Satz, sofern vorhanden und nicht Teil eines Filmtyps, häufig Genrebezeichner. Ebenso bietet IMDb auch Informationen zu Filmgenres, die extrahierbar sind.

### 3.8.2 Transducer-Grammatiken

Wie bereits in 3.6.2 angedeutet, wäre die Erweiterung der Grammatiken um mehr Transducer-Regeln zur Verbesserung der Annotation denkbar. Dadurch kann eine Reduzierung von fehlerhaften Titeln in den endgültigen Relationsmengen erreicht werden, indem Fehltaggings innerhalb der Titel-Objekte bereinigt werden. Für eine solche Verbesserung sind jedoch entweder bezüglich der Laufzeit effizientere Grammatiken oder ein leistungsstärkeres Arbeitssystem erforderlich.

Zur Verkürzung der Laufzeit bestünde die Option, der Grammatik zur Bereinigung der Titel nur die ersten 50 oder 100 Zeichen zu übergeben, da davon auszugehen ist, dass relevante Titel-Objekte meistens in diesem Bereich vorkommen.

#### Relationen

In Bezug auf die Relationen ibt es selbstverständlich immer die Möglichkeit, mehr Angaben aufzunehmen. So enthalten Artikel zu Filmen oft Details wie den Originaltitel, das produzierende Studio oder den Regisseur. Weiterhin könnte ebenso über kurze Informationen zur Handlung des Films als Relation nachgedacht werden.

Zusätzlich ist es möglich, durch die Anwendung von anderen computerlinguistischen Verfahren mit kontext-sensitiver Extraktion zu arbeiten, um der Entnahme von falschen Informationen entgegen zu wirken.

# 4 Schlusswort

Ziel der präsentierten Arbeit war die Extraktion von relationalen Informationen zu Filmen aus einem Wikipedia-Textkorpus. Zur Realisierung dieses Vorhabens erfolgte die Verwendung von regelbasierten Methoden, um Klassifikations-, Annotations- und Extraktionsaufgaben zu lösen.

Die Vorbereitung dafür bestand in der geforderten Kürzung des Textkorpus und der Beschaffung der für die Generierung von Regeln erforderlichen Daten. Ergebnis waren ein Korpus von Artikeleinleitungen und zwei Datensätze, einmal für Genre-Begriffe und ein Weiterer mit Schlagwörtern.

Der erste Arbeitschritt war die Lösung der Klassifikationsaufgabe. Zu diesem Zweck wurde mit Hilfe der gesammelten Daten ein regelbasierter Klassifikator in Python entwickelt, der anhand von 4 Komponenten prüft, ob der Artikel einen Film beschreibt. Aus diesem Prozess resultierte eine Datei mit allen Film-Artikeln, sowie eine sparate Liste der Titel. Um die Annotationsaufgabe zu lösen, mussten zunächst weitere Datensätze aufgebaut werden, wozu der Autor sowohl das Internet, als auch eigenes Fachwissen nutzte. Die resultierenden Ersetzungswörterbücher wurden mit Hilfe der plmlm-Technik in Transducer umgewandelt und danach durch ein Python-Programm automatisiert auf die Texte angewandt. Zur Finalisierung der Vorannotation kamen zusätzlich selbst entwickelte Transducer-Grammatiken zur Bereinigung der Titel- und Genre-Objekte zum Einsatz, deren Anwendung in selber Art automatisiert werden konnte. Aus dem Resultat, einer komplett annotierten Version der Film-Artikel-Datei, ließen sich nun die gesuchten Relationen mittels eines weiteren Python-Programms extrahieren und zusammensetzten. Damit lag das Endergebnis der Arbeit in Form einer Liste von Relationsmengen vor.

Eine Evalutation der Ergebnisse ergab zwar überzeugende Ergebnisse im Bereich der Annotation und Extraktion der einzelnen relationalen Informationen, die kombinierten Relationsmengen zu den Filmen sind jedoch nur in knapp 60% der Fälle vollständig korrekt. Durch die Zusammensetzung dieser aus fünf einzelnen Elementen mit durchschnittlich 10% Fehlerquote entsteht eine vergleichsweise hohe Wahrscheinlichkeit für zumindest einzelne Fehler in den Mengen. Die folgende Fehleranalyse zeigte, dass Fehler bei der Annotation hauptsächlich durch ambige Einträge in den Lexika sowie seltener aufgrund fehlender Einträge zustande kommen. Der zweite Fall ist jedoch nicht immer korrigierbar, da in manchen Fällen die Inklusion von Begriffen aufgrund der Mehrdeutigkeit zu mehr neuen Fehlern führt, als dadurch vermieden werden können. Bei der Extraktion hingegen ist das Hauptproblem die fehlende Kontextsensitivität, wodurch Informationen aus unpassenden Zusammenhängen extrahiert werden. Die Folge sind falsche Informationen in Bezug auf den eigentlichen Kontext der Relation.

Aus den Ergebnissen lässt sich schließen, dass sich mit der regelbasierten Informationsextraktion nach den vorgestellten Verfahren vielversprechende Ergebnisse in relativ kurzer Zeit erarbeiten lassen. Hauptvorteil ist hier die Möglichkeit, Fachwissen direkt als Kriterien der Annotation zu nutzen, anstatt in langwieriger Vorarbeit Trainingsdaten zu erstellen. Dennoch besteht Verbesserungspotential: Gerade die Einbindung von Kontextsensitivität bei der Extraktion der Information könnte zu deutlichen Verbesserungen führen, bietet also Stoff für weitere Arbeiten im vorgestellten Bereich.

# Literaturverzeichnis


[1] 3.3 regular languages and relations. Website - `http://www.cs.union.edu/~striegnk/courses/nlp-with-prolog/html/node22.html`. abgerufen am 29.11.2019.

[2] Lingolia - Ländernamen und Nationalitäten auf Deutsch. Website - `https://deutsch.lingolia.com/de/wortschatz/laender-nationalitaeten`, 2019. abgerufen am 10.11.2019.

[3] Press room - imdb statistics. Website - `https://www.imdb.com/pressroom/stats/`, 2019. abgerufen am 28.11.2019.

[4] Wikipedia - Filmgenre. Website - `https://de.wikipedia.org/wiki/Filmgenre`, 2019. abgerufen am 01.11.2019.

[5] Wikipedia - Kategorie:Filmgenre. Website - `https://de.wikipedia.org/wiki/Kategorie:Filmgenre`, 2019. abgerufen am 03.11.2019.

[6] Mustafa Abdul Salam. Sentiment analysis of product reviews using bag of words and bag of concepts. *International Journal of Electronics*, 11:49–60, 12 2019.

[7] Georg Bonczek und Florian Matthes Bernhard Waltl. Rule-based information extraction: Advantages, limitations, and perspectives. *Jusletter IT 22*, 02 2018.

[8] J. Cowie and Yorick Wilks. Information extraction. *Handbook of Natural Language Processing*, pages 241–260, 01 2000.

[9] Yves Schabes Emmanuel Roche. *Introduction to Finite-State Devices in Natural Language Processing*. Mitsubishi Electric Research Laboratories, Inc., 1996.

[10] Andrea Esuli and Fabrizio Sebastiani. Evaluating information extraction. pages 100–111, 09 2010.

[11] Eric Goutte, Cyril und Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. volume 3408, pages 345–359, 04 2005.

[12] Craig Hagerman. Evaluating the performance of automated part-of-speech taggers on an l2 corpus. 10 2011.

[13] Hopcroft, John E., Motwani, Rajeev, Ullman, and Jeffrey D. *Introduction to Automata Theory, Languages, and Computation, 3rd Edition*. Pearson Education, Inc., 01 2007.

[14] Thorsten Joachims. Text categorization with support vector machines. *Proc. European Conf. Machine Learning (ECML'98)*, 01 1998.

[15] Ronald Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20:331–378, 01 1994.

[16] L. Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, and A. Schille. Regular expressions for language engineering. *Natural Language Engineering*, 2:305–328, 1996.

[17] Stoyan Mihov and Klaus U. Schulz. *Finite-State Techniques - Automata, Transducers and Bimachines*. Cambridge University Press, 2018.

[18] Marcos Mouriño-García, Roberto Perez-Rodriguez, and Luis Anido-Rifón. Bag-of-concepts document representation for textual news classification. *International Journal of Computational Linguistics and Applications*, 6:173–188, 06 2015.

[19] Patrick Paroubek. *Evaluating Part-of-Speech Tagging and Parsing Patrick Paroubek*, pages 99–124. Springer Netherlands, Dordrecht, 2007.

[20] Klaus U. Schulz. Informationen zur regelbasierten informationsextraktion aus der wikipedia. Website - `https://www.cis.uni-muenchen.de/people/Schulz/bachelor/ReadMeWikiExtraktion.txt`. Abgerufen am 17.09.2019.

[21] Ewan Klein und Edward Loper Steven Bird. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[22] V. K. Balasubrahmanyan und S. Naranan. Algorithmic information, complexity and zipf's law. pages 1–26, 2002.

[23] Peipei Wang, Gina R. Bai, and Kathryn T. Stolee. Exploring regular expression evolution. *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 502–513, 2019.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Inhalt der beigelegten CD

- Alle im Verlauf der Bachelorarbeit entwickelte Programme in den Ordnern Python-Code und Bash-Script
- Die verwendeten Text-Ressourcen im Ordner Ressourcen, inklusive
  - Erstellte Lexika
  - Gekürzter Korpus
  - Film-Artikel-Datei und Titelliste
  - Ergebnis-Dateien
- Eine Kopie der Ergebnisse im Ordner Ergebnisse
- Die verwendeten Werkzeuge inklusive der selbst verfassen Transducer-Grammatiken im Ordner Tools
- Die genutzten Webseiten im Ordner Literatur
- Die aus den Lexika erstellten Transducer im Ordner Transducer
- Die schriftliche Arbeit als PDF sowie LaTex-Datei sowie benötigte Dateien und Bilder im Ordner Schriftliche Arbeit
- Eine Readme-Datei