Thema:

Aufbau und Realisierung einer Software zur Konvertierung von L^AT_EX-Dokumenten, in einem XML-Standard

Diplomarbeit

Arbeitsgruppe Managementinformationssysteme

Themensteller: Prof. Dr. rer. pol. habil. Hans-Knud Arndt
Betreuer: Dipl.-Kfm. Henner Graubitz

vorgelegt von: Mathias Kant

Abgabetermin: 19. April 2009

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

AA	Action Aiders
ACT	Acts
AI	Artificial Intelligence
API	Application Programming Interface
BDM	Backward Dawg Matching
BM	Boyer-Moore
CASE	Computer-Aided Software Engineering
CDT	Conceptual Dependency Theory
CGM	Computer Graphics Metafile
CSS	Cascading Style Sheets
CSV	Comma separated values
DB	Datenbank
DBMS	Datenbankmanagementsystem
DFA	Deterministic Finite Automaton
DM	Data Mining
EPK	Ereignisgesteuerte Prozesskette
FIN	Fakultät für Informatik
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HWPF	Horrible Word Processor Format
IE	Information Extraction
IEF	Image Exchange Format
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
JDBC	Java Database Connectivity
JPEG	Joint Photographic Expert Group
JRE	Java Runtime Environment
KMP	Knuth-Morris-Pratt
LOC	Location
MathML	Mathematical Markup Language
MPEG	Moving Picture Experts Group
MUC	Message Understanding Conference
NATO	North Atlantic Treaty Organization
NFA	Nondeterministic Finite Automaton
ODF	Open Document Format
OLAP	On-Line-Analytical-Processing
OOXML	Office Open XML
PA	Picture Aiders
PDF	Portable Document Format
PM	Pattern Matching
PNG	Portable Network Graphics
PP	Picture Producers
RTF	Rich Text Format
SC	Similarity Coefficient
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SMV	Selectable Mode Vocoder

SVG	Scalable Vector Graphics
T	Time
TIFF	Tagged Image File Format
URE	Uno Runtime Environment
URI	Uniform Resource Identifier
VML	Vector Markup Language
VMS	Virtual Memory System
WYSIWYG	What You See Is What You Get
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
ZIP	zig-zag in-line package

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Mit einer zunehmenden Informationsflut in der heutigen Gesellschaft wird die Extraktion von Informationen immer bedeutender. Immer mehr Daten werden in digitaler Form bereitgestellt. Durch unstrukturierte und heterogene Datenbestände können die Daten manuell durch den Nutzer nicht mehr in einer annehmbaren Zeit analysiert und ausgewertet werden. Daher gibt es eine Vielzahl von Methoden und Programmen, um Informationen und Daten aus Dokumenten oder dem Internet mit einem möglichst geringen Aufwand und einem möglichst exaktem Ergebnis zu extrahieren. In diesem Zusammenhang behandelt die Arbeit die verschiedenen Methoden zum Auslesen und Verarbeiten von Informationen aus unterschiedlichsten Quellmedien.

1.2 Aufbau der Arbeit

In *Kapitel 2* wird anhand der Zielsetzung für die zu entwickelnde Software die grundlegende Problemstellung verdeutlicht.

Zur Extraktion von Informationen werden im *dritten Kapitel* die verschiedenen Verfahren *Information Retrieval*, *Data Mining*, *Pattern Matching* und *Information Extraction* vorgestellt und analysiert. Des Weiteren werden die ISO-Standards *ISO/IEC 26300* und *ISO/IEC 29500* als mögliche Ausgabeformate eingeführt und miteinander verglichen. Anschließend werden Beispielprogramme zum Konvertieren von T_EX- in Open Office-Dokumente vorgestellt und auf der Basis der vorher definierten Anforderungen untersucht und bewertet.

Das folgende *vierte Kapitel* stellt den Status quo bei der Erstellung des Jahresberichts der Fakultät für Informatik, der als Beispiel für die Problemstellung der Arbeit dient, vor. Aufbauend auf Interviews, deren Durchführung kurz beschrieben wird, werden der aktuelle Ablauf der Erstellung und dessen Vorteile und Nachteile, die durch eine Schwachstellenanalyse erörtert werden, herausgestellt. Anschließend werden die Anforderungen an das neue System durch eine Anforderungsanalyse erörtert.

Der grundlegende Aufbau des neuen Systems wird im *Kapitel 5* vorgestellt. Dabei werden der neue Ablauf mit den neuen Eingabemöglichkeiten, die benötigten Technologien sowie dessen Vor- und Nachteile beschrieben. Zum Ende des Kapitels wird das neue mit dem alten System verglichen und ausgewertet.

Im abschließenden *sechsten Kapitel* wird anhand der Problemdefinition die Lösung des Problems bewertet und zusammengefasst. Außerdem wird ein Ausblick für die zukünftige Nutzung und Verbesserungsmöglichkeiten gegeben.

In Abbildung 1.1 wird der Aufbau der Arbeit zur besseren Übersicht graphisch verarbeitet.

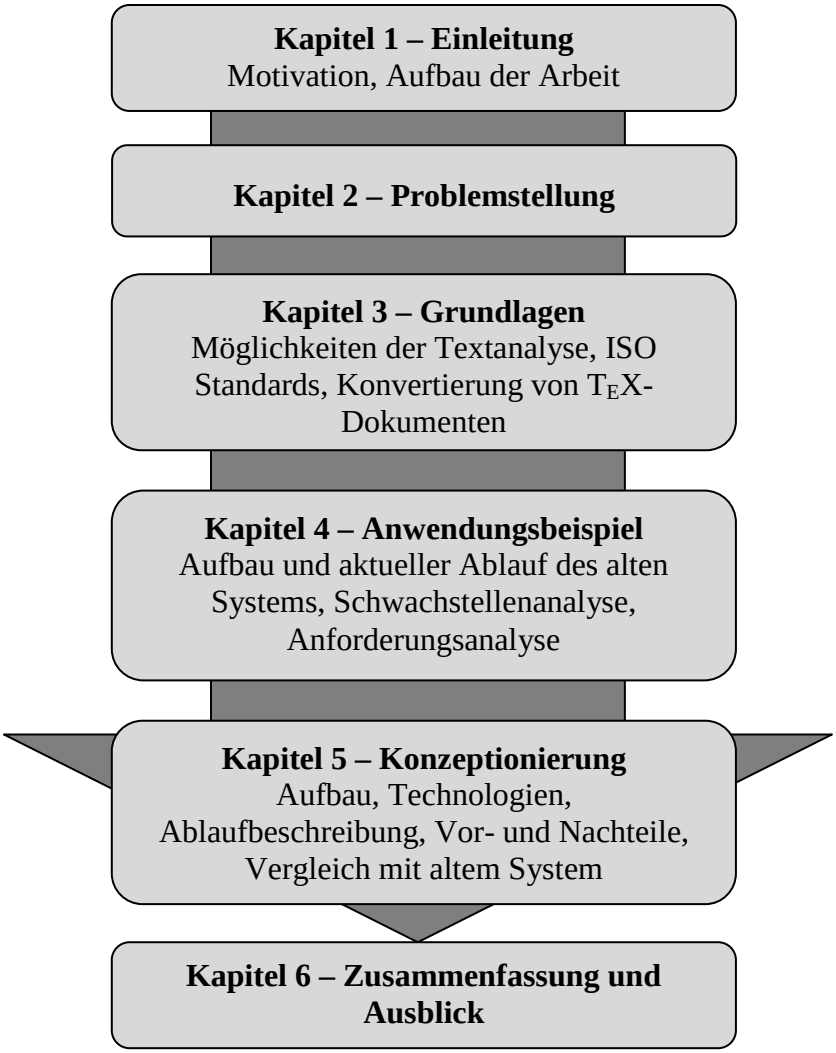


Abbildung 1.1: Aufbau der Arbeit

2 Problemstellung

In dieser Arbeit wird das Problem der Extraktion von Informationen aus Dokumenten am Beispiel des Jahresberichts der Fakultät für Informatik an der Otto-von-Guericke-Universität behandelt. In dem Jahresbericht werden alle wichtigen Informationen zu der Fakultät und deren Institute übersichtlich aufbereitet. Das aktuelle System zur Erstellung des Berichts birgt jedoch eine Reihe von Fehlerquellen. Es werden weitestgehend vorbereitete T_EX-Masken zur Eingabe der Informationen durch die Mitarbeiter verwendet. Andere Formate wie z. B. Word-Dokumente oder einfache Text-Dateien werden nur in Ausnahmefällen akzeptiert. Um die Korrektheit und die Vollständigkeit der Dokumente zu sichern, ist es notwendig, dass die Mitarbeiter den Inhalt und die Syntax der ausgefüllten T_EX-Masken überprüfen. In dieser Arbeit gilt es herauszufinden, warum das aktuelle System nicht optimal funktioniert. In diesem Zusammenhang wird dieses analysiert, um die Eigenschaften bzw. Anforderungen für das neue System herauszukristallisieren. Bei der Anfertigung des Berichts sollen verschiedene Dokumenttypen eingelesen und die Informationen anschließend in einem Format, basierend auf einem ISO-Standard, ausgegeben werden.

Ziel der Arbeit ist die Weiterentwicklung des bestehenden Systems zur Erstellung des Berichts, da bei dieser Verbesserungspotentiale, insbesondere bei den Eingabe-möglichkeiten und dem automatischen Einlesen, erkannt wurden. Durch die Verbesserungen im Aufwand und in der Bedienerfreundlichkeit werden alle an der Erstellung beteiligten Mitarbeiter von dem neuen System profitieren.

Bei der Entwicklung des neuen Systems stehen folgende Fragen im Mittelpunkt:

- Welche Dokumentenformate sollen zur Eingabe möglich sein?
- Welcher ISO-Standard soll die Grundlage des Ausgabedokuments sein?
- Welche Methoden zur Textanalyse werden verwendet?

3 Grundlagen

In diesem Kapitel werden eine Reihe möglicher Methoden und Verfahren zur Textextraktion erörtert und auf ihre Relevanz bezüglich des Problems geprüft. Anschließend werden der *ISO/IEC-Standard 26300* (*Open Document Format for Office Application*) und der *ISO/IEC-Standard 29500* (*Office Open XML File Formats*) vorgestellt. In diesem Zusammenhang werden diese analysiert, verglichen und die Vor- und Nachteile heraus gearbeitet. Danach wird auf das Thema des Auslesens und Umwandeln von T_EX- in Open Office-Dokumente eingegangen, wobei verschiedene Programme, die die Umwandlung ermöglichen, nach vorher definierten Anforderungen untersucht und bewertet werden.

3.1 Möglichkeiten der Textanalyse

Zur Extraktion von Informationen aus Dokumenten oder anderen Informationsquellen existieren eine Vielzahl von Methoden und Verfahren. Es gibt verschiedene Ansätze für das Problem der Extraktion. In diesem Kapitel wird eine Auswahl an Techniken – *Information Retrieval*, *Data Mining*, *Pattern Matching* und *Information Extraction* – vorgestellt.

3.1.1 Information Retrieval

Eine weit verbreitete Methode zum Filtern von relevanten Informationen aus vielen verschiedenen Dokumenten ist das *Information Retrieval*. In dem Paper „Readings in Information Retrieval“ wird diese Methode als das Suchen von Dokumenten oder Text mit Informationsgehalt, der relevant für die Informationsbedürfnisse des Nutzers ist, bezeichnet. [Jones; Willett 1997] *Information Retrieval* beinhaltet hauptsächlich die Repräsentation, Speicherung, Organisation von und den Zugang zu Informations-elementen. Bei der Repräsentation und Organisation der Informationen soll dem Nutzer der Zugang zu den für ihn relevanten Inhalten erleichtert werden. Das Einschätzen, welche Informationen der Nutzer benötigt, ist kein triviales Problem. Zunächst muss der Nutzer den Informationswunsch in eine Anfrage übersetzen, die anschließend von einer Suchmaschine bzw. einem *Information Retrieval*-System verarbeitet wird. Bei der Übersetzung entsteht eine Reihe von Schlüsselwörtern, die den Informationswunsch zusammenfassen. Das wichtigste Ziel des *Information Retrieval* ist das Abfragen und Herausfiltern von, für den Nutzer, relevanten oder nützlichen Informationen. Die Basis dieser Methode ist hauptsächlich Text der natürlichen Sprache, der oft unstrukturiert ist und semantisch mehrere Bedeutungen haben kann. [Baeza-Yates; Ribeiro-Neto 1999, S. 1-2]

In den letzten 20 Jahren ist die Forschung auf dem Gebiet des *Information Retrieval* stark ausgeweitet worden. Diese geht über das primäre Ziel der Indexierung der Texte und dem Suchen nützlicher und relevanter Dokumente aus einer großen Sammlung hinaus. Bei der Erforschung des *Information Retrieval* werden beispielsweise die Themen Modellierung, Dokumentenklassifikation und –kategorisierung, System-architektur, Benutzeroberfläche, Datenvisualisierung, Filterung und Sprache behandelt. Zu Beginn wurde *Information Retrieval* noch als ein abgegrenztes Gebiet überwiegend für Bibliothekare und Informationsexperten angesehen. Diese Ansicht blieb bis zur Einführung des World Wide Web bestehen, das als universeller Speicher für das Wissen der Menschen dient und das Teilen von Informationen in einem sehr großen Ausmaß unterstützt. Durch die große Menge an Informationen im Internet und das somit aufwendigere Finden von nützlichen Informationen gewann das *Information Retrieval* immer mehr an Bedeutung. [Baeza-Yates; Ribeiro-Neto 1999, S. 2]

3.1.1.1 Information Retrieval-Modelle

Es existiert zurzeit, wie in Abbildung 3.1 zu sehen ist, eine Reihe von verschiedenen Modellen zum *Information Retrieval*. Bevor auf die Modelle eingegangen wird, werden die Grundlagen und die Voraussetzungen für diese beschrieben. Am Anfang muss eine Text-Datenbank initialisiert werden. Diese beinhaltet zum einen die Dokumente und die Operationen, die auf die Texte angewendet werden und zum anderen das Textmodell, das die Textstruktur und die Elemente, die abgefragt werden können, umfasst. Die Textoperationen transformieren die Originaldokumente und generieren eine logische Sicht auf diese. Anschließend ist es nötig, jeweils einen Index pro Text zu erstellen, der das Dokument durch eine Sammlung von repräsentativen Schlüsselwörtern beschreibt. Diese Schlüsselwörter sind meistens einfache Wörter, die semantisch einen Teil des Hauptthemas des Dokuments beschreiben und damit zusammenfassen. Es gibt geeignete und weniger geeignete Wörter, um ein Dokument zu beschreiben. Für diese Wörter werden zum großen Teil Substantive verwendet, da sie selbst eine Bedeutung haben und dadurch semantisch leichter zu identifizieren sind. Adjektive und Adverbien sind dagegen wenig nützlich, da sie meistens nur als Ergänzung anderer Wörter dienen. Zusätzlich sind Wörter, die in vielen Dokumenten vorkommen, dabei weniger brauchbar als solche, die nur in einigen Dokumenten enthalten sind. Um die meist große Menge an Dokumenten nach relevanten und nicht relevanten Dokumenten zu klassifizieren, stehen verschiedene mögliche Modelle zur Verfügung. In dieser Arbeit wird speziell auf das *Boolesche Modell*, das *Vektorraummodell* und das *probabilistische Modell* eingegangen. [Baeza-Yates; Ribeiro-Neto 1999, S. 24]

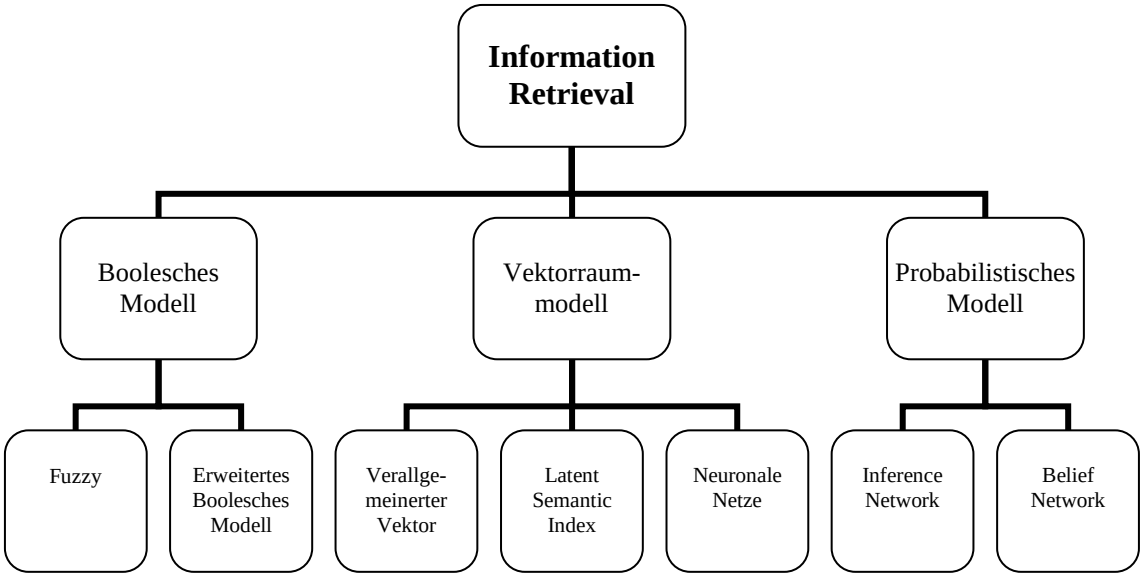


Abbildung 3.1: Information Retrieval Modelle

Boolesches Modell

Das Konzept des *Booleschen Modells* ist intuitiv und wird häufig bei kommerziellen *Information Retrieval*-Systemen verwendet. Jede Anfrage des Nutzers ist spezifiziert durch boolesche Ausdrücke. Diese können, wie in Abbildung 3.2 zu sehen ist, in einem Venn-Diagramm grafisch dargestellt werden. Die Strategie des Modells basiert auf binären Entscheidungskriterien – relevant oder nicht relevant – ohne Verwendung jeglicher Abstufungsskalen. Boolesche Ausdrücke haben eine eindeutige Semantik. Es ist jedoch schwierig, einen Informationswunsch in einen booleschen Ausdruck zu übersetzen. Durch diese Schwierigkeit sind die meisten Anfragen durch boolesche Ausdrücke sehr einfach gehalten. [Baeza-Yates; Ribeiro-Neto 1999, S. 25-27]

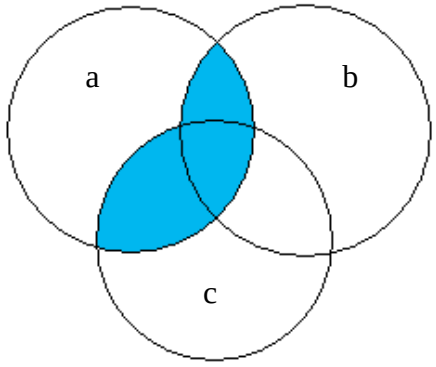


Abbildung 3.2: Bildliche Darstellung der booleschen Anfrage $q = a \cap (b \cup c)$

In einigen *Information Retrieval*-Systemen geben die Nutzer die booleschen Ausdrücke direkt ein. In aufwendigeren Systemen können die Anfragen in natürlicher Sprache eingegeben werden und das System übersetzt die Eingabe in die entsprechenden

booleschen Ausdrücke. [Frakes, Baeza-Yates 1992, S. 265] Anschließend wird geprüft, ob das Schlüsselwort in einem Dokument enthalten ist oder nicht. Falls ein Schlüsselwort in einem Dokument enthalten ist, ist das Dokument automatisch relevant. Es können keine partiellen Übereinstimmungen bei diesem Modell auftreten. Ein großer Vorteil des Modells ist, dass es auf einem sauberen Formalismus basiert und dadurch relativ intuitiv und einfach ist. Dagegen ist ein Nachteil, dass durch die exakten Übereinstimmungen entweder nur wenig bis keine oder zu viele Dokumente angezeigt werden können. [Baeza-Yates; Ribeiro-Neto 1999, S. 25-27] Ein weiterer Kritikpunkt ist, dass das *Boolesche Modell* nur eine ungeordnete Menge an Dokumenten liefert. Es wird kein Ranking vorgenommen, wodurch keine Bewertung in mehr oder weniger relevant getroffen wird. [Ferber 2003, S. 33] In Tabelle 3.1 und Tabelle 3.2 werden zwei kleine Beispiele zum besseren Verständnis vorgestellt.

Tabelle 3.1: Beispiel 1 zum Boolesches Modell

Anfrage:	*Finde alle Dokumente, die „Information Retrieval" beinhalten.*
Boolescher Ausdruck:	*„Information Retrieval"*
Ergebnis:	*Alle Dokumente, die die Wortgruppe „Information Retrieval" enthalten.*

Bei den booleschen Ausdrücken kann ein *AND,* ein *OR* oder ein *XOR* benutzt werden. Bei einem *AND* müssen beide Schlüsselwörter zusammen in dem Dokument vorkommen. Wie in Tabelle 3.2 zu sehen ist, werden bei einem *OR* alle Dokumente ausgegeben, die beide oder auch nur eines der beiden Schlüsselwörter enthalten. Wenn ein *XOR* verwendet wird, muss das eine oder das andere Schlüsselwort, aber nicht beide gemeinsam, vorkommen.

Tabelle 3.2: Beispiel 2 zum Booleschen Modell

Anfrage:	*Finde alle Dokumente, die „Information" oder „Retrieval" beinhalten.*
Boolescher Ausdruck:	*„Information" OR „Retrieval"*
Ergebnis:	*Alle Dokumente, die „Information" und alle Dokumente die „Retrieval" beinhalten. Und alle Dokumente die beide Schlüsselwörter beinhalten.*

Formal kann das Anfrageergebnis wie nachfolgend beschrieben werden. U bezeichnet die Namen aller Dokumente und $D1$ und $D2$ stellen die Dokumente dar, die $P1$ und $P2$ (Schlüsselwörter) enthalten. [Frakes, Baeza-Yates 1992, S. 267-268]

$U \neg D1$ → Dokumente, die *nicht* $P1$ enthalten (NOT)

$D1 \cap D2$ → Dokumente, die $P1$ *und* $P2$ enthalten (AND)

$D1 \cup D2$ → Dokumente, die $P1$ *oder* $P2$ enthalten (OR)

$(D1 \cup D2) \neg (D1 \cap D2)$ → Dokumente, die $P1$ *oder* $P2$ enthalten aber *nicht* beide zusammen (XOR)

Vektorraummodell

Die Praxis hat gezeigt, dass eine binäre Gewichtung wie beim *Booleschen Modell* zu begrenzt ist. Bei dem *Vektorraummodell* werden zusätzlich partielle Übereinstimmungen erlaubt. Dies ist durch eine nicht-binäre Gewichtung der Indizes in Anfragen und in den Dokumenten möglich. Wie in Abbildung 3.3 zu sehen ist, werden hierbei die Anfrage und jedes Dokument durch Vektoren repräsentiert. Zur Eruierung der relevanten Dokumente wird der Grad der Ähnlichkeit – je höher dieser Grad ist, umso relevanter ist das Dokument – zwischen jedem Dokument- und dem Anfrage-Vektor errechnet. Anschließend werden die Ergebnisse absteigend nach dem Grad der Ähnlichkeit sortiert. Im Gegensatz zum *Booleschen Modell* werden auch Dokumente berücksichtigt, die nur teilweise mit der Anfrage übereinstimmen. Die Indizes können auf verschiedene Art und Weise berechnet werden. [Baeza-Yates; Ribeiro-Neto 1999, S. 27] [Grossmann, Frieder 2004, S. 11]

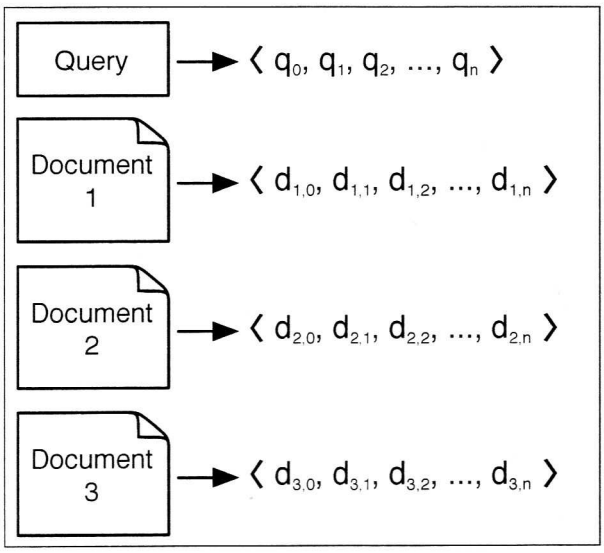


Abbildung 3.3: Vektorraummodell [Grossmann, Frieder 2004, S. 13]

Eine Möglichkeit ist, einen Differenzvektor aus den zwei Vektoren – Anfrage und Dokument – zu bilden. Das Problem an dieser Methode ist, dass große Dokumente bei vielen Anfragen als nicht relevant eingestuft werden, da viele Anfragen sehr kurz sind. In den meisten Fällen wird der Winkel zwischen zwei Vektoren zur Bestimmung der Ähnlichkeit berechnet. In diesem Zusammenhang wird das *Skalarprodukt*[1] der beiden Vektoren gebildet. Das Ergebnis dieses Vergleichs wird *Similarity Coefficient* (*SC*)

[1]Das Skalarprodukt zweier Vektoren $\vec{x}$ und $\vec{y}$ berechnet sich nach der Formel $\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}| \cos \angle(\vec{x}, \vec{y})$. Dabei sind $|\vec{x}|$ und $|\vec{y}|$ jeweils die Längen der Vektoren. Mit $\cos \angle(\vec{x}, \vec{y})$ wird der Kosinus des Winkels zwischen den beiden Vektoren bezeichnet.

genannt. Dieses Maß gibt die Ähnlichkeit zwischen der Anfrage und jedem einzelnen Dokument an. [Grossmann, Frieder 2004, S. 11] Anhand des folgenden Beispiels soll die Berechnung des *Similarity Coefficient* näher gebracht werden.

Variablen:

- t_j – j-ter Index in der Dokumentensammlung
- tf_{ij} – Anzahl des Auftreten von Index t_j in Dokument D_i - Indexfrequenz
- df_j – Anzahl der Dokumente die t_j enthalten – Dokumentenfrequenz
- $w_{Q,j}$ – Vektor für die Anfrage
- d – absolute Anzahl der Dokumente

Formeln:

$idf_j = \log(\frac{d}{df_j})$ → inverse Dokumentenfrequenz

$d_{ij} = tf_{ij} \cdot idf_j$ → Kombination aus Indexfrequenz und inverser Dokumentenfrequenz

$SC(Q, D_i) = \sum_{j=1}^{|t|} w_{Q,j} \cdot d_{ij}$ → *Similarity Coefficient*

Beispiel:

Anfrage Q = *„Vektorraummodell Retrieval“*

D_1 = *„Ein Modell des Information Retrieval ist das Boolesche Modell.“*

D_2 = *„Information Retrieval: Vektorraummodell ist eines von vielen Modellen des Information Retrieval.“*

D_3 = *„Ein anderes Modell ist das Vektorraummodell.“*

Auswertung:

d = 3 → Dokumentenanzahl

$idf_{ein} = \log(3 / 2) \approx 0.176$
$idf_{Modell} = \log(3 / 3) \approx 0$
$idf_{des} = \log(3 / 3) \approx 0$
$idf_{Retrieval} = \log(3 / 2) \approx 0.176$
$idf_{ist} = \log(3 / 3) \approx 0$
$idf_{das} = \log(3 / 2) \approx 0.176$
$idf_{Vektorraummodell} = \log(3 / 2) \approx 0.176$
$idf_{eines} = \log(3 / 1) \approx 0.477$
$idf_{von} = \log(3 / 1) \approx 0.477$
$idf_{vielen} = \log(3 / 1) \approx 0.477$
$idf_{boolesche} = \log(3 / 1) \approx 0.477$
$idf_{anderes} = \log(3 / 1) \approx 0.477$

Tabelle 3.3: Dokumenten-Vektoren des Beispiels

	ein	Modell	des	Retrieval	ist	das	Vektor-raum-modell	eines	von	vielen	boolesche	anderes
d_{ij}												
D_1	0.176	0	0	0.176	0	0.176	0	0	0	0	0.477	0
D_2	0	0	0	0.176	0	0	0.176	0.477	0.477	0.477	0	0
D_3	0.176	0	0	0	0	0.176	0.176	0	0	0	0	0.477
$w_{Q,j}$												
Q	0	0	0	0.176	0	0	0.176	0	0	0	0	0

$SC(Q,D_1) = (0 \cdot 0.176) + (0 \cdot 0) + (0 \cdot 0) + (0.176 \cdot 0.176) + (0 \cdot 0) + (0 \cdot 0.176) + (0.176 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0) + (0 \cdot 0.477) + (0 \cdot 0) = 0.176^2 \approx 0.031$

$SC(Q,D_2) = 0.176^2 + 0.176^2 \approx 0.062$

$SC(Q,D_3) = 0.176^2 \approx 0.031$

Den vorangestellten Berechnungen ist zu entnehmen, dass das Dokument D_2 das relevanteste, gefolgt von D_1 und D_3 – beide mit gleicher Relevanz –, ist. Alternativen zu dem *Skalarprodukt* sind das *Cosinus-Maß*, das *Dice-Maß* oder das *Jaccard-Maß*. An dieser Stelle wird jedoch nicht näher auf die Alternativen eingegangen.

Ein Vorteil des *Vektorraummodells* ist zum Einen die Verbesserung der Anfrageperformance durch das Gewichtungsschema der Indizes. Ein anderer ist, dass partielle Übereinstimmungen erlaubt sind und diese eine Annäherung an die Anfrage möglich machen. Die gerankten Antworten sind jedoch schwer zu verbessern. Dies kann nur durch eine Erweiterung der Anfrage oder dem *Relevance Feedback*, das später im Kapitel 3.1.1.2 näher erläutert wird, erfolgen. [Baeza-Yates; Ribeiro-Neto 1999, S. 30]

Probabilistisches Modell

In dem *probabilistischen Modell* werden wahrscheinlichkeitstheoretische Ansätze verwendet. Das bedeutet, dass die Wahrscheinlichkeit des Auftretens eines Index der Anfrage in einem relevanten Dokument berechnet wird. Für jeden Index, bei dem die Anfrage mit einem Dokument übereinstimmt, wird die Ähnlichkeit als Kombination der Wahrscheinlichkeiten berechnet. Der *Similarity Coefficient* zwischen Dokument und Anfrage ist hierbei die Wahrscheinlichkeit, inwieweit das Dokument relevant bezüglich der Anfrage ist. Zu Beginn ist eine Nutzeranfrage gegeben, zu der eine Dokumentenmenge existiert, in der alle relevanten Dokumente entsprechend der

Anfrage enthalten sind. Diese wird als ideale Antwortmenge bezeichnet. Wenn die Beschreibung der Menge vorhanden wäre, würden alle relevanten Dokumente leicht gefunden werden. Aus diesem Grunde kann der Anfrageprozess als ein Prozess der Spezifizierung der Eigenschaften der idealen Antwortmenge gesehen werden. Diese Eigenschaften sind jedoch zum Anfang der Anfrage nicht bekannt und müssen daher geschätzt werden. Damit wird eine vorübergehende Wahrscheinlichkeitsbeschreibung der idealen Antwortmenge generiert. Anschließend gibt der Nutzer an, welche Dokumente relevant sind und welche nicht. Mit diesen Zusatzinformationen kann das System die Anfrage verfeinern und sich nach mehreren Durchläufen der Beschreibung der idealen Antwortmenge annähern. In den folgenden Formeln sei R die ideale Antwortmenge, Q die Anfrage und D_i ein Dokument. [Baeza-Yates; Ribeiro-Neto 1999, S. 30-31] [Grossmann, Frieder 2004, S. 21]

$$SC(Q, D_i) = \frac{P(R \mid \vec{D}_i)}{P(\overline{R} \mid \vec{D}_i)}$$

Diese Formel berechnet die Ähnlichkeit eines Dokuments D_i zu der Anfrage Q. Unter Benutzung der *Bayesschen Regel* ergibt sich folgende Formel:

$$SC(Q, D_i) = \frac{P(\vec{D}_i \mid R) \cdot P(R)}{P(\vec{D}_i \mid \overline{R}) \cdot P(\overline{R})}$$

$P(\vec{D}_i \mid R)$ ist die Wahrscheinlichkeit, dass ein Dokument D_i zufällig aus der Menge der relevanten Dokumente ausgewählt wird. $P(R)$ symbolisiert die Wahrscheinlichkeit, dass ein zufällig ausgewähltes Dokument aus der gesamten Menge aller Dokumente relevant ist. $\overline{R}$ ist die Menge aller Dokumente, die nicht in R enthalten sind. Damit sind die Wahrscheinlichkeiten $P(\vec{D}_i \mid \overline{R})$ und $P(\overline{R})$ analog der obigen zu verstehen. [Baeza-Yates; Ribeiro-Neto 1999, S. 32]

Unter der Annahme, dass die Indizes unabhängig sind, kann die folgende Formel verwendet werden:

$$SC(Q, D_i) = \sum_{j=1}^{|t|} w_{Q,j} \cdot w_{i,j} \cdot \left(\log \frac{P(t_j \mid R)}{1 - P(t_j \mid R)} + \log \frac{1 - P(t_j \mid \overline{R})}{P(t_j \mid \overline{R})} \right)$$

$w_{Q,j}$ und $w_{i,j}$ sind binäre Indexgewichte für die Dokumente bzw. die Anfrage. $P(t_j \mid R)$ kennzeichnet die Wahrscheinlichkeit, dass Index t_j in einem zufällig ausgewählten Dokument aus der idealen Antwortmenge R vorhanden ist. Dabei gilt $P(t_j \mid R) + P(t_j \mid \overline{R}) = 1$. [Baeza-Yates; Ribeiro-Neto 1999, S. 32]

Um $P(t_j \mid R)$ und $P(t_j \mid \overline{R})$ zu Beginn zu berechnen, gibt es mehrere Möglichkeiten. Zur Vereinfachung kann angenommen werden, dass $P(t_j \mid R)$ konstant – typischerweise der Mittelwert 0.5 – für alle Indizes t_j ist. Bei der zweiten Möglichkeit wird die Anzahl der Dokumente, die den Index t_j enthalten (n_j) durch die Anzahl aller Dokumente N dividiert. In diesem Zusammenhang wird angenommen, dass die Verteilung der Indizes in den nicht relevanten Dokumenten durch die Verteilung der Indizes in allen Dokumenten der Sammlung approximiert werden kann. [Baeza-Yates; Ribeiro-Neto 1999, S. 32-33]

$$P(t_j \mid R) = 0.5 \qquad P(t_j \mid \overline{R}) = \frac{n_j}{N}$$

Der größte Vorteil des Modells ist, dass die Dokumente in absteigender Wahrscheinlichkeit der Relevanz angezeigt werden. Ein Nachteil ist die Notwendigkeit des Schätzens bezüglich der Relevanz für die Dokumente. Ein weiterer Nachteil besteht darin, dass dieses Modell nicht die Häufigkeit eines Index innerhalb eines Dokuments berücksichtigt. [Baeza-Yates; Ribeiro-Neto 1999, S. 33]

3.1.1.2 Information Retrieval-Bewertung

Zur Bewertung bzw. Verbesserung der Ergebnisse können verschiedene Maße und Verfahren angewendet werden.

Precision und Recall

Zur Bewertung der Ergebnisse von *Information Retrieval*-Systemen können die beiden am häufigsten verwendeten Evaluierungsmaße *Precision* und *Recall*, die Werte zwischen 0 und 1 annehmen können, verwendet werden. Wie Abbildung 3.4 zu entnehmen ist, enthält die Menge aller Dokumente die Menge der gefundenen und die der relevanten Dokumente. *Precision* ist eine relative Kennzahl, die die Genauigkeit der Antwort, und *Recall* eine, die die Vollständigkeit in Bezug auf die Anfrage, beschreibt. Der optimale Wert der beiden Maße ist 1, d. h. alle relevanten Dokumente sind im Ergebnis der Anfrage enthalten. Bei einer Spezifizierung der Anfrage und der damit verbundenen Verkleinerung der Ergebnismenge verbessert sich der *Precision* während der *Recall* sich verschlechtert. Im Gegensatz dazu vergrößert sich der *Recall* und verkleinert sich die *Precision* bei einer Verallgemeinerung der Anfrage und der damit zusammenhängenden Vergrößerung der Ergebnismenge. Bei einem Vergleich zweier *Information Retrieval*-Systeme müssen beide Maße eines der Systeme besser sein, um dieses als besser bezeichnen zu können. Falls nur ein Maß besser ist, können die

verschiedenen Systeme für unterschiedliche Aufgaben geeignet sein. Ein Wert nahe 1 im *Precision* ist wichtig, wenn die Ergebnisse der Extraktion nicht manuell kontrolliert werden. In anderen Fällen, bei denen die Extraktion nur eine Filterfunktion der Information ausführt, ist ein hoher *Recall* nahe 1 wichtig. Bei einem hohen *Precision*-Wert enthalten die extrahierten Informationen keine oder nur wenig Fehler. Ein hoher *Recall*-Wert bedeutet, dass nahe zu alle Informationen, die extrahiert werden sollten, auch extrahiert wurden. [Ferber 2003, S. 86-87] [Moens 2006, S. 181-182]

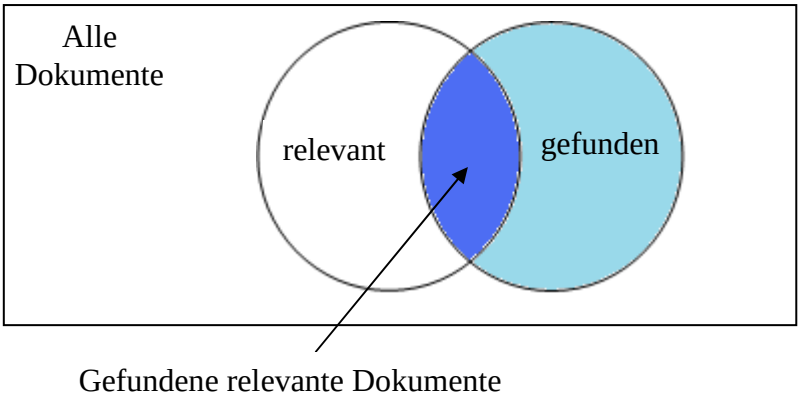


Abbildung 3.4: *Precision* und *Recall*

$$\text{Precision} = \frac{\text{gefundene relevante Dokumente}}{\text{gefundene Dokumente}}$$

$$\text{Recall} = \frac{\text{gefundene relevante Dokumente}}{\text{relevante Dokumente}}$$

Relevance Feedback

Beim *Relevance Feedback* verfeinert der Nutzer, wie in Abbildung 3.5 zu sehen ist, die Anfrage jedes Mal auf Grundlage der Ergebnisse der letzten Anfrage. In den meisten Fällen gibt der Nutzer jeweils an, welche Dokumente relevant sind. Darauf basierend werden der Anfrage neue Indizes hinzugefügt.

Bei dem *Vektorraummodell* sieht das *Relevance Feedback* folgendermaßen aus:

$$Q' = \alpha\, Q + \beta \sum_{i=1}^{n_1} \frac{r_i}{n_1} - \gamma \sum_{i=1}^{n_2} \frac{s_i}{n_2}$$

Dabei gehört r_i zur Menge R der n_1 relevanten und s_i zur Menge S der n_2 nicht relevanten Dokumentenvektoren. Q repräsentiert den ursprünglichen Vektor der Anfrage, Q' den der nach *Relevance Feedback* erzeugten Anfrage. Die Vektoren der Menge R werden zu dem Anfragevektor addiert und die der Menge S davon

subtrahiert. Das Ergebnis dieser Berechnung ist der neue Vektor der verfeinerten Anfrage. Zusätzlich können die einzelnen Funktionselemente noch mit Faktoren α, β und γ, die die Werte zwischen 0 und 1 annehmen können, gewichtet werden. Um die neue Anfrage ohne Einfluss der nicht relevanten Dokumente zu generieren, wird γ oftmals auf Null gesetzt. [Grossmann, Frieder 2004, S. 94-99] [Ferber 2003, S. 72]

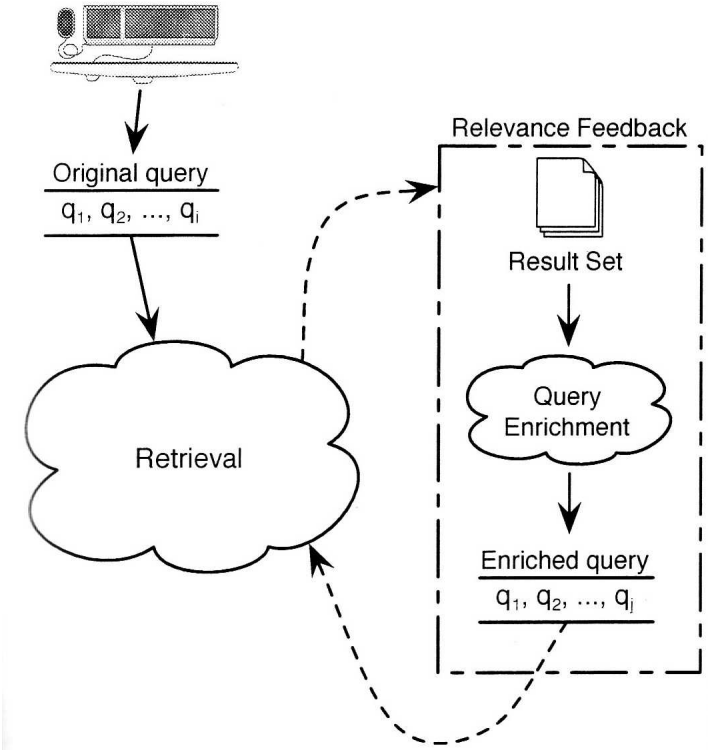


Abbildung 3.5: Prozess des *Relevance Feedback* [Grossmann, Frieder 2004, S. 95]

3.1.2 Data Mining

Unter *Data Mining*, umgangssprachlich auch Datenabbau oder Datengewinnung genannt, versteht man die Gewinnung relevanter und brauchbarer Daten aus einer großen Menge. Diese Daten werden einem mehrstufigen Prozess der Verarbeitung unterzogen, damit sie am Ende einem bestimmten Nutzenanspruch genügen und das gesuchte Muster und die Strukturmodelle zu erkennen sind. Strukturmodelle sind dabei schematische Aufbereitungen und Typisierungen der Daten. [Petersohn 2005, S. 8-10] *Data Mining* ist eine umfangreiche Extraktion von potentiell geeigneten Daten, die meist vorher nicht bekannt sind. Wesentliche Bestandteile dieser Methode sind die Erforschung und die Analyse großer Datenmengen durch automatische oder halbautomatische Verfahren, um aussagekräftige Muster zu finden. Die Aufgabe des *Data Mining* ist es, Ideen aus der künstlichen Intelligenz (engl: „Artificial Intelligence“ – AI), dem maschinellen Lernen, der Mustererkennung, der Statistik und der Datenbanksysteme zu extrahieren und zusammenzufassen. Durch die große Anzahl

an Daten, deren Dimensionalität und Heterogenität sind traditionelle Techniken für diese ungeeignet. Daher bedient sich *Data Mining*, wie in Abbildung 3.6 zu sehen ist, mehrerer Verfahren anderer Disziplinen, ist jedoch auch gleichzeitig eine eigene Disziplin. Aus der Statistik werden zum Beispiel die Cluster- und Zeitreihenanalyse verwendet. Aus dem Gebiet der künstlichen Intelligenz sind die künstlichen neuronalen Netze nützliche Verfahren. Die Datenbanktechnologie, die im *Data Mining* verwendet wird, ist zum großen Teil *OLAP* (On-Line-Analytical-Processing). [Tan, Steinbach, Kumar 2006, S. 2-6]

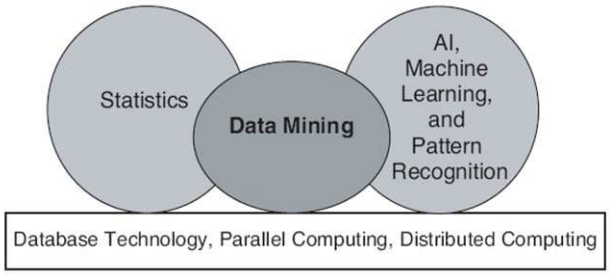


Abbildung 3.6: *Data Mining* als Zusammenschluss verschiedener Disziplinen
[Tan, Steinbach, Kumar 2006, S. 6]

3.1.2.1 Data Mining-Prozess

Das *Data Mining* kann, wie in Abbildung 3.7 zu sehen ist, in sieben verschiedene Phasen aufgeteilt werden. In der Aufgabendefinition wird die Aufgabe, mit dem Wissen über schon existierende und relevante Daten, aus der Problemstellung durch den Anwender erstellt. Diese Aufgaben können Klassifikations-, Assoziations- und Zeitreihenanalyseaufgaben beinhalten. Bei der Datenselektion steht die zielgerichtete Auswahl von Daten im *Data Mining*-Prozess im Vordergrund. Dabei werden die Daten in einem Data Warehouse oder in verteilten Datenbanken gespeichert. Hieraus werden für die Analyse nur die Daten ausgewählt, die für sinnvoll bzw. geeignet gehalten werden. Für die Entscheidung, ob Daten geeignet sind, werden Kriterien wie Qualität, Struktur und Inhalt dieser herangezogen. Aufgabe und Ziel der Datenaufbereitung ist die Erkennung und anschließende Eliminierung falscher oder fehlender Eintragungen. Die daraus resultierenden vorverarbeiteten Daten werden transformiert. Oft werden anschließend durch Reduktion und Projektion die Anzahl der Variablen verringert. Innerhalb der beiden Phasen Datenanalyse und Modellevaluation werden die Verfahren der Klassifikation, Assoziationsanalyse oder auch Zeitreihenanalyse verwendet. Die Ergebnisse dieser Verfahren bedürfen aufgrund unterschiedlicher Resultate einer Evaluation. Der *Data Mining*-Prozess leitet aus Daten entscheidungsrelevante Zusammenhänge her. Der Prozess wird nicht sequentiell durchgeführt, denn es können auch Rücksprünge vorkommen. [Petersohn 2005, S. 11-12]

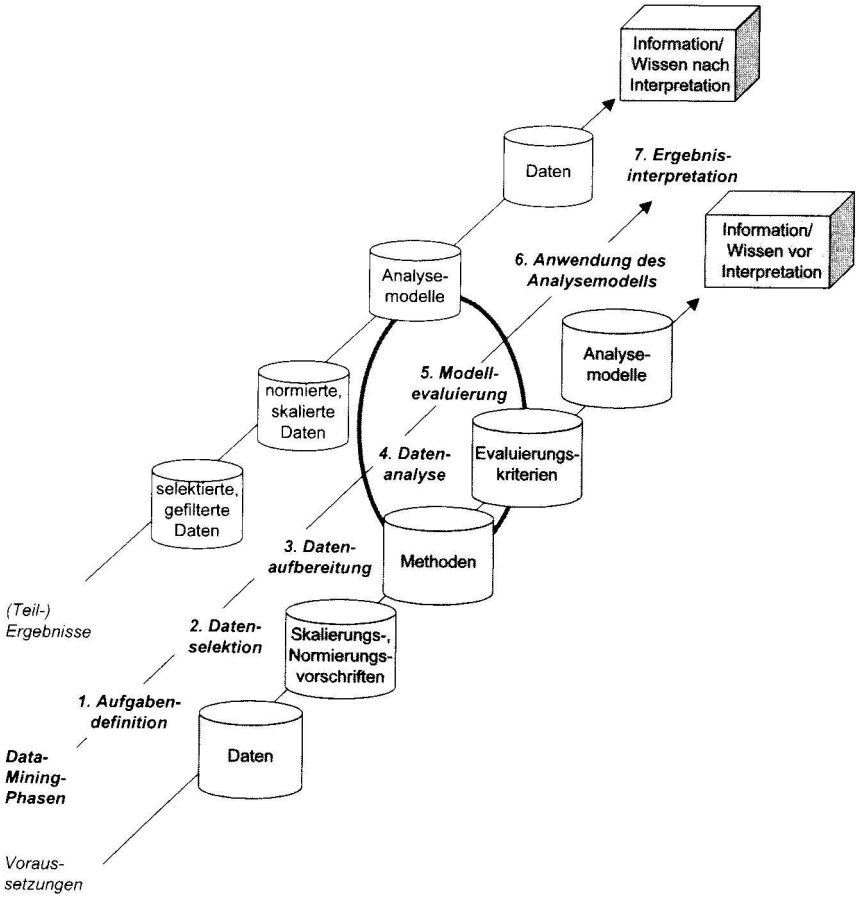


Abbildung 3.7: *Data Mining*-Prozess [Petersohn 2005, S. 13]

3.1.2.2 Data Mining-Methoden

Data Mining-Methoden können in deskriptive Methoden und Vorhersagemethoden unterteilt werden. Bei den deskriptiven Methoden werden durch den Menschen interpretierbare Muster, die die Daten beschreiben, gesucht. Vorhersagemethoden nutzen Variablen, um unbekannte oder zukünftige Werte anderer Variablen vorherzusagen. Die Assoziationsanalyse und das Clustering sind deskriptive Methoden und die Klassifikation ist eine Vorhersagemethode. In den folgenden Abschnitten werden diese näher beschrieben [Tan, Steinbach, Kumar 2006, S. 7-10]

3.1.2.2.1 Assoziationsanalyse

Die Assoziationsanalyse kann bei Unternehmen eingesetzt werden, die viele Daten über das eigene Unternehmen und über ihre Kunden bei den täglichen Abläufen sammeln. Ein Beispiel für solche Daten ist der Tabelle 3.4 zu entnehmen. Sie wird oft für die Erforschung interessanter, relevanter und wichtiger Korrelationen, die in großen Datenmengen auftreten, verwendet.

Tabelle 3.4: Beispiel von Verkaufsdaten

ID	Items
1	{Computer, Monitor}
2	{Computer, Maus, Scanner, Monitor}
3	{Scanner, Drucker, Laptop}
4	{Maus, Scanner, TV}

Die gefundenen Beziehungen können als Assoziationsregeln oder als *Frequent Itemset* dargestellt werden. Aus Tabelle 3.4 kann beispielsweise folgende Regel abgeleitet werden:

{Computer} → {Monitor}

Diese Regel besagt, dass eine enge Beziehung zwischen dem Verkauf von Computern und Monitoren besteht. Das wiederum bedeutet, dass viele Kunden neben einem Computer auch einen Monitor kaufen. Die Assoziationsanalyse kann ebenfalls in den Bereichen der Bioinformatik, medizinischen Diagnose, wissenschaftlichen Datenanalyse und des Web Mining angewendet werden. [Tan, Steinbach, Kumar 2006, S. 327-328]

Itemset und Support Count

Ein Item kann als binäre Variable dargestellt werden. Der Wert der Variable ist 1, wenn das Item in einer Transaktion vorhanden ist. Eine wichtige Eigenschaft von einem Itemset ist der *Support Count*, der sich auf die Anzahl von Transaktionen, die dieses spezifische Itemset enthält, bezieht. I ist die Menge aller möglichen Items in der Datenmenge. T entspricht der Menge aller auftretenden Transaktionen, wobei jede Transaktion t_j eine Teilmenge aus I enthält. Wenn ein Itemset aus k Items besteht, wird es k-Itemset genannt. [Tan, Steinbach, Kumar 2006, S. 329] Mathematisch lässt sich der *Support Count* für ein Itemset X folgendermaßen beschreiben:

$$\text{Support Count } \delta(\text{X}) = |\{t_j \mid X \subseteq t_j, t_j \in T\}|$$

Assoziationsregel

Eine Assoziationsregel ist ein Implikationsausdruck der Form $X \rightarrow Y$, wobei X und Y disjunkte Itemsets sind – $X \cap Y = \varnothing$. Die Stärke einer Assoziationsregel kann durch *Support* und *Confidence* der Regel gemessen werden. *Support* enthält die Information, wie oft eine Regel für gegebene Datenmengen anwendbar ist. *Confidence* dagegen gibt

an, wie oft Items der Menge Y in Transaktionen auftreten, die wiederum X beinhalten. [Tan, Steinbach, Kumar 2006, S. 329-330]

$$\text{Support}(X \rightarrow Y) = \frac{\delta(X \cup Y)}{N}$$

$$\text{Confidence}(X \rightarrow Y) = \frac{\delta(X \cup Y)}{\delta(X)}$$

Frequent Itemset

Frequent Itemsets sind die Itemsets, die den vorher festgelegten minimalen *Support Count* erfüllen. Durch eine netzartige Struktur, wie in Abbildung 3.8 zu sehen ist, können alle möglichen Itemsets dargestellt werden. Ein Datensatz, der k Items beinhaltet, hat dadurch potentiell bis zu 2^k-1 *Frequent Itemsets*, wodurch der Suchraum bei großen Datenmengen schnell exponentiell groß wird. Um die potentiellen *Frequent Itemsets* zu finden, wird für jedes Item der *Support Count* berechnet. Die Berechnung kann jedoch sehr aufwendig werden, da sehr viele Vergleiche benötigt werden.

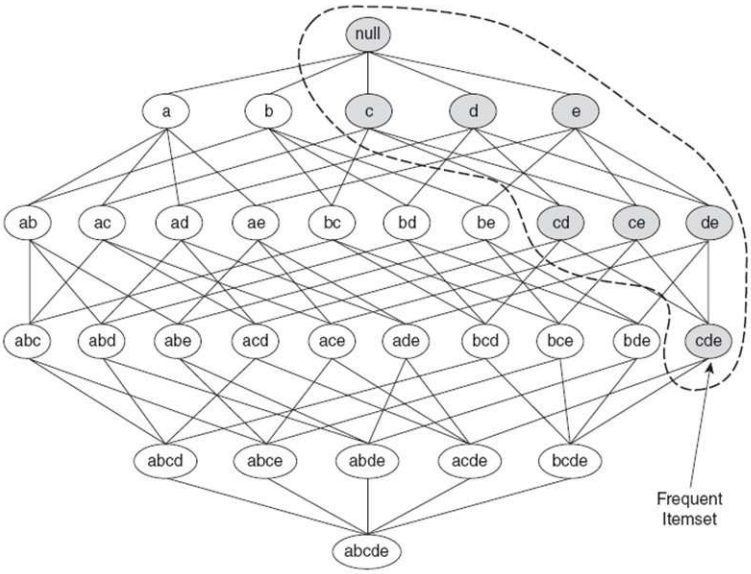


Abbildung 3.8: *Frequent Itemsets* nach dem *Apriori* Algorithmus [Tan, Steinbach, Kumar 2006, S. 334]

Eine Möglichkeit, die potentiellen *Frequent Itemsets* zu reduzieren, ist die Anwendung des *Apriori* Algorithmus. Dieses Verfahren nimmt an, dass aus einem *Frequent Itemset* folgt, dass alle dazugehörigen Subsets ebenfalls *frequent* sind. Analoges wird für nicht *frequente* Itemsets vorausgesetzt. Mit Hilfe des Algorithmus werden zuerst alle 1-Itemsets betrachtet und der *Support Count* berechnet. Ein vorher festgelegtes Minimum darf der *Support Count* nicht unterschreiten. Itemsets, bei denen dieser Wert

unterschritten wird, werden als nicht *frequent* eingestuft. Nach dem ersten Durchlauf werden alle durch den *Apriori* Algorithmus noch potentiell *frequenten* 2-Itemsets betrachtet und ebenfalls der *Support Count* dieser ermittelt. Abhängig von der Komplexität der Itemsets wird dieser Schritt so lange durchgeführt, bis es keine weitere Ebene mehr gibt. Die Effektivität des *Apriori* Algorithmus kann in der Anzahl generierter potentieller Itemsets gemessen werden. [Tan, Steinbach, Kumar 2006, S. 332-336]

3.1.2.2.2 Clustering

Die Basis beim Clustering bildet eine Menge von Datenpunkten. Jeder Punkt enthält eine Menge von Attributen, um sie nach ihrer Ähnlichkeit zueinander zu unterscheiden. Dabei können auf der einen Seite Cluster gefunden werden, deren Datenpunkte in verschiedenen Clustern weniger ähnlich sind zu anderen und auf der anderen Seite können Cluster gefunden werden, deren Datenpunkte in einem Cluster ähnlich zu anderen sind. Als eine Definition der Ähnlichkeit solcher Punkte kann die euklidische Distanz herangezogen werden. Bei der Clusteranalyse steht das Finden von Objektmengen im Vordergrund, innerhalb derer Menge die Objekte ähnlich sind oder in Beziehung zueinander stehen und sich gleichzeitig von Objekten anderer Mengen unterscheiden. Die Distanz zwischen Objekten innerhalb eines Clusters ist dabei in der Regel minimal, die zu Objekten anderer Cluster meist sehr groß. [Tan, Steinbach, Kumar 2006, S. 490-491]

Der Begriff eines Clusters kann, wie in Abbildung 3.9 zu sehen ist, verschieden interpretiert werden. Zu der gleichen Anzahl und Anordnung von Objekten können verschiedene Cluster gebildet werden.

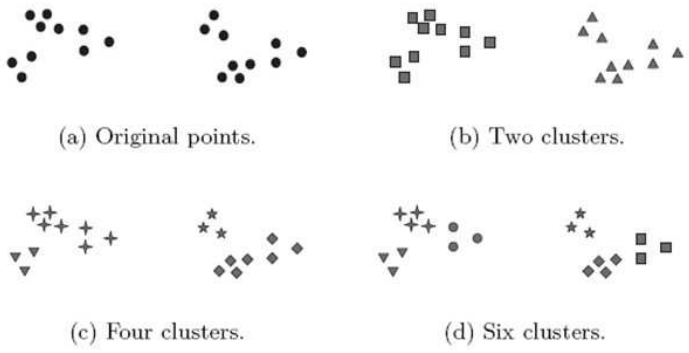


Abbildung 3.9: Verschiedene Arten der Clusterbildung [Tan, Steinbach, Kumar 2006, S. 491]

Es existieren unterschiedliche Arten des Clustering – jedes Clustering besteht aus einer Menge von Clustern. Auf der einen Seite gibt es das hierarchische Clustering und auf der anderen Seite das partitionierende Clustering. Beim partitionierenden Clustering

wird, wie in Abbildung 3.10 zu sehen ist, jedes Objekt in nicht überlappende Teilmengen, die sogenannten Cluster, eingeteilt, so dass jedes Objekt in genau einem Cluster ist. Im Gegensatz dazu wird die Gesamtheit aller Objekte beim hierarchischen Clustering verschachtelten Clustern zugeordnet, deren Struktur als hierarchischer Baum darstellbar ist. [Tan, Steinbach, Kumar 2006, S. 491-493] Weitere oft genutzte Clustering-Arten sind das exklusive Clustering, das Fuzzy-Clustering und das komplette Clustering. Beim exklusiven Clustering gehört jedes Objekt zu genau einem Cluster, während beim nicht exklusiven bzw. überlappenden Clustering ein Objekt gleichzeitig zu mehreren Clustern gehören kann. Im Gegensatz dazu gehört beim Fuzzy-Clustering jedes Objekt zu jedem Cluster, wobei jedes Objekt ein Zugehörigkeitsgewicht hat, das zwischen 0 und 1 liegt. Die Summe aller Gewichte ist 1. Beim kompletten Clustering wird jedes Objekt einem Cluster zugeordnet, im Gegensatz zu dem partiellen Clustering, bei dem nicht alle Objekte einem Cluster zugeordnet werden müssen. [Tan, Steinbach, Kumar 2006, S. 491-493]

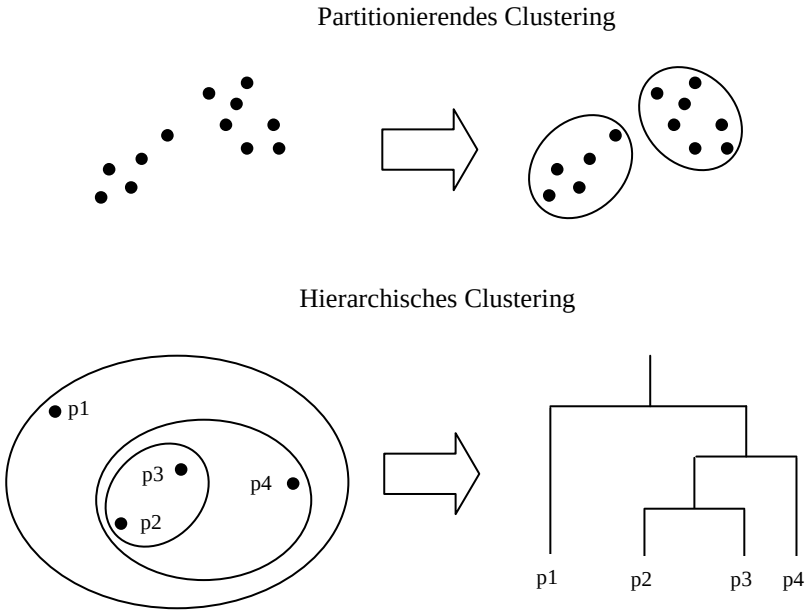
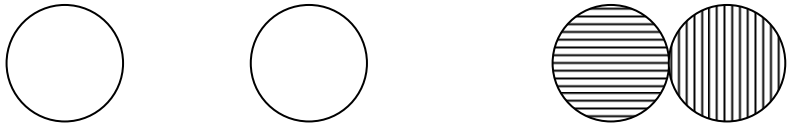


Abbildung 3.10: Partitionierendes und hierarchisches Clustering

Zusätzlich zu den verschiedenen Arten des Clustering gibt es auch verschiedene Clusterarten. In Abbildung 3.11 sind zwei Beispiele graphisch dargestellt. Gut verteilte Cluster sind Cluster, in denen jeder Punkt näher zu allen anderen Punkten in seinem Cluster ist als zu jedem anderen Punkt anderer Cluster. Bei den mittelpunktbasierten Clustern liegt jeder Punkt näher am Mittelpunkt seines eigenen Clusters als zu jedem Punkt der anderen Cluster. Weitere Clustertypen sind die nachbarschaftsbasierten Cluster, die dichtebasierten Cluster und die konzeptionellen Cluster bzw. Cluster mit gemeinsamen Eigenschaften. [Tan, Steinbach, Kumar 2006, S. 493-496]

Zur Durchführung des Clustering können verschiedene Algorithmen verwendet werden. Ein sehr oft benutzter Algorithmus neben dem hierarchischen und dem dichtebasierten Clustering ist der *k-means* Algorithmus.



a) b)

Abbildung 3.11: Clustertypen: a) gut verteilte Cluster b) mittelpunktbasierte Cluster

Zum Anfang dieses Algorithmus werden *k* Anfangsschwerpunkte festgelegt, wobei *k* als nutzerspezifischer Parameter die Anzahl gewünschter Cluster ist. Jeder Punkt wird dem nächsten Schwerpunkt zugeordnet. Der Schwerpunkt jedes Clusters wird neu berechnet basierend auf den Punkten, die dem Cluster zugeordnet sind. Dies wird solange wiederholt, bis sich die Schwerpunkte nicht mehr verändern. In Abbildung 3.12 wird der *k-means* Algorithmus kurz formal dargestellt. [Tan, Steinbach, Kumar 2006, S. 497]

```
Select k points as initial centroids
Repeat
      Form k clusters by assigning each point to its closed centroid.
      Recompute the centroid of each cluster.
Until Centroids  do not change.
```

Abbildung 3.12: *k-means* Algorithmus [Tan, Steinbach, Kumar 2006, S. 497]

3.1.2.2.3 Klassifikation

Die Hauptaufgabe der Klassifikation ist es, Objekte vorher definierten Kategorien zuzuordnen. In der Praxis wird dies sehr häufig beim Erkennen von Spam-Mails basierend auf der Nachricht, der Kopfzeile und dem Inhalt der E-Mail verwendet. Ein allgemeines Klassifikationsmodell ist in Abbildung 3.13 dargestellt.

Klassifikationstechniken erzeugen Klassifikationsmodelle von den Input-Daten. Techniken sind dabei *Entscheidungsbäume, regelbasierte Klassifikation*, *neuronale Netze, Support Vector Machine* und die *naïve Bayes Klassifikation*. Jede der Techniken benutzt einen Lernalgorithmus, um die Modelle zu identifizieren.

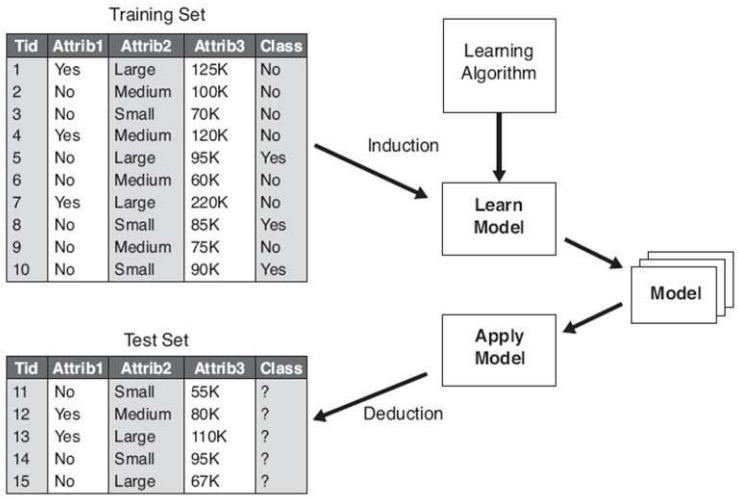


Abbildung 3.13: Allgemeines Klassifikationsmodell [Tan, Steinbach, Kumar 2006, S. 148]

Die Bewertung der Klassifikationsmodelle wird auf Basis der richtigen und falschen Vorhersagen durchgeführt. Anschaulich lässt sich die dazugehörige Analyse, wie in Tabelle 3.5 zu sehen ist, in einer Verschmelzungsmatrix darstellen. In dieser Tabelle sind $f_{11} + f_{00}$ die korrekten und $f_{01} + f_{10}$ die falschen Vorhersagen, da bei diesen prognostizierte und wirkliche Klasse übereinstimmen bzw. nicht übereinstimmen.

Tabelle 3.5: Verschmelzungsmatrix für ein 2-Klassen Problem

		Vorhergesagte Klasse	
		Klasse = 1	Klasse = 0
Aktuelle Klasse	Klasse = 1	f_{11}	f_{10}
	Klasse = 0	f_{01}	f_{00}

Auf dieser Grundlage kann die Bewertung durch die beiden Parameter *Accuracy* und *Error Rate* erfolgen. *Accuracy* entspricht dem Anteil der richtigen zu allen Vorhersagen. Im Gegensatz dazu gibt die *Error Rate* an, wie viele Vorhersagen im Verhältnis zu allen getroffenen falsch sind. Dementsprechend ist das Ziel immer, eine hohe *Accuracy* und eine niedrige *Error Rate* zu erreichen. [Tan, Steinbach, Kumar 2006, S. 149]

$$\text{Accuracy} = \frac{\text{Anzahl der korrekten Vorhersagen}}{\text{Anzahl aller Vorhersagen}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error Rate} = \frac{\text{Anzahl der falschen Vorhersagen}}{\text{Anzahl aller Vorhersagen}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Eine Technik, um Objekte zu klassifizieren, ist der *Entscheidungsbaum*. Diese Bäume sind hierarchisch strukturiert und bestehen aus Knoten und gerichteten Kanten. Durch

gezielte Fragen über die Attribute eines Objekts kann sich der Baum verzweigen. Es gibt drei verschiedene Arten von Knoten. Der Wurzelknoten hat, wie in Abbildung 3.14 zu sehen ist, keine eingehende und null oder mehr ausgehende Kanten. Ein interner Knoten besitzt genau eine eingehende Kante und mindestens zwei ausgehende Kanten. Ein Blattknoten hat genau eine eingehende Kante und keine ausgehenden Kanten; jeder Blattknoten entspricht einer Klassenbezeichnung. [Tan, Steinbach, Kumar 2006, S. 150]

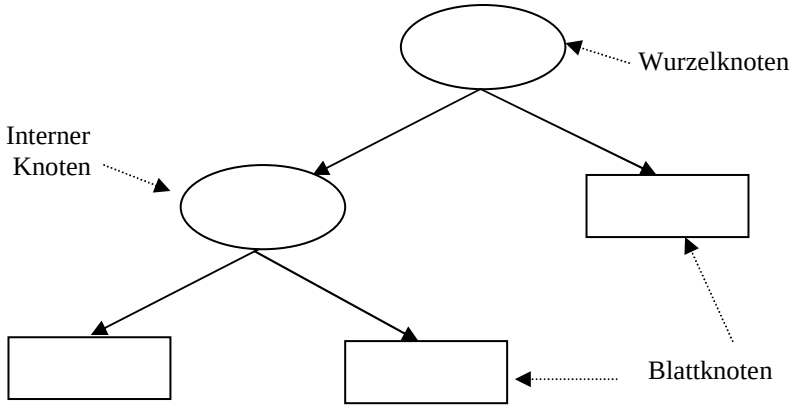


Abbildung 3.14: Modell eines Entscheidungsbaums

Der *Hunt´s* Algorithmus ist ein Algorithmus, mit dem ein *Entscheidungsbaum* auf rekursive Art und Weise aufgebaut wird. Bei dieser Methode ist D_t die Menge der Trainingsdatensätze, die mit den Knoten t verbunden sind, und $y = \{y_1, y_2, ..., y_c\}$ die Menge der Klassenbezeichnungen. Im ersten Schritt wird geprüft, ob alle Datensätze in D_t zu derselben Klasse y_t gehören. In diesem Fall wäre t ein Blattknoten und würde als y_t bezeichnet werden. Wenn D_t Datensätze enthält, die zu mehr als einer Klasse gehören, dann wird in Schritt 2 eine Attributtestbedingung ausgewählt, um die Datensätze in kleinere Unterklassen zu partitionieren. Für jedes Resultat der Bedingung wird ein sogenannter Kindknoten generiert. Der Algorithmus wird anschließend auf jeden Kindknoten angewendet. [Tan, Steinbach, Kumar 2006, S. 152-153]

Um *Entscheidungsbäume* zu erstellen, gibt es mehrere Möglichkeiten. Eine ist die Erstellung der Bedingungen mit binären Attributen, aus denen sich genau zwei Resultate ergeben. Die Bedingungen können ebenfalls mit nominalen, ordinalen oder fortlaufenden Attributen erstellt werden, bei denen jeweils auch mehr als zwei Resultate möglich sind. Bei den Resultaten nominaler Attribute können diese beliebig gruppiert werden. Resultate ordinaler Attribute, wie in Abbildung 3.15 zu sehen ist, stehen in einer bestimmten Reihenfolge zueinander. Bei den fortlaufenden Attributen wird ein Bereich in verschiedene Teilbereiche eingeteilt. Beispielsweise kann es bei Zahlen die Teilbereiche, also Resultate, „kleiner 30“, „zwischen 30 und 50“ und „größer 50“ geben. [Tan, Steinbach, Kumar 2006, S. 155-158]

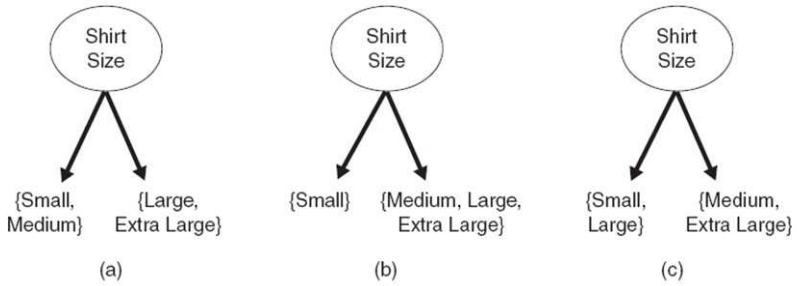


Abbildung 3.15: Verschiedene Arten ordinale Attribute zu gruppieren
[Tan, Steinbach, Kumar 2006, S. 157]

3.1.3 Pattern Matching

Das *Pattern Matching* besteht aus Pattern (dt: „Muster“), die eine Menge von syntaktischen Merkmalen beinhalten und in Texten vorkommen. Im Folgenden werden verschiedene Pattern-Typen und -Algorithmen vorgestellt. [Baeza-Yates, Ribeiro-Neto 1999, S. 104-105]

3.1.3.1 Pattern-Typen

Beim *Pattern Matching* können verschiedene Pattern-Typen verwendet werden, die von einfachen Wörtern bis zu regulären Ausdrücken reichen.

Wörter

Ein String, d. h. eine Folge von Buchstaben, ergibt in diesem Fall ein Wort. Dieses ist hierbei das Pattern. Wenn das Wort „Computer“ als Pattern vorgegeben ist, können alle Vorkommen von genau dem Wort gesucht und gefunden werden. [Baeza-Yates, Ribeiro-Neto 1999, S. 105]

Präfixe und Suffixe

Bei dem Typ *Präfixe* ist das Pattern der Anfang eines Worts. Wenn solch ein Pattern-Typ gegeben ist, wird, im Gegensatz zu dem Typ *Wörter*, nicht nur ein bestimmtes Wort, sondern es werden alle Wörter mit dem gegebenen Präfix gesucht. Mit dem Pattern „ab“ würden bei entsprechender Existenz z. B. die Wörter „abnehmen“ und „ablegen“ gefunden werden. Wenn das Pattern vom Typ *Suffixe* ist, dann handelt es sich um das Ende eines Worts. Mit dem Pattern „legen“ würden bei Vorkommen der entsprechenden Wörter beispielsweise „ablegen“ und „anlegen“ gefunden werden. [Baeza-Yates, Ribeiro-Neto 1999, S. 105]

Teilstrings

Teilstrings sind eine Erweiterung der *Präfixe* und *Suffixe*. Dabei können Patterns angegeben werden, die innerhalb von Wörtern vorkommen können. Dies kann innerhalb von Wörtern oder auch wortübergreifend passieren. Bei Auftreten der nachfolgenden Wörter werden beim Pattern „seh“ zum Beispiel die Wörter „Fernseher“, „sehen“ und „sehr“ gefunden werden. [Baeza-Yates, Ribeiro-Neto 1999, S. 105]

Bereiche

Wenn ein Pattern den Typ *Bereich* hat, dann wird eine Menge von Strings zwischen zwei vorgegebenen Strings in lexikographischer Reihenfolge als Pattern gesucht. Beispielsweise werden bei dem Bereich zwischen „Data“ und „Donner“ die Worte „Dieb“ und „Dolch“, sofern diese in den entsprechenden Dokumenten enthalten sind, gefunden. [Baeza-Yates, Ribeiro-Neto 1999, S. 105]

Fehler erlauben

Bei Pattern dieses Typs wird eine Fehlerschwelle bei den gesuchten Wörtern festgelegt. Das bedeutet, dass die Wörter einen bestimmten Fehlergrad haben dürfen. Bei dieser Methode werden die Anzahl der notwendigen Änderungen, um das richtige Wort zu generieren, als Fehlergrad bezeichnet. Grundlage dieses Ansatzes ist die Annahme, dass beim Schreiben Fehler entstehen können. Das Wort „Lam pe“ hat z. B. einen Fehlergrad von 1. Es würde eine Korrektur benötigen, um zum richtigen Wort „Lampe“ zu gelangen. [Baeza-Yates, Ribeiro-Neto 1999, S. 105]

Reguläre Ausdrücke und erweiterte Pattern

Bei den *regulären Ausdrücken* wird das Pattern durch einfache Strings, die mit Operatoren verbunden sind, gebildet. Operatoren sind in diesem Zusammenhang Vereinigung, Verknüpfung oder Wiederholung bzw. Kombinationen dieser Grundoperatoren. Bei der Vereinigung werden zwei verschiedene reguläre Ausdrücke generiert (z. B. a|b). Verknüpfung bedeutet, dass ein Ausdruck einem anderen folgt (z. B. ab). Wiederholung gibt an, wie oft ein *regulärer Ausdruck* vorkommen darf. Ein Beispiel eines komplexen *regulären Ausdrucks* ist „pro(blem|tein)(s|ε)(0|1|2)*“. ε ist in diesem Zusammenhang ein leerer String. Dabei entstehen die Wörter „problem02“ oder auch „proteins“. Diese Methode kann auch mit dem Typ *Fehler erlauben* kombiniert werden.

Erweiterte Pattern sind Untermengen der *regulären Ausdrücke* mit einer vereinfachten Syntax. Das bedeutet, dass sie intern durch einen Algorithmus in *reguläre Ausdrücke* konvertiert werden. Für diesen Typ existiert keine formale Definition, da sie von jedem System individuell gestaltet wird. [Baeza-Yates, Ribeiro-Neto 1999, S. 105-106]

3.1.3.2 Ansätze, Methoden und Algorithmen

Das grundlegende Anliegen der in diesem Kapitel vorgestellten Algorithmen ist möglichst alle Vorkommen des Patterns $p = p_1 p_2 ... p_m$ in einem großen Text $T = t_1 t_2 ... t_n$ zu finden. T und p sind dabei jeweils eine Folge von Buchstaben aus einer begrenzten Menge an Buchstaben $\sum$. Bei den gegebenen Strings a, b und c ist a ein Präfix von ab, ein Suffix von ba und ein Faktor von abc. [Navarro, Raffinot 2004, S. 15]

3.1.3.2.1 Präfix-basierter Ansatz

Bei diesem Ansatz wird der Text bis zu einer Position i gelesen, wobei die Länge des längsten Suffixes des Texts, das gleichzeitig ein Präfix des Patterns ist, bekannt ist. Zwei häufig verwendete Algorithmen, die in diesem Kapitel vorgestellt werden, sind der *Knuth-Morris-Pratt* Algorithmus und der *Shift-And* bzw. *Shift-Or* Algorithmus. [Navarro, Raffinot 2004, S. 17]

Knuth-Morris-Pratt

Bei dem *Knuth-Morris-Pratt* Algorithmus (*KMP*) ist das Ziel, das längste Präfix des Pattern zu finden, das zeitgleich auch ein Suffix des Texts ist. Der String $v\beta$ in Abbildung 3.16 kann das neue längste Präfix von p und gleichzeitig ein Suffix von $t_1 ... t_{i+1}$ sein. Zu sehen ist, dass v ein Suffix von u und ein Präfix von p ist. v wird Rand von u genannt. Außerdem muss der Buchstabe β mit t_{i+1} (in der Abbildung 3.16: σ) übereinstimmen. Zu Beginn wird der längste Rand $b(u)$ für jedes Präfix u des Patterns berechnet. Beim *KMP* wird angenommen, dass u das längste Präfix ist, das gleichzeitig ein Suffix von $t_1 ... t_i$ ist. Anschließend wird der Buchstabe $\sigma = t_{i+1}$ ausgelesen. Gilt $\sigma = p_{|u|+1}$, das in der Abbildung α ist, dann ist das neue längste Präfix $up_{|u|+1}$. Ist andererseits $\sigma \neq p_{|u|+1}$, so wird σ mit $p_{|b(u)|+1}$ verglichen. Ist dann $\sigma = p_{|b(u)|+1}$, wird $b(u)p_{|b(u)|+1}$ als längstes Präfix von p und gleichzeitig als Suffix von $t_1 ... t_{i+1}$ angenommen. Andernfalls ($\sigma \neq p_{|b(u)|+1}$) wird σ wiederum mit $p_{|b(b(u))|+1}$ verglichen. Dies wird fortgesetzt bis σ nach einem Rand kommt oder kein Rand mehr vorhanden ist. [Navarro, Raffinot 2004, S. 17-18]

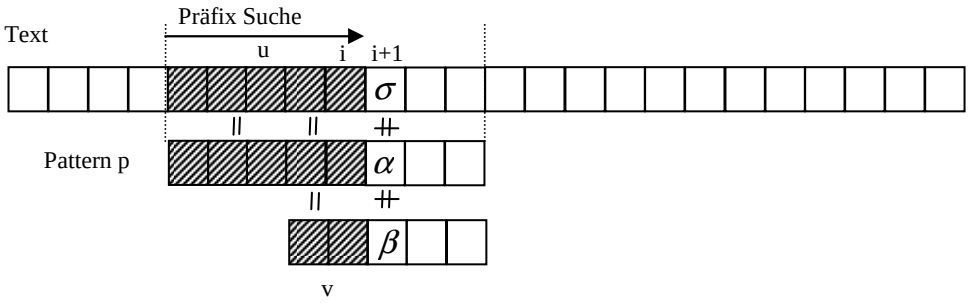


Abbildung 3.16: *KMP*: String v ist Suffix vom Präfix u und gleichzeitig ein Präfix

Der *KMP* hat sowohl im schlechtesten als auch im durchschnittlichen Fall in der Suchphase eine Laufzeit von $O(n)$. Es existieren zwei Ziele bei der Berechnung. Auf der einen Seite soll für jedes geeignete Präfix u des Pattern der längste Rand v und auf der anderen Seite für das Pattern selbst der eigene längste Rand gefunden werden. [Navarro, Raffinot 2004, S. 17-18]

Shift-And und Shift-Or

Der *Shift-And* und der *Shift-Or* Algorithmus sind einfacher als der *KMP*. Bei diesen Methoden wird die Menge der Präfixe von p, die mit den Suffixen der Texte übereinstimmen, gespeichert. Der Algorithmus benutzt Bit-Parallelität, um diese Menge für jeden neuen Buchstaben zu aktualisieren. $D = d_m ... d_1$ ist bei dem *Shift-And* eine Bitmaske für jeden neuen Buchstaben im Text. In Abbildung 3.17 wird der *Shift-And* Algorithmus kurz dargestellt. [Navarro, Raffinot 2004, S. 19-22]

Shift-And $(p = p_1 p_2 ... p_m, T = t_1 t_2 ... t_n)$
  Preprocessing
    For $c \in \sum$ **Do** $B[c] \leftarrow 0^m$
    For $j \in 1...m$ **Do** $B[p_j] \leftarrow B[p_j] \mid 0^{m-j}10^{j-1}$
  Searching
    $D \leftarrow 0^m$
    For $pos \in 1...n$ **Do**
      $D \leftarrow ((D << 1) \mid 0^{m-1}1) \,\&\, B[t_{pos}]$
      If $D \,\&\, 10^{m-1} \neq 0^m$ **Then** report an occurence at $pos - m + 1$
    End of for

Abbildung 3.17: *Shift-And* Algorithmus [Navarro, Raffinot 2004, S. 20]

Eine 1 wird in die j-te Position von D gespeichert, wenn $p_1 ... p_j$ ein Suffix von $t_1 ... t_i$ ist, und diese Position wird damit auf *aktiv* gestellt. Sobald d_m *aktiv* ist, wird ein Treffer angezeigt. Nach dem Lesen vom Buchstaben t_{i+1} muss die neue Menge D' berechnet werden. Eine Position $j+1$ wird nur dann *aktiv* gesetzt, wenn die Position j in der Menge D *aktiv* war. Das wiederum bedeutet, dass $p_1 ... p_j$ ein Suffix von $t_1 ... t_i$ ist und

t_{i+1} mit p_{j+1} übereinstimmt. Die neue Menge D' ist leicht in konstanter Zeit durch das Nutzen von Bit-Parallelitätsoperationen zu berechnen. Der Algorithmus füllt zuerst eine Tabelle B, die die Bitmaske $b_m...b_1$ für jeden Buchstaben speichert. Die Maske $B[c]$ beinhaltet die j-te Bitmenge, wenn $p_j = c$. Am Anfang wird $D = 0^m$ gesetzt und für jeden neuen Buchstaben t_{i+1} mit der folgenden Formel aktualisiert:

$$D' \leftarrow ((D << 1) \mid 0^{m-1}1) \,\&\, B[t+1]$$

Das „<<“ schiebt die Positionen nach links, um Schritt $i+1$ zu markieren, wobei die Positionen von p Suffixe in Schritt i waren. Außerdem wird der leere String ε als ein Suffix markiert. Von diesen Positionen werden nur die ausgewählt, bei denen t_{i+1} mit p_{j+1} übereinstimmt. Der *Shift-Or* Algorithmus ist eine komplizierte Implementierung des *Shift-And* Algorithmus. Dabei wird die Verwendung der $0^{m-1}1$-Maske in der oben erwähnten Formel vermieden, um die Geschwindigkeit der Berechnung zu erhöhen. Stattdessen werden alle Bitmasken von B ergänzt. Wenn die Schiebeoperation eine 0 rechts von D' einführt, kommt das neue Suffix von dem leeren String, der bereits in D' vorhanden ist. [Navarro, Raffinot 2004, S. 19-22]

3.1.3.2.2 Suffix-basierter Ansatz

Eine der größten Schwierigkeiten bei dem Suffix-basierten Ansatz ist es, das Suchfenster auf eine sichere Art und Weise zu schieben, ohne ein Auftreten des Musters zu überspringen. Eine Lösung für dieses Problem liefert der *Boyer-Moore* Algorithmus, der hier vorgestellt wird. [Navarro, Raffinot 2004, S. 22]

Boyer-Moore

Dieser Algorithmus berechnet die drei Schiebefunktionen d_1, d_2 und d_3, die den drei folgenden Situationen angepasst sind.

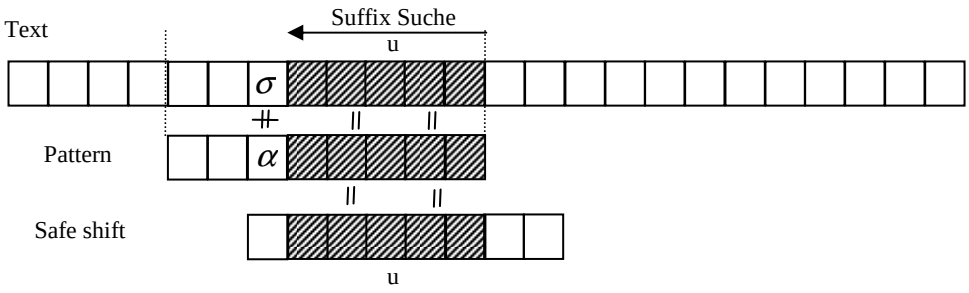


Abbildung 3.18: Erste Schiebung des *Boyer-Moore* Algorithmus

Fall 1: Das Suffix u kommt an einer anderen Position als Faktor von p vor. Sicheres Schieben bedeutet in dem Fall, das Fenster so zu bewegen, dass u im Text mit dem

nächsten Vorkommen von u im Pattern übereinstimmt, wie in Abbildung 3.18 zu sehen ist. Die Idee ist, für jedes Suffix des Pattern die Distanz zu der Position des nächsten Auftretens im Text zu berechnen. Diese Funktion wird mit d_1 bezeichnet. Wenn das Suffix u von p nicht noch einmal in p auftritt, dann ist u durch d_1 mit der Größe m des Pattern verbunden. [Navarro, Raffinot 2004, S. 23-24]

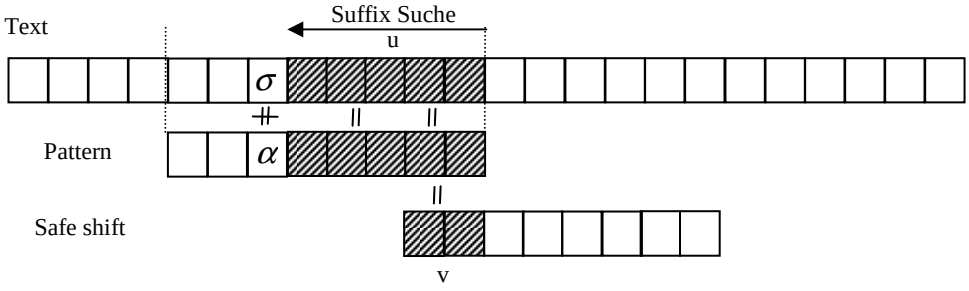


Abbildung 3.19: Zweite Schiebung des *Boyer-Moore* Algorithmus

Fall 2: Das Suffix u ist kein Faktor von p in anderen Positionen. Trotzdem sollte nicht das gesamte Suchfenster übersprungen werden. Wie in Abbildung 3.19 zu sehen ist, kann ein Suffix v von u ein Präfix vom Pattern sein. Daher wird eine zweite Funktion d_2 für alle Suffixe des Pattern berechnet. Für jedes Suffix u von p wird die Länge des längsten Präfixes v von p, das auch ein Suffix von u ist, ermittelt. [Navarro, Raffinot 2004, S. 23-24]

Fall 3: Ausgangspunkt ist die Annahme, dass die Rückwärtssuche auf den Textbuchstaben σ kein positives Ergebnis gebracht hat. Wenn das Fenster mit der ersten Funktion d_1 geschoben wird und der Buchstabe nicht mit einem σ im Pattern übereinstimmt, wird eine unnötige Verifikation von dem neuen Suchfenster ausgeführt, wie in Abbildung 3.20 zu sehen ist. Die dritte Funktion d_3 wird berechnet, um sicherzustellen, dass der Buchstabe σ des Texts mit σ aus dem Pattern für die nächste Verifikation übereinstimmt. Dafür wird jeder Buchstabe σ aus dem Alphabet $\sum$ mit der Distanz vom rechten Vorkommen bis zum Ende des Pattern verbunden. [Navarro, Raffinot 2004, S. 23-24]

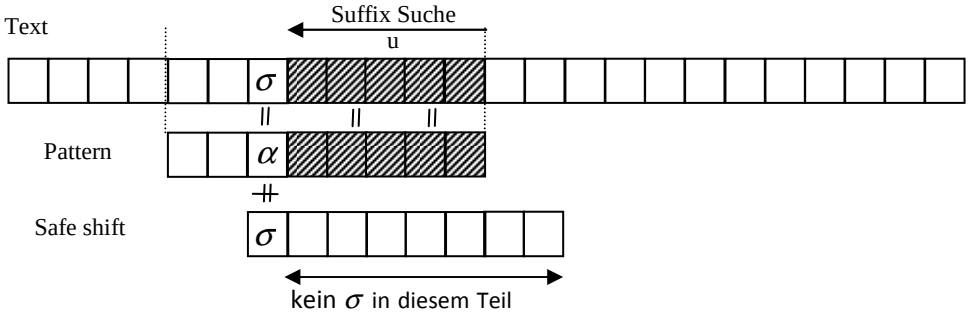


Abbildung 3.20: Dritte Schiebung des *Boyer-Moore* Algorithmus

Nachdem u eingelesen wurde und an der Stelle σ fehlgeschlagen ist, vergleicht der *Boyer-Moore* Algorithmus (*BM*) zwei Schiebungen. Auf der einen Seite wird das Maximum zwischen den Schiebungen $d_1(u)$ und $d_1(\sigma)$ berechnet und auf der anderen Seite wird das Minimum des zuvor berechneten Maximums und $m - d_2(u)$ ermittelt. Der Suchpart des *BM* hat im schlechtesten Fall eine Komplexität der Größenordnung $O(mn)$, im Durchschnittsfall jedoch eine sublineare Komplexität. [Navarro, Raffinot 2004, S. 24]

3.1.3.2.3 Faktor-basierter Ansatz

Bei diesem Ansatz wird rückwärts nach Faktoren des Patterns gesucht. Als häufig verwendete Methode zum Faktor-basierten Ansatz wird hier die *Backward Dawg Matching* Idee vorgestellt.

Backward Dawg Matching

Das *Backward Dawg Matching* benutzt Suffix-Automaten, die sehr mächtig aber auch komplex sind, um die Faktorsuche durchzuführen. Bei den Suffix-Automaten steht das Erkennen eines gegebenen Worts u als Faktor des Pattern p im Mittelpunkt. Es existieren mehrere indexierte Strukturen, die dies in $O(|u|)$ durchführen. Die klassische Struktur ist der kompakte Suffix-Baum. In diesem werden Übergänge als Faktoren des Patterns dargestellt. Suffix-Automaten haben die gleiche Effizienz wie der Suffix-Baum, jedoch werden ihre Übergänge mit einfachen Buchstaben bezeichnet. Dies beschleunigt die Suche und den *Pattern Matching*-Algorithmus. [Navarro, Raffinot 2004, S. 28]

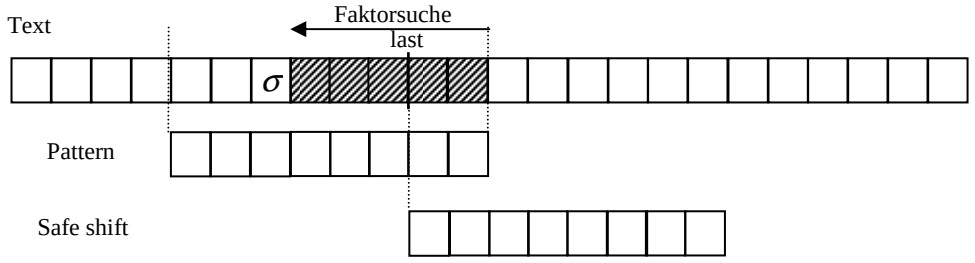


Abbildung 3.21: Backward Dawg Matching

Suffix-Automaten haben im Allgemeinen drei Basiseigenschaften. Die erste Eigenschaft ermöglicht dem Nutzer in $O(|u|)$ rauszufinden, ob ein String u ein Faktor des Strings p ist. Der String u ist ein Faktor des Strings p im Automaten, wenn ein Pfad mit u gekennzeichnet am Anfangsknoten beginnt. Wenn ein Pfad, beginnend am Anfangsknoten, einen abschließenden Status des Automaten erreicht hat, ist dies ein Suffix des Pattern. Das ist erreicht, wenn die Bezeichnung des Pfads ein Suffix von p ist. Diese

Möglichkeit des Erkennens der Suffixe ist die zweite Eigenschaft des Automaten. Bei der dritten Eigenschaft kann jeder Buchstabe p_j nacheinander der Struktur hinzugefügt werden. Innerhalb der Struktur wird bei jedem Schritt j der Suffix-Automat des Präfixes $p_1...p_{j-1}$ aktualisiert. [Navarro, Raffinot 2004, S. 28]

Der Suchalgorithmus benutzt die eben beschriebenen Eigenschaften des Suffix-Automaten. Es wird ein Suffix-Automat $p^{rv} = p_m p_{m-1}...p_1$ gebildet, um das Pattern $p = p_1 p_2...p_m$ in einem Text $T = t_1 t_2...t_n$ zu suchen. Der Algorithmus sucht unter Benutzung des Suffix-Automaten rückwärts entlang des Fensters nach einem Faktor des Patterns. Wenn ein abschließender Status erreicht wurde, der nicht zum gesamten Pattern passt, wird die Position im Fenster in der Variable *last* gespeichert. Die zweite Eigenschaft des Suffix-Automaten kann genutzt werden, um ein Präfix des Pattern, beginnend an der Position *last* im Fenster und endend am Ende des Fensters, zu finden. Möglich macht dies die Tatsache, dass die Suffixe von p^{rv} die umgedrehten Präfixe von p sind. Diese Rückwärtssuche kann in zwei Wegen enden. Im ersten Fall wird ein Buchstabe σ erreicht, der nicht zu einem Übergang im Suffix-Automaten von p^{rv} passt. Anschließend wird, wie in Abbildung 3.21 zu sehen ist, das Fenster so geschoben, dass die neue Startposition *last* ist. Beim zweiten Fall wird der Anfang des Fensters erreicht, das wiederum bedeutet, dass ein Muster gefunden wurde. Anschließend wird das Fenster wie im ersten Fall geschoben.

Der Algorithmus hat im schlechtesten Fall eine Komplexität der Größenordnung $O(mn)$. Im Durchschnittsfall ist die Komplexität $O((n \log_{|\Sigma|} m)/m)$ unter der Annahme, dass die Buchstaben im Text unabhängig voneinander sind und die gleiche Wahrscheinlichkeit des Auftretens haben. [Navarro, Raffinot 2004, S. 28-29]

3.1.3.2.4 Reguläre Ausdrücke

Reguläre Ausdrücke sind komplexer als einfache oder erweiterte Strings. Sie werden häufig im Zusammenhang mit *Text Retrieval* oder Computerbiologieanwendungen herangezogen.

„A regular expression RE is a string on the set of symbols $\Sigma \cup \{\varepsilon, |, \bullet, *, (,)\}$*, which is recursively defined as the empty character* ε*; a character* $\alpha \in \Sigma$*; and* (RE_1)*,* $(RE_1 \bullet RE_2)$*,* $(RE_1 | RE_2)$*, and* $(RE_1 *)$*, where* (RE_1) *and* (RE_2) *are regular expressions."* [Navarro, Raffinot 2004, S. 99]

In Abbildung 3.22 werden die klassischen Anwendungen zusammengefasst. Zuerst wird der reguläre Ausdruck in einen Baum gegliedert. Dieser Baum wird anschließend in einen *Nondeterminstic Finite Automaton* (*NFA*) transformiert. Es existieren mehrere

Möglichkeiten für diese Transformation. Zwei davon sind die *Thompson construction* und die *Glushkov construction*. Dieser *NFA* kann danach auch in einen *Deterministic Finite Automaton* (*DFA*) konvertiert werden. Dies hat den Vorteil, dass die Komplexität im besten Fall eine Komplexität der Größenordnung von $O(n)$ besitzt, im schlechtesten Fall ist diese jedoch $O(2^m)$. [Navarro, Raffinot 2004, S. 100-101]

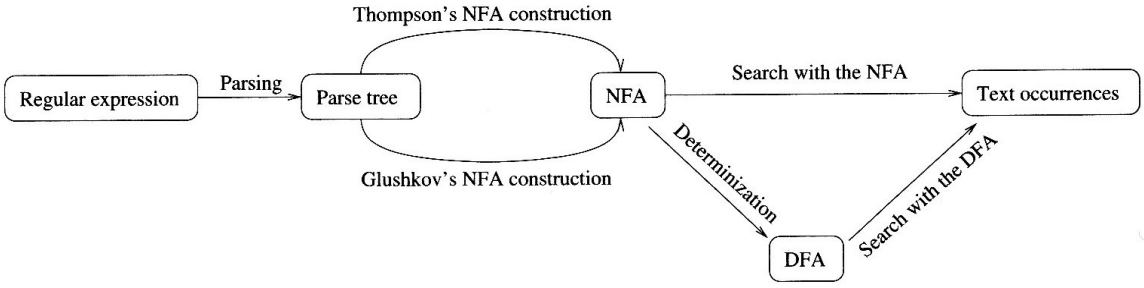


Abbildung 3.22: Klassische Anwendungen für die Suche nach regulären Ausdrücken
[Navarro, Raffinot 2004, S. 100]

Wie in Abbildung 3.23 zu sehen ist, kann ein regulärer Ausdruck durch einen Baum repräsentiert werden. Die Blätter des Baums werden durch die Buchstaben von $\sum$ gekennzeichnet, interne Knoten durch die Operatoren. Die Operatoren "|" und "•" haben genau zwei Kinder, die Unterausdrücke RE_1 und RE_2. Im Gegensatz dazu hat der Operator "*" nur ein Kind RE_1. [Navarro, Raffinot 2004, S. 101]

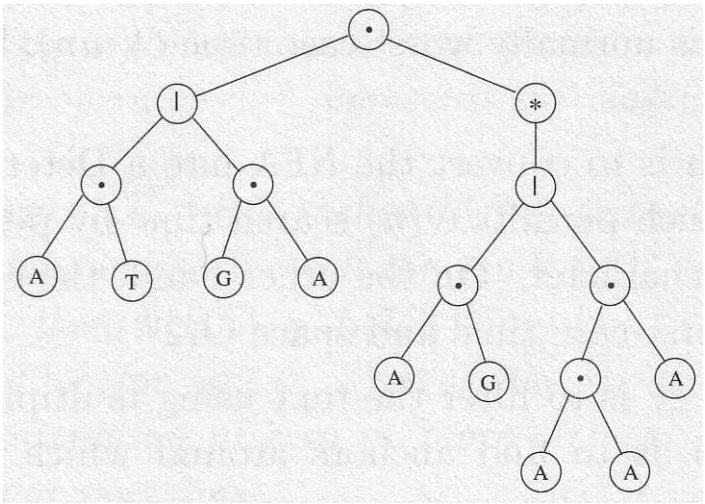


Abbildung 3.23: Baumrepräsentation des *regulären Ausdrucks* $(AT \mid GA)((AG \mid AAA)^*)$
[Navarro, Raffinot 2004, S. 102]

3.1.3.2.5 Erweiterte Pattern

Erweiterte Pattern sind komplexitätstechnisch zwischen einfachen Strings und regulären Ausdrücken angesiedelt. In diesem Kapitel werden vier Erweiterungen des Problems,

Strings zu suchen, beschrieben: *Klassen von Buchstaben, bounded length gaps, optionale Buchstaben* und *wiederholbare Buchstaben*. [Navarro, Raffinot 2004, S. 77]

Klassen von Buchstaben

Die einfachste Erweiterung, die *Klasse von Buchstaben*, ermöglicht, jeder Patternposition mit einer Menge an Buchstaben übereinzustimmen. Das Pattern ist eine Sequenz über die Potenzmenge $\wp(\Sigma)$, wobei $p = p_1 p_2 ... p_m$ und $p_j \subseteq \Sigma$. $p' \in \Sigma^*$ ein Vorkommen von p ist, wenn $p'_j \in p_j$ für alle $j \in 1...m$ ist. Es werden meistens eckige Klammern benutzt, um die Menge der möglichen Buchstaben darzustellen. Ein Beispiel dafür ist „[Ee]xample“, bei dem die Strings „Example“ und „example“ übereinstimmen würden, sofern diese existieren. [Navarro, Raffinot 2004, S. 78]

Bounded length gaps

Beim *Bounded length gaps* werden die Buchstaben oder Klassen des Patterns durch einen Bindestrich getrennt; x(a,b) kennzeichnet einen Bereich der Länge a bis b. Beispielsweise werden bei dem Pattern „a – d – e – f – x(1,3) – k – r“ die Strings „adefcdkr“ und „adefskr“ gefunden, sofern diese existieren. „x(1,3)“ bedeutet hierbei, dass mindestens 1 und maximal 3 Buchstaben zwischen den restlichen Buchstaben stehen. [Navarro, Raffinot 2004, S. 81]

Optionale Buchstaben

Bei den *optionalen Buchstaben* kann ein Buchstabe des Pattern im Text vorkommen oder nicht. Dies wird durch ein Fragezeichen nach dem optionalen Buchstaben gekennzeichnet. Beim Pattern „ert?zu?io“ werden z. B. die Strings „ertzuio“ und „erzio“ gefunden, wenn sie im Text enthalten sind. [Navarro, Raffinot 2004, S. 87-88]

Wiederholbare Buchstaben

Die *wiederholbaren Buchstaben* werden durch einen Stern nach dem Buchstaben gekennzeichnet. Der Buchstabe kann nicht oder beliebig oft vorkommen. Beispielsweise werden bei dem Pattern „WER*T“ die Strings „WERRRRRT“ und „WERRT“ im Text gefunden, wenn dieser die enthält. [Navarro, Raffinot 2004, S. 89-90]

3.1.4 Information Extraction

Information Extraction (*IE*) beschreibt in den meisten Fällen die Aufgabe des Aufspürens spezifischer Informationen von einem natürlich-sprachlichen Dokument. [Eikvil 1999, S. 5] Viele Firmen auf der gesamten Welt nutzen Datenbanken mit Millionen von Datensätzen, die sie effektiv halten wollen, um relevante bzw. benötigte Informationen aus der Datenbank in geringer Zeit abzufragen. Gemeinsamer Ausgangspunkt aller Situationen ist die Anfrage nach Informationen. Die Antwort auf diese Anfrage kommt meistens aus unstrukturierten Datenquellen wie z. B. Texten und Bildern. Es ist unmöglich, für einen Menschen diese großen Datenmengen zu überblicken. Computer sind meistens nicht in der Lage die benötigten Informationen direkt abzufragen, da sie in einem unstrukturierten Format vorliegen. *Information Extraction* ist eine Unterdisziplin der künstlichen Intelligenz, die versucht diese oder ähnliche Probleme zu lösen. Die folgende Definition fasst die Aufgaben und Tätigkeiten des *Information Extraction* zusammen. [Moens 2006, S. 1-3]

„Information extraction is the identification, and consequent or concurrent classification and structuring into semantic classes, of specific information found in unstructured data sources, such as natural language text, making the information more suitable for information processing tasks." [Moens 2006, S. 4]

IE wird in den meisten Fällen verwendet, um Informationen aus unstrukturierten Datenquellen zu gewinnen, wobei unstrukturiert nicht bedeutet, dass die Daten strukturell zusammenhangslos vorhanden sind. Die Informationen sind so verschlüsselt, dass es für einen Computer schwer ist, diese sofort zu interpretieren. Genauer sollte von unklaren und nicht transparenten Daten gesprochen werden. Der Prozess des *Information Extraction* fügt den unstrukturierten und unverarbeiteten Daten – egal ob Bild, Text, Video oder Audio – eine Bedeutung hinzu. Innerhalb dieser Arbeit stehen die Textdaten im Vordergrund. Nach dem *IE*-Prozess liegen strukturierte oder zumindest semi-strukturierte Daten vor. Auf diese Art aufbereitete Daten können besser vom Computer durch beispielsweise *Information Retrieval* oder *Data Mining*, verarbeitet werden. Texte können unterschiedliche Typen wie z. B. Artikel, Verträge oder Polizeiberichte besitzen. Schriftliche Daten, gerade in elektronischer Form, sind dafür bekannt, zusammenhangslos zu sein und viele Grammatik- und Rechtschreibfehler zu enthalten. [Moens 2006, S. 4-5]

3.1.4.1 Geschichte des Information Extraction

Zwei Faktoren waren in der gesamten Zeit der Entwicklung des *IE* von großer Bedeutung. Einer dieser beiden ist das exponentielle Wachstum der Menge an on- und

offline Textdaten. Der andere ist der Fokus auf das Gebiet durch die *Message Understanding Conferences* (*MUC*). Seit den späten 80ern förderte die US-Regierung *MUC*, um den aktuellen Stand des *Information Extraction* zu bewerten und zu verbessern. Die *MUC* bindet dabei mehrere Teilnehmer verschiedener Forschungseinrichtungen mit ein. Meistens war dies ein Mix aus akademischen und industriellen Forschungslaboren. Jeder Teilnehmer entwickelte ein *IE*-System für einen vorher festgelegten Bereich. Diese Systeme wurden anschließend auf dem gleichen Bereich und Textbestand getestet, bewertet und in ein offizielles Scoring-Programm eingetragen. Die Zielsetzung der Konferenzen war es, eine quantitative Bewertungsmöglichkeit von *IE*-Systemen zu entwickeln. Die Konferenzen haben den ersten groß-angelegten Versuch unternommen, natürlich-sprachliche Verarbeitungs-systeme zu bewerten. Durch die Konferenzen sollten Aufgaben wie latein-amerikanischer Terrorismus, Joint-Ventures und Mikroelektronik bewältigt werden. Anfang der 90er war die Forschung bei den *IE*-Systemen sehr erfolgreich und erzielte eine fast menschliche Performance in Englisch und Japanisch. [Eikvil 1999, S. 6]

3.1.4.2 Architektur und Aufgaben eines Information Extraction-Systems

Architektur

In Abbildung 3.24 wird ein typischer Aufbau eines *Information Extraction*-Systems dargestellt. Zu entnehmen ist der Darstellung, dass Systeme dieser Art zwei verschiedene Phasen haben: eine Trainingsphase (T) und eine Bereitstellungsphase (D)**.** In der Trainingsphase werden zunächst die Extraktionspattern erarbeitet. Im ersten Schritt wird ein Textkörper ausgewählt, der repräsentativ für die Aufgabe des Systems ist. Diese Texte durchlaufen eine Vorverarbeitungsphase (T2), nachdem Extraktions-regeln für die Texte extrapoliert wurden. In dieser Vorverarbeitungsphase werden die formalen Eigenschaften der Texte normalisiert. Zu dieser Phase gehört auch die Anreicherung der Textdaten mit sprachlichen Metadaten, die als Parameter im Beschaffungsprozess (T2.2) verwendet werden. Zum Ende des Prozesses wird noch eine Reihe von Tools zur Verarbeitung der natürlichen Sprache angewendet. Ein Informationsspezialist benutzt den Großteil des Vorverarbeitungsprozesses während der Lernphase (T3) als Basis für das Schreiben einer Extraktionsgrammatik. Im Falle des maschinellen Lernens wird der Trainingskörper zuerst manuell beschrieben, um abzubilden, welche Elemente in den Texten relevant sind für die Extraktionsaufgabe. Das Modul des maschinellen Lernens benutzt diese Beschreibung in der Lernphase, um automatisch die Grammatikregeln für den Großteil zu erzeugen. Die

Extraktionsgrammatik kann in Form von mathematischen Funktionen, die die Klasse von einem Beispiel vorhersagen, dargestellt werden. [Moens 2006, S. 36-38]

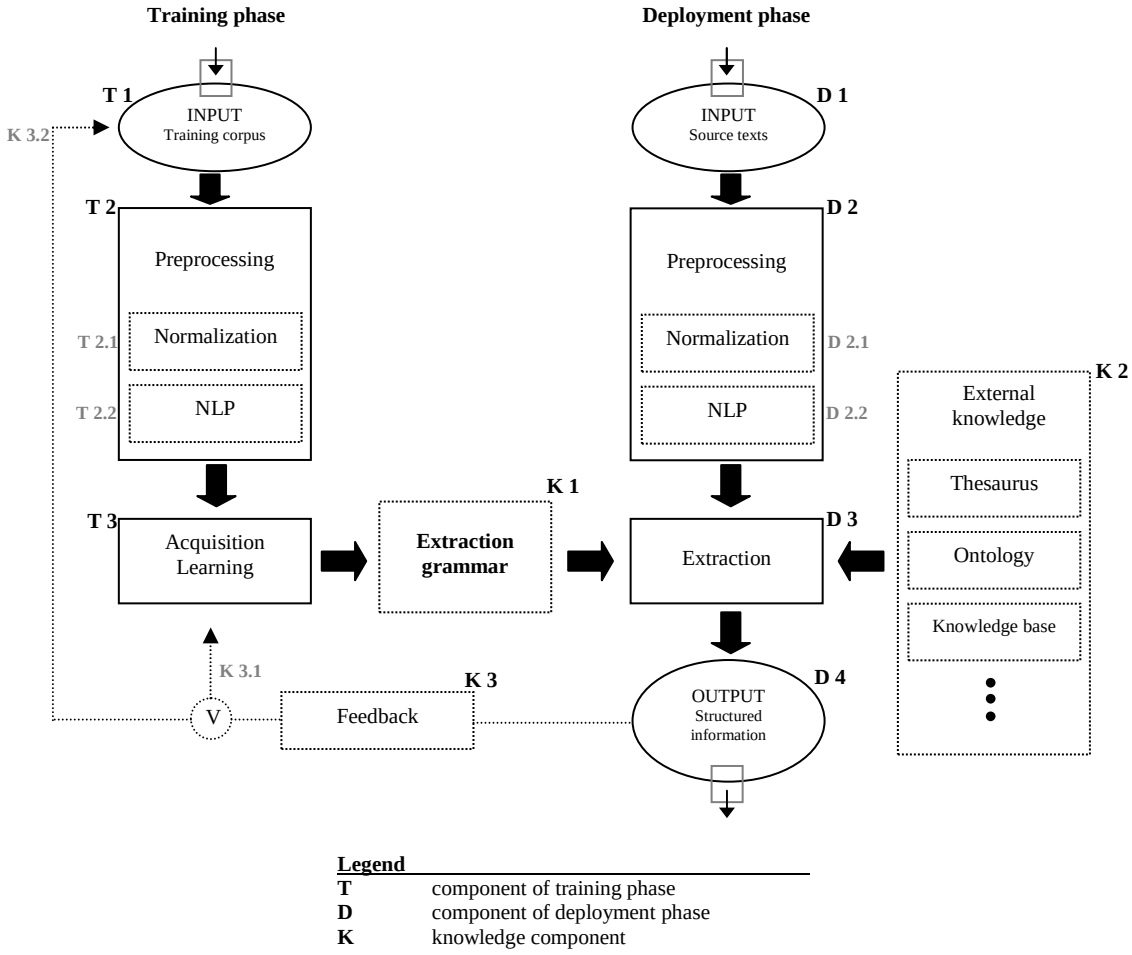


Abbildung 3.24: Ein typisches *Information Extraction*-System [Moens 2006, S. 37]

Es ist auch möglich, dass der Trainingskörper nicht manuell oder nur teilweise bezüglich entsprechender nicht überwachter und schwach beaufsichtigter Techniken beschrieben ist. In der Bereitstellungsphase (D1-4) werden relevante, semantische Informationen in neuen Texten identifiziert und klassifiziert. Die Vorverarbeitungskomponente in der Bereitstellungsphase (D2) verläuft sehr ähnlich der in der Lernphase. Nach der Vorverarbeitung werden die Inputtexte an die Extraktionsphase (D3) weitergegeben, die die Extraktionsgrammatik (K1) und geeignetes zusätzliches Wissen (K2) anwendet, um zu bestimmen, welche Elemente in den Inputtexten für die Extraktionsaufgabe relevant sind und wie sie mit bestimmten semantischen Klassen zusammenhängen. Es werden die textuellen Elemente extrahiert und anschließend klassifiziert, um sie in einem strukturierten Format (D4) wiederzugeben. Zusätzlich existieren Systeme, die ein Feedback zurückgeben und den Output des Systems korrigieren, um die Lernkomponente durch inkrementelles Lernen neu zu trainieren. Die Bereitstellungsphase wird oft Test- oder Evaluationsphase genannt. Es ist offensichtlich,

dass die Hauptaufgabe eines *IE*-Systems die Extraktion von semantischen Informationen aus verschiedenen Texten ist. Verschiedene Arten von semantischen Informationen können, abhängig von der Größe der linguistischen Einheiten, die auf den Extraktionsprozess und den linguistischen Kontext gerichtet sind, von jedem Text extrahiert werden. [Moens 2006, S. 36-38]

In der Zukunft können, wie in Abbildung 3.25 zu sehen ist, *IE*-Systeme stufenförmige Systeme sein, bei denen der Output des einen *IE*-Systems der Input für ein anderes System ist. [Moens 2006, S. 42]

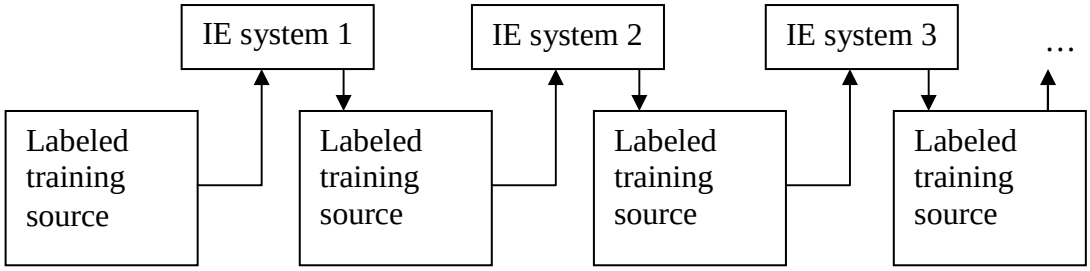


Abbildung 3.25: Ein stufenförmiges *Information Extraction*-System [Moens 2006, S. 42]

Aufgaben

Es gibt eine Reihe von typischen Aufgaben eines *IE*-Systems. In Tabelle 3.6 werden diese Aufgaben kurz aufgelistet und für jede die Extraktionseinheiten, der linguistische Kontext und das letztendliche Ziel der Aufgabe beschrieben. [Moens 2006, S. 38]

Tabelle 3.6: Aufgaben eines *Information Extraction*-Systems [Moens 2006, S. 41]

Information Extraction task	**Extraction unit**	**Linguistic context**	**Eventual goal**
Named entity recognition	Word/Word group	Sentence/text	Entity understanding
Noun phrase coreference resolution	Word/Word group	Sentence/text/multiple texts	Entity understanding
Semantic role recognition	Word/Word group	Sentence	Sentence understanding
Entity relation recognition	Words/Word groups	Sentence/text/multiple texts	(Multi-text) discourse/story understanding
Timeline extraction	Words/Word groups	Sentence/text/multiple texts	(Multi-text) discourse/story understanding

Named entity recognition

Es werden genannte Ausdrücke, wie z. B. Personen, Orte oder Firmen, in einem Text erkannt und klassifiziert. Für das Beispiel *„Max Mustermann arbeitet für die Post.“* ist „Max Mustermann“ die Person und die „Post“ die Firma. [Moens 2006, S. 38-41]

Noun phrase coreference resolution

Zwei oder mehr Hauptsätze sind *coreferent*, wenn sie sich auf dieselbe beschriebene Situation beziehen. Viele Referenzen in einem Text sind als *phoric*-Referenzen verschlüsselt, wie z. B. linguistische Elemente, die direkt die Bedeutung von einem Element verschlüsseln, beziehen sich auf eine direkte Beschreibung des Elements, die sich vor oder nach der Referenz im Text befindet. Sie werden entsprechend als *anaphoric*- und *cataphoric*-Referenzen bezeichnet.

Beispiel: *„Max Mustermann fährt nach Magdeburg. Er ist momentan in München.“*

Hierbei beziehen sich „Max Mustermann“ und „Er“ auf das gleiche Element. „Er“ ist in diesem Beispiel eine *anaphoric*-Referenz. [Moens 2006, S. 38-41]

Semantic role recognition

Diese Aufgabe analysiert die Abgrenzung der semantischen Rollen zu den syntaktischen Bestandteilen eines Satzes. Sie betrachtet dabei bestimmte Aktionen und Zustände, ihre Mitwirkenden und ihre Umstände. Die semantischen Rollen können allgemein oder spezifisch definiert werden. Auf diese Rollen wird später in diesem Kapitel genauer eingegangen. [Moens 2006, S. 38-41]

Entity relation recognition

Die Beziehung zwischen zwei oder mehr Entitäten ist ermittelt und möglicherweise durch eine semantische Rolle beschrieben. Bei dem oben erwähnten Beispiel *„Max Mustermann arbeitet bei der Post.“* ist „Max Mustermann“ die Person, „Post“ die Firma und „arbeitet bei der“ die Beziehung zwischen den beiden anderen. [Moens 2006, S. 38-41]

Timeline extraction

Eine andere Aufgabe ist das Aufspüren der temporären Ausdrücke in einem Text. Diese Ausdrücke können absolut wie z. B. „2. Juli“ oder relativ wie z. B. „gestern“ sein. Ausdrücke können auch Zeiten über eine Dauer („eine Stunde“), Ereignis-verankert („zwei Tage bevor Weihnachten“) oder eine Menge von Zeiten („jede Woche“) sein.

Klassische temporäre Beziehungen zwischen verschiedenen Ereignissen können beispielsweise „X vor Y“, „X während Y“, „X startet Y“, „X beendet Y“, „X trifft Y“ oder „X ist gleich Y“ sein. [Moens 2006, S. 38-41]

Beispiel: *„Am **12.03.2008** hab ich meine Prüfung bestanden. Die **drei Wochen** davor habe ich nur gelernt.“*

Diese Aufgaben sind bereichsunabhängig. Sie ermöglichen es, viele Details bezüglich eines Events zu identifizieren. Bereichsabhängige Extraktionsaufgaben können definiert werden, um die Beschreibung eines Events zu komplementieren. *Information Extraction* ist daran interessiert, die Informationen zu einem individuellen Event, die Lage der Teilnehmer in diesen Events und ihre räumliche, zeitliche und kausale Situation zu extrahieren. [Moens 2006, S. 38-41]

3.1.4.3 Techniken und Methoden

Für das *IE* existieren eine Reihe von Techniken und Methoden zum Extrahieren von Informationen. An dieser Stelle werden die Symboltechnik und die Pattern-Erkennung vorgestellt.

Symboltechniken

Bei den Symboltechniken werden verschiedene Rollen definiert. *Picture Producers* (*PP*) symbolisieren hierbei die physikalischen Objekte und *Acts* (*ACT*) einfache Aktionen. *Picture Aiders* (*PA*) modifizieren die *PPs* und sind gewöhnlich ein Zustand mit einem speziellen Wert (z. B. Attribut) und *Action Aiders* (*AA*) modifizieren die *ACTs*. Zusätzlich gibt es die Rollen *Location* (*LOC*) und *Time* (*T*). [Moens 2006, S. 47-48]

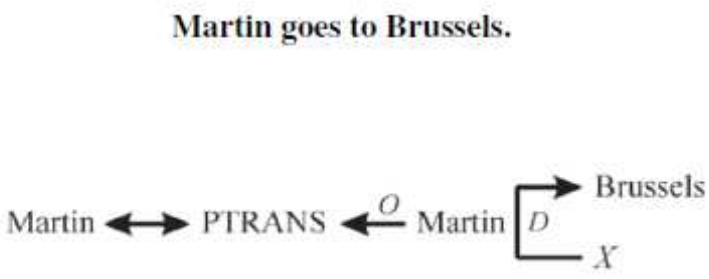


Abbildung 3.26: CDT-Repräsentation [Moens 2006, S. 48]

Der Kern der *Conceptual Dependency Theory* (*CDT*) ist eine Menge von elf *ACTs*, welche es ermöglichen, alle Aktionen eines Akteurs, die er möglicherweise in der realen Welt ausführen könnte, abzubilden. Beispielsweise ist *PTRANS* eines der am häufigsten genutzten *ACTs* und drückt eine Änderung des Orts von einem *PP* aus. Dies wird durch die *PTRANS*-Konstruktion transponiert, wie in Abbildung 3.26 zu sehen ist. [Moens 2006, S. 47-48]

Der Doppelpfeil in der Abbildung symbolisiert eine Beziehung zwischen einem ACT und einem Akteur. *O* bildet eine objektive Beziehung ab und *D* bedeutet eine direkte Beziehung. Im Beispiel ist „Martin“ das physikalische Objekt, das eine Änderung des Orts nach „Brussels“ von einem unbekannten Ort (*X*) durchführt.

Eine weitere Möglichkeit der Symboltechnik ist die Frame Theory. Ein Frame repräsentiert das Wissen, das auf ausgewählte relevante Bestandteile begrenzt ist und anhand von Typ-Wert-Paaren dargestellt wird. Die Typen, wie in Abbildung 3.27 zu sehen ist, sind konstant für ein bestimmtes Frame. Die Werte des Frames sind leere Slots und werden nach und nach mit den entsprechend gefundenen Werten aus den Texten gefüllt. Eine wichtige Aufgabe ist es, die Slots mit den richtigen Daten zu füllen. [Moens 2006, S. 47-48]

DATE		
Year	:	[yyyy]
Month	:	[*m-name* ∈ {January,…, December}]
Month-no	:	[*m-number* = integer and 0 < *m-number* ≤ 12; procedure *p1* for calculating *m-number* if *m-name* is given]
Day	:	[*d-name* ∈ {Monday,…,Sunday}]
Day-no	:	[*d-number* = integer and 0 < *d-number* ≤ length(month)]

Abbildung 3.27: Einfaches Beispiel eines Frames [Moens 2006, S. 54]

Pattern-Erkennung

Wie vorher beschrieben, konzentriert sich *Information Extraction* auf das Erkennen bestimmter Informationen in Texten und das beruht auf Pattern-Erkennungsmethoden. Pattern-Klassifikation wird das System genannt, das automatisch die Pattern in Klassen oder Kategorien einsortiert. *IE* nutzt eine Menge von Pattern in Form von Regeln oder einer Grammatik.

Pattern-Erkennung unterscheidet Objekte basierend auf den Pattern, die das Objekt aufweist, und ordnet diese Klassen oder Kategorien zu. Diese Objekte besitzen eine Reihe von Eigenschaften mit ihren Werten. Ein Objekt kann durch einen Vektor von Eigenschaften beschrieben werden: $x = [x_1, x_2, ..., x_p]^T$, wobei p die Anzahl der Eigenschaften ist. Die Eigenschaften können lexikalisch, syntaktisch und semantisch sein. [Moens 2006, S. 65-74]

3.1.4.4 Evaluation von IE-Techniken

IE-Systeme müssen natürlich auch bewertet werden, um beispielsweise ihre Wirksamkeit oder Effektivität beurteilen und sie mit anderen Systemen vergleichen zu können. Klassische Bewertungsgrößen sind beispielsweise die beiden Maße *Precision* und *Recall*, die im Kapitel 3.1.1.2 eingeführt worden sind. [Moens 2006, S. 181-182] Weitere klassische Bewertungsgrößen sind das *F-Maß*, *Fallout*, *Error Rate* und *Accuracy*. Die *Error Rate* und *Accuracy* sind bereits in Kapitel 3.1.2.2.3 beschrieben worden. *Fallout* bezeichnet die Proportion von nicht korrekten Ergebnissen bezogen auf die Anzahl nicht korrekter Ergebnisse, die das System maximal generieren könnte. Im Gegensatz zu *Precision* und *Recall*, die im Idealfall den Wert 1 annehmen sollten, ist der Wert von *Fallout* im Idealfall 0. Das *F-Maß* ist eine übliche Metrik für das Kombinieren von *Precision* und *Recall* in einer Größe, wobei P *Precision*, R *Recall* und β einen Faktor darstellt, der die relative Wichtigkeit von *Precision* und *Recall* beschreibt. Wenn β den Wert 1 annimmt, sind die beiden Werte P und R gleich wichtig und das Maß wird als *F_1-Maß* bezeichnet. [Moens 2006, S. 181-182]

$$F = \frac{(1+\beta^2)\cdot(P\cdot R)}{(\beta^2\cdot P + R)}$$

Die Definition der richtigen Bewertungsmessgrößen ist eine schwierige Aufgabe, wobei verschiedene Gewichte bestimmter Fehlertypen ein gewisses Maß an Subjektivität und Kontextabhängigkeit in den Bewertungsprozess einbringen. [Moens 2006, S. 181-182]

3.2 ISO Standards

Die Arbeit soll auf einem ISO-Standard basieren. In diesem Abschnitt werden zwei Spezifikationen, der *ISO/IEC-Standard 26300* und der *ISO/IEC-Standard 29500*, vorgestellt und analysiert. Die Eigenschaften, die Vor- und die Nachteile werden betrachtet und zum Schluss miteinander verglichen.

3.2.1 ISO/IEC 26300 – Open Document Format for Office Applications

Der *ISO/IEC-Standard 26300* ist ein offenes XML-basiertes Dokumentenformat für Büroanwendungen, die Texte, Tabellen, Präsentationen, Charts oder graphische Elemente beinhalten können. Als Dateiendung werden *.odt* für Texte, *.ods* für Tabellen, *.odp* für Präsentationen und *.odg* für Zeichnungen benutzt. *ODF* (*Open Document Format for Office Applications*) wird zurzeit in Programmen, wie z. B. *OpenOffice.org*, *Google Text und Tabellen*, *Lotus Symphony*, *KOffice* und teilweise *Microsoft Office*, unterstützt. Die Anzahl an Programmen, die das *Open Document Format* verwenden, steigt stetig. Zurzeit unterstützen mehr als 40 Anwendungen und ungefähr 15 Länder das ISO-standardisierte Format. Die NATO hat das Format in ihre Liste vorgeschriebener Standards zur Schaffung von Interoperabilität aufgenommen. [Krempl 2008] [Durusau, Brauer, Oppermann 2007, S. 33-34] *ODF* ist ein ZIP-Archiv, das aus einer Sammlung verschiedener XML-Dateien besteht:

- *manifest.xml*
- *meta.xml*
- *settings.xml*
- *styles.xml*
- *content.xml*

Bei diesen XML-Dateien steht, wie in Abbildung 3.28 zu sehen ist, der aktuelle Inhalt des Dokuments in der Datei *content.xml*. Aufgrund dessen ist sie die wichtigste XML-Datei aus der Sammlung. In *content.xml* werden die Formatvorlagen, zugeordnet zu den einzelnen Textpassagen im Dokument, dargestellt. In der *settings.xml* werden die allgemeinen Einstellungen, z. B. ob leere Seiten gedruckt werden oder ob der Seitenhintergrund gedruckt wird, für das Dokument gespeichert. Die *styles.xml* beinhaltet Style-Eigenschaften des Dokuments. Beispiele für Style-Typen, die in der *content.xml* zur Anwendung kommen, sind Seiten-Style, Absatz-Style und Buchstaben-Style. [Durusau, Brauer, Oppermann 2007, S. 39-40]

```
<text:h style-name="Heading_2">DIES IST EIN TITEL</text:h>
<text:p style-name="Text_body"/>
<text:p style-name="Text_body">
   Dies ist ein Absatz. Die Formatinformationen dazu stehen im
   Style Text_body. Das leere text:p oben ist eine leere Zeile.
</text:p>
```

Abbildung 3.28: Auszug einer *content.xml*-Datei

Die *meta.xml* enthält Metainformationen zum Dokument, unter anderem den Namen des Bearbeiters und das Datum der letzten Bearbeitung. Weitere Metainformationen sind

beispielsweise die Anzahl der Bilder, Tabellen, Seiten oder Wörter. Die *manifest.xml* fasst, wie in Abbildung 3.29 zu sehen ist, die anderen eben beschriebenen XML-Dateien zusammen. Dementsprechend können durch die XML-Dateien alle wichtigen Informationen des Dokuments ausgelesen und bearbeitet werden. [Durusau, Brauer, Oppermann 2007, S. 39-40]

```
<file-entry media-type="text/xml" full-path="content.xml"/>
<file-entry media-type="text/xml" full-path="styles.xml"/>
<file-entry media-type="text/xml" full-path="meta.xml"/>
<file-entry media-type="" full-path="Thumbnails/thumbnail.png"/>
<file-entry media-type="" full-path="Thumbnails/"/>
<file-entry media-type="text/xml" full-path="settings.xml"/>
```

Abbildung 3.29: Auszug einer *manifest.xml*-Datei

3.2.1.1 Eigenschaften

In diesem Kapitel werden die wichtigsten Eigenschaften des *Open Document Format Standards ISO/IEC 26300* vorgestellt.

Es gibt verschiedene Standards, die einzelne Features von Office Anwendungen unterstützen. Beispiele dieser Standards sind SVG für graphische Inhalte sowie HTML und XSL-FO für Textinhalte. Aber keiner der Standards unterstützt alle Features von Office Anwendungen – im Gegensatz zu *Open Document Format for Office Applications*. [Macnaghten 2007, S. 2-3]

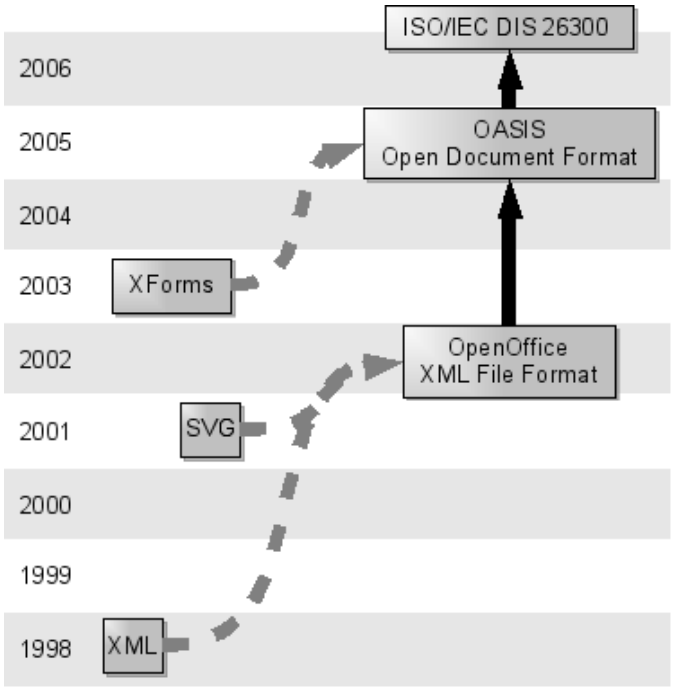


Abbildung 3.30: Die im *ODF* verwendeten Standards stammen aus unterschiedlichen Zeiten [Zendel 2006]

Kompatibilität / Interoperabilität

Für eine einfache Transformation und maximale Interoperabilität und Kompatibilität zu anderen Formaten und Plattformen benutzt das Format etablierte Standards wie z. B. HTML, SVG, XSL, SMIL, XLink, XForms, MathML und Dublin Core. [Durusau, Brauer, Oppermann 2007, S. 34-35] Abbildung 3.30 zeigt eine historische Verbindung zwischen *ODF* und den eben erwähnten Standards. *ODF* ermöglicht es, eine Vielzahl von Bild-, Audio-, Video- und Texttypen zu verwenden. In Tabelle 3.7 sind einige erlaubte Typen kurz dargestellt. [Durusau, Brauer, Oppermann 2007, S. 713]

Tabelle 3.7: Beispiele verschiedener Typen [Durusau, Brauer, Oppermann 2007, S. 713]

Bild	**Audio**	**Video**	**Text**
gif	mpeg	mp4	css
jpeg	smv	mpeg	csv
tiff		quicktime	html
png		raw	sgml
ief			xml
cgm			

Internationalisierung

Im *ISO/IEC-Standard 26300* werden alle Sprachen, die im *ISO-Standard 639* und *ISO-Standard 3166* aufgelistet werden, unterstützt. Darin inbegriffen ist die Möglichkeit, sowohl links-nach-rechts, rechts-nach-links als auch bidirektionale Sprachen zu verwenden. [Durusau, Brauer, Oppermann 2007, S. 731] Beim Textfluss können verschiedene Einstellungen beispielsweise „vor den Text“ und „mit Text in Zeile“ vorgenommen werden. [Durusau, Brauer, Oppermann 2007, S. 677]Mit numerischen, alphanumerischen und römischen Zahlenformaten stehen drei der wichtigsten Zahlenformate zur Verfügung. [Durusau, Brauer, Oppermann 2007, S. 454] Kalendarische Daten können in den Formaten gregorianisch, japanisch, chinesisch, koreanisch, arabisch-islamisch, jüdisch und buddhistisch dargestellt werden. Diese verschiedenen Formate und Einstellungen stellen einen sehr hohen Grad an Internationalisierung dar. [Durusau, Brauer, Oppermann 2007, S. 524-525]

Geringe Grenze für Entwicklungseinführung

Ein erfahrener Entwickler kann schnell und mit einer kurzen Einarbeitungszeit in die Spezifikation eine *ODF*-Anwendung schreiben. Dies ist durch eine kurze, übersichtliche und intuitiv erklärte Spezifikation möglich. [Durusau, Brauer, Oppermann 2007]

Kompaktheit

Open Document Format XML-Dateien werden in einem ZIP-Archiv gespeichert, um die Daten zu packen und zu komprimieren. Im Gegensatz zu den vorher genutzten binären Dateiformaten, die sehr kryptisch und schwierig zu verarbeiten waren, garantiert das ZIP-Format relativ kleine Dateigrößen und reduziert damit Datenspeicher- und Übertragungsbandbreitenanforderungen. *ODF* ist das erste allgemeingültig verwendete Dokumenten-Datei-Format, das das ZIP-Konzept mit verschiedenen XML-Dateien nutzt. Bei verschiedenen Anwendungstypen, wie z. B. Textverarbeitung und Tabellenkalkulation, wird jeweils dieselbe Sammlung von XML-Dateien verwendet. Die Definitionen für Elemente (z. B. Tabellen) sind konsistent über die verschiedenen Anwendungstypen. [Durusau, Brauer, Oppermann 2007, S. 709-710]

Modularität

Das Benutzen der oben aufgezählten XML-Dateien macht den Zugriff und damit die Ansicht auf den Dokumenteninhalt, sonstigen Eigenschaften und Informationen zu dem Dokument einfach, da der Inhalt und damit auch die Einstellungen und Formatierungen nicht monolithisch sind. Die XML-Dateien können mit einem einfachen Texteditor geöffnet und bearbeitet werden. [Durusau, Brauer, Oppermann 2007, S. 39-40]

3.2.2 ISO/IEC 29500 – Office Open XML file formats (OOXML)

Im Dezember 2006 wurde die Formatspezifikation *Office Open XML* bei der *International Organization for Standardization* (*ISO*) vorgelegt. Im März 2008 wurde die Spezifikation zunächst als ISO-Standard anerkannt. Anschließend legten vier nationale Standardisierungsgremien Einspruch gegen eine Veröffentlichung als ISO/IEC Norm ein. Am 18.11.2008 wurde entgegen den Einsprüchen diverser Länder die Spezifikation als *ISO/IEC 29500:2008, Information technology – Office Open XML formats* veröffentlicht. [Beld 2006] [Frost, Brougham 2007] [Frost 2008] [Frost, Buck 2008]

ISO/IEC-Standard 29500 ist ein offener Standard für Textverarbeitungsdokumente (.docx), Präsentationen (.pptx) und Tabellenkalkulationen (.xlsx). Er kann von unterschiedlichen Anwendungen auf verschiedenen Plattformen implementiert werden. Vorteile des Standards sind u. a. Stabilität, Erhaltung und Kompatibilität. Es ist möglich, Dokumente automatisch aus Unternehmensdaten zu generieren. Weiterhin können diese aus Dokumenten extrahiert und Unternehmensanwendungen mit diesen gefüllt werden.

```
- <w:body>
- <w:p w:rsidR="003B0A72" w:rsidRDefault="006452D4" w:rsidP="006452D4">
- <w:pPr>
  <w:pStyle w:val="Überschrift1" />
  </w:pPr>
- <w:r>
  <w:t>DIES IST EIN TITEL</w:t>
  </w:r>
  </w:p>
  <w:p w:rsidR="006452D4" w:rsidRDefault="006452D4" />
- <w:p w:rsidR="006452D4" w:rsidRDefault="006452D4" w:rsidP="006452D4">
- <w:r w:rsidRPr="00CD41B3">
  <w:t>Dies ist ein Absatz.</w:t>
  </w:r>
- <w:r>
  <w:t xml:space="preserve"></w:t>
  </w:r>
  </w:p>
- <w:sectPr w:rsidR="006452D4" w:rsidRDefault="003B0A72">
  <w:pgSz w:w="11906" w:h="16838" />
  <w:pgMar w:top="1417" w:right="1417" w:bottom="1134" w:left="1417"
   w:header="708" w:footer="708" w:gutter="0" />
  <w:cols w:space="708" />
  <w:docGrid w:linePitch="360" />
  </w:sectPr>
  </w:body>
```

Abbildung 3.31: Auszug einer *document.xml*-Datei

Dokumente dieser Spezifikation bestehen aus einer ZIP-Datei mit folgender Containerstruktur:

- *[Content_Types].xml*
- *_rels*
- *_rels/.rel*
- *docProps/core.xml*
- *word/document.xml*

Die XML-Datei *[Content_Types].xml* beschreibt den Inhalt des Pakets sowie ein Abbild der Dateiendungen für spezifische *URIs* (*Uniform Resource Identifier*). Das Verzeichnis *_rels* enthält die Beziehungen für die Dateien innerhalb des Pakets. Um Beziehungen für eine spezifische Datei zu finden, muss in dem *_rels* Verzeichnis nach einer Datei mit dem Dateinamen und der Endung *.rels* gesucht werden (z. B. *[Content_Types].xml.rels*). In der Datei *docProps/core.xml* sind die Kerneigenschaften jedes *OOXML*-Dokuments enthalten. Die Datei *word/document.xml* ist der Hauptteil jedes Word-Dokuments; sie beinhaltet den Kerninhalt des Dokuments. In Abbildung 3.31 wird ein Beispiel vorgestellt. In Abbildung 3.32 werden die Hauptkomponenten des *OOXML* dargestellt. [Ngo 2008, S. 10-13]

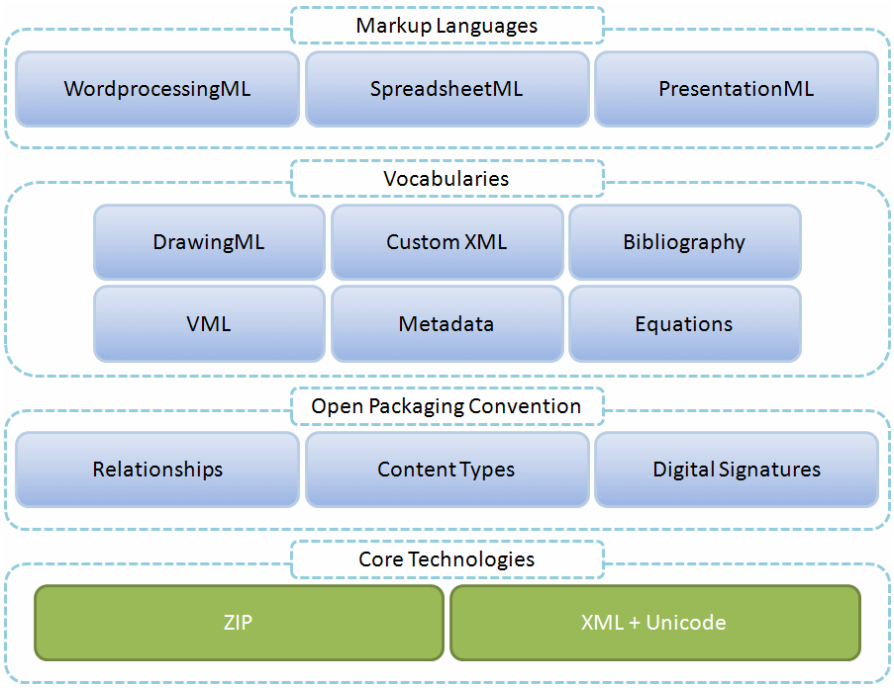


Abbildung 3.32: Hauptkomponenten des *OOXML* [Pattison, Predeek, van Vugt 2007]

3.2.2.1.1 Eigenschaften

In diesem Kapitel werden die wichtigsten Eigenschaften des *Office Open XML Standards ISO/IEC 29500* dargestellt und erläutert.

Kompatibilität / Interoperabilität

Entwickler haben die Möglichkeit, Anwendungen, die *Office Open XML* benutzen, auf verschiedenen Plattformen zu entwickeln. Ein wichtiger Aspekt der Kompatibilität ist die Unabhängigkeit einzelner spezifischer Typen von Inhaltsquellen. *Office Open XML* beinhaltet keine Restriktionen an Bild-, Audio- und Videotypen. Die Spezifikation vermeidet Abhängigkeiten bezüglich der Laufzeitumgebung der Anwendung, die das Dokument erstellt. Ein weiterer wichtiger Bestandteil der Kompatibilität über alle Plattformen und Betriebssysteme ist die Übereinstimmung zu offenen W3C-Standards wie XML. [Ngo 2008, S. 3-4]

Internationalisierung

Bei dem *ISO/IEC-Standard 29500* werden internationale Features, die von diversen Sprachen wie Arabisch, Chinesisch, Hebräisch, Japanisch, Koreanisch, Russisch oder Türkisch benötigt werden, unterstützt. Bei der Textorientierung sind links nach rechts, rechts nach links und bidirektionale Sprachen, wie z. B. Arabisch, Urdu oder Hebräisch, möglich. Der Textfluss kann bei der Textverarbeitung eingestellt werden und es werden

alle potentiellen Textlayouts unterstützt. Zusätzlich existieren viele Zahlenformate wie z. B. numerisch, alphanumerisch oder römisch. Kalendarische Daten können in verschiedenen Formaten wie gregorianisch, hebräisch oder japanisch dargestellt werden. [Ngo 2008, S. 5]

Geringe Grenze für Entwicklungseinführung

Ein erfahrener Entwickler ist in der Lage, schnell mit einer kurzen Lesezeit für die Spezifikation eine einfache *Office Open XML*-Anwendung zu schreiben. Obwohl die Spezifikation eine große Menge an Features beschreibt, müssen *Office Open XML*-Anwendungen nicht alle Features der Spezifikation unterstützen. [Ngo 2008, S. 5-6]

Kompaktheit

Office Open XML-Dateien werden gewöhnlich in einem ZIP-Archiv gespeichert, um die Daten zu packen und zu komprimieren. *Office Open XML*-Dateien sind im Durchschnitt 25% kleiner, jedoch maximal 75% kleiner als ihr binärer Gegenpart. [Ngo 2008, S. 6-7]

Modularität

Eine Anwendung kann viele Aufgaben durch Parsen oder Modifizieren einer kleinen Untermenge eines Dokuments ausführen. Ein *OOXML*-Dokument ist nicht monolithisch, da es aus mehreren Teilen gebildet wird. Beziehungen zwischen diesen werden ihrerseits in einzelnen Teilen gespeichert. Das ZIP-Format für *Office Open XML*-Dateien unterstützt den wahllosen Zugriff auf jeden Teil. [Ngo 2008, S. 7]

Hohe Migrationsgenauigkeit

Office Open XML ist entwickelt worden, um alle Features von Microsoft Office 97-2003 zu unterstützen. Es wird bei *Office Open XML* beabsichtigt, Änderungen in der Zukunft auf gleichem Abstraktionsgrad zu zulassen. [Ngo 2008, S. 7-8]

Integration mit Unternehmensdaten

Es wird Unternehmen ermöglicht, Produktivitätsanwendungen mit Informations-systemen, die Geschäftsprozesse managen, zu integrieren, indem sie das Benutzen von

maßgeschneiderten Schemata mit *Office Open XML*-Dokumenten ermöglichen. [Ngo 2008, S. 8-9]

Raum für Innovationen

Office Open XML wurde entwickelt, um Entwickler anzuregen, neue Anwendungen zu implementieren, die bei der Definition der Spezifikation nicht betrachtet wurden. [Ngo 2008, S. 9]

3.2.3 Vergleich ISO/IEC 26300 gegenüber ISO/IEC 29500

Beim *ODF* steht die Entwicklung eines Mechanismus, der existierende Standards verwendet und maximale Interoperabilität bietet, im Vordergrund. Im Gegensatz dazu wurde *OOXML* von Microsoft hauptsächlich für die Office Suite 2007, die nur mit der Microsoft-Umgebung operiert und Inkonsistenzen mit existierenden Standards aufweist, entwickelt. Im Folgenden werden die beiden Spezifikationen kritisch miteinander verglichen. Tabelle 3.8 stellt die Unterschiede der beiden ISO-Formate in einer Übersicht dar.

Namen der Spezifikationen

Der Name *Office Open XML* (*OOXML)* wird oft mit *Open Office XML* verwechselt, dem ursprünglichen Namen des *ODFs* in seiner frühen Entwicklung. Dadurch kann es zu Verwirrungen beim Leser kommen, der denkt, dass der *ISO/IEC-Standard 29500* (*OOXML*) der *ISO/IEC-Standard 26300* (*ODF*) ist bzw. zu dem OpenOffice.org Produkt gehört. Eine Projektion sowohl der negativen als auch der positiven Eigenschaften des einen Standards auf den anderen ist nicht ausgeschlossen. Jemand, der mit der Thematik nicht intensiv vertraut ist, kann dadurch sehr schnell verwirrt bzw. irritiert sein. [o. V. 2007]

Tags und Tagnamen

Tags sind Terme im XML, die durch „<“ und „>“ umschlossen werden. Ein Tag, der mit einem „/“ beginnt und dem Namen am Anfang entspricht, ist in der Syntax das Ende eines Blocks. *OOXML* hat kürzere Tagnamen als *ODF*, benötigt dafür jedoch eine höhere Anzahl von Tags, wobei dies Einfluss auf den Speicherplatz und die Geschwindigkeit, die für das Parsen der Daten und das Konvertieren in interne Strukturen benötigt werden, hat. *ODF* verwendet gewöhnlich Wörter als Tagnamen, die

den XML-Konventionen für Tagnamen entsprechen. Dies erleichtert die Interoperabilität und steigert die Parsingperformance, vergrößert jedoch den Speicherplatz. Die Unterschiede beim Speicherplatz und der Performance sind jedoch durch das ZIP-Archiv, in dem beide Formate gepackt sind, minimal. [Macnaghten 2007, S. 6-13]

ODF

```
<text:p text:style-name="Standard">
       Dies ist ein Beispieltext.
</text:p>
```

OOXML

```
<w:p>
       <w:pPr>
              <w:pStyle w:val="Heading1"/>
       </w:pPr>
       <w:r>
              <w:t>
                     Dies ist ein Beispieltext.
              </w:t>
       </w:r>
</w:p>
```

Abbildung 3.33: Beispiel für Tagnamen der beiden Spezifikationen

Wesentliche Unterschiede kommen dagegen bei der Lesbarkeit und Klarheit der Tagnamen, die die Akzeptanz des Formats bei den Entwicklern beeinflusst, vor. Wenn der Standard kryptisch wie *OOXML* ist, sind zusätzliche Ressourcen notwendig, um ihn zu implementieren und zu erhalten. In jedem Format sind sehr viele Tags und umso intuitiver diese sind, desto leichter ist es für die Entwickler, diese zu verstehen und Bezug darauf zu nehmen. In Abbildung 3.33 sind die Unterschiede zwischen den beiden Spezifikationen dargestellt. Aus dem Beispiel zu *OOXML* kann nicht intuitiv interpretiert werden, was dieser Ausdruck bedeutet. Sind die Bedeutungen, wie bei dem Beispiel zu *ODF*, der Tagnamen eindeutig, kann die Bereitstellung des Formats schneller und günstiger durchgeführt werden. [Macnaghten 2007, S. 6-13]

Span Feature – Wechsel der Formatierung

Spanning ist die Methode, um Attribute eines Absatzes in einem Bereich zu verändern. Eine Methode, diese Veränderung in einem XML-Dokument zu speichern, ist das Setzen eines Tags zu Beginn des Absatzes und eines abschließenden Tags am Ende des Absatzes. *ODF* unterstützt dieses Feature mit der Zeichenfolge *text:span*, wie in Abbildung 3.34 zu sehen ist. *OOXML* dagegen unterstützt dies nicht. Der Standard benötigt das Ende des Bereiches und das Beginnen eines neuen Bereiches zu Beginn der Änderung. Um den ersten Bereich weiterzuführen, muss nochmals ein neuer Bereich

gestartet werden. Dadurch wird das Dokumentenformat komplexer und unübersichtlicher. [Macnaghten 2007, S. 7]

ODF

```
<text:p text:style-name="Standard">
       Dies enthält
       <text:span text:style-name="T1">
              fett gedruckte Wörter und
       </text:span>
       <text:span text:style-name="T2">
              kursiv geschriebene Wörter.
       </text:span>
…
</text:p>
```

OOXML

```
<w:p>
       <w:r>
              <w:t>
                     Dies enthält
              </w:t>
       </w:r>
       <w:r>
              <w:rPr>
                     <w:b/>
              </w:rPr>
              <w:t>
                     fett gedruckte Wörter und
              </w:t>
       </w:r>
       <w:r>
              <w:rPr>
                     <w:i/>
              </w:rPr>
              <w:t>
                     kursiv geschriebene Wörter.
              </w:t>
       </w:r>
…
</w:p>
```

Abbildung 3.34: Repräsentation einer Formatänderung in *ODF*- und *OOXML*-Notation

Verwenden bestehender Standards

Das Verwenden bereits existierender Standards vereinfacht den Prozess, ein konformes Produkt zu kreieren. Das ist auch ein offensichtliches Ziel von Standards. Beispielsweise werden Standards wie z. B. *xlink*, *SVG*, *SMIL* oder *HTML* bei *ODF* verwendet. *OOXML* nutzt außer XML und Microsoft Standards selbst keine anderen etablierten Standards. [Macnaghten 2007, S. 2-3]

Tabelle 3.8: Vergleich zwischen *OOXML* und *ODF*

Merkmal	**OOXML**	**ODF**
Dateien	*.docx	*.odt
	*.pptx	*.odp
	*.xlsx	*.ods
Tags und Tagnamen	kurze, kryptische Tagnamen	lange, sprechende Tagnamen
Span Feature	Attribute können innerhalb eines Blocks nicht verändert werden	Attribute können innerhalb eines Blocks verändert werden
Verwendete Standards	XML, Microsoft Standards	XML, SVG, MathML, XLink, SMIL, XForms, XSL:FO
Repräsentation eines Datums	als Integer (Anzahl an Tagen seit 31.12.1899)	JJJJ-MM-TT
Containerstruktur		
Dokumenteninhalt	/word/document.xml	content.xml
Dokumentoptionen	/docsProps/core.xml	settings.xml
Verknüpfungen	/_rels/.rels	/META/manifest.xml
Zielgruppe	Microsoft Office Nutzer	Nutzer für Office-Dokumente
Entwicklerfreundlichkeit	teilweise entwickler-unfreundlich z. B.	entwicklerfreundlich z. B.
	- kryptische Tagnamen	- sprechende Tagnamen
	- Inkonsistenzen mit ISO-Standards	- Verwenden von ISO-Standards
	- operiert hauptsächlich mit der Microsoft Umgebung	- hohe Interoperabilität
Verfügbarkeit	frei verfügbar	frei verfügbar
Spezifikation	über 4000 Seiten	ca. 700 Seiten
Verwendung	bisher nur in MS Office 2007 implementiert	in über 25 Programmen implementiert

Repräsentation eines Datums

Wichtig bei der Verarbeitung von Daten ist das Datum. Die beiden Spezifikationen speichern ein Datum unterschiedlich. *OOXML* speichert beispielsweise den 5. März 2007 als Zahl (39146). Dies ist die Anzahl von Tagen seit dem 31. Dezember 1899, wobei ein kleiner Fehler in der Spezifikation auftritt. Das Jahr 1900 ist kein Schaltjahr, *OOXML* behandelt es aber wie ein solches. *ODF* stellt den 5. März 2007 als „2007-03-05“ in Übereinstimmung mit dem existierenden ISO-Standard 8601 dar. [Macnaghten 2007, S. 14] Abbildung 3.35 stellt dieses Beispiel in der jeweiligen Notation dar.

ODF

```
<table:table-cell table:style-name="ce4" office:value-type="date"
office:date-value="2007-03-05">
      <text:p>
            3/5/2007
      </text:p>
</table:table-cell>
```

OOXML

```
<c r="B1" s="1">
      <v>
            39146
      </v>
</c>
```

Abbildung 3.35:Repräsentation des Datums "05.03.2007" in *ODF*- und *OOXML*-Notation

Konsistenz von Tabellen mit dem Rest der Spezifikation

Beim *ODF* werden die Spezifikationen für die Tabellenkalkulation genauso behandelt wie Tabellendefinitionen in Textverarbeitungsdokumenten. Die Tags und Attribute sind, im Gegensatz zu denen bei *OOXML*, die gleichen. Bei *OOXML* sind diese nicht als ein Teil der Tabelle gespeichert, sondern in der *sharedStrings.xml*. Dadurch wird die Portabilität in der Praxis erschwert. [Macnaghten 2007, S. 15]

Tabelle 3.9: Standards mit denen *OOXML* Inkonsistenzen hat [Macnaghten 2007, S. 20]

Standard	Inkonsistenzen (anderes Wort?)
ISO 216 - Page size Names	- *OOXML* nutzt eigenen numerischen Code o Beispiel „A4“: ▪ ISO 216 („A4“) <> *OOXML* („9“)
ISO 639 - Language codes	- *OOXML* nutzt eigenen Sprachencode, der inkonsistent mit dem ISO 639 ist o Beispiel: „Deutschland“: ▪ ISO 639 („ger“) <> *OOXML* („1031“)
ISO/IEC 15445 - Colour Names	- Einige Farbdefinitionen in *OOXML* sind anders als in dem Standard o Beispiel: „Green“ (hexadezimal): ▪ ISO 15445 (008000) <> *OOXML* (0000FF00)
ISO 8601 - Dates and times	- *OOXML* repräsentiert Daten als Integer o Beispiel: DATEVALUE("01-Feb-2006") ▪ ISO 8601 (2006-02-01) <> *OOXML* (38749.0000000…)
ISO 8879 - Readable XML	- Tagnamen sind sehr kryptisch und menschlich nicht leicht lesbar o Beispiele: `scrgbClr, algn, dir`

Widersprüche in OOXML

Da mit *ODF* schon ein Standard für die Interoperabilität von Office Produkten entwickelt wurde und den meisten Anforderungen der Industrie entspricht, ist es nicht notwendig einen konkurrierenden Standard mit *OOXML* zu entwickeln. Die Kosten, einen weiteren Standard zu implementieren, wären für die Industrie, die Regierungen und anderen Institutionen sehr hoch. [Macnaghten 2007, S. 19]

Weiterhin hat der *ISO/IEC-Standard 29500* und damit *OOXML* Inkonsistenzen mit existierenden ISO-Standards, wobei in Tabelle 3.9 einige dieser beschrieben werden. Dabei kristallisiert sich heraus, dass *OOXML* weitestgehend auf Standards von Microsoft zurück greift und existierende Standards, die sich bereits bewährt haben, nicht verwendet. [Macnaghten 2007, S. 20-21]

Tabelle 3.10: *OOXML* Referenzen zu externen oder redundanten nicht Standard-Ressourcen [Macnaghten 2007, S. 22]

Ressource	**Inkonsistenzen**
Vector graphics	- *OOXML* hat seine eigenen Vektorgraphiken – DrawingML. o Dazu gehörender Standard ist SVG.
Objects	- *OOXML* bezieht sich auf „Windows Metafiles“ und „Enhanced Metafiles“, die beide veraltete Microsoft Formate sind.
Configuration	- *OOXML* beinhaltet spezifische Anwendungs-konfigurationseintellungen, wie z. B. autoSpaceLikeWord95 oder footnoteLayoutLikeWW8.
Percentages	- *OOXML* ist bei der Repräsentation von Prozentwerten in sich selbst inkonsistent. - Prozentangaben z. B. o als Dezimalzahl (bei Wiedergabeeinstellungen), ▪ <w:zoom w:percent=“71“ /> als 71% o reale Prozentzahl * 50 (bei Tabellen-Breite-Maßeinheiten) ▪ Beispiel: 4975 = 99,5% o reale Prozentzahl * 1000 (als generische Prozent-Maßeinheit) ▪ Wert von 1 = 0,001%, 100000 = 100%

In Tabelle 3.10 werden die Referenzen zu Ressourcen dargestellt, die redundant zu bestehenden Standard-Ressourcen sind und von *OOXML* verwendet werden. *OOXML* verwendet meistens eigene Referenzen zu Ressourcen, die zum Teil veraltet oder nicht als Standard anerkannt sind. Außerdem werden beispielsweise Prozentangaben inkonsistent, durch drei unterschiedliche Repräsentationen, dargestellt. [Macnaghten 2007, S. 22-23]

Zusammenfassung

Offene Standards sollten breit eingesetzt werden. Sie sollten durch geringe Barrieren zum Implementieren eine größtmögliche Übereinstimmung bei allen Beteiligten erreichen. Der Standard ist offen, sobald er von jeder Software implementiert werden darf und gut dokumentiert ist. Office-Dokumentenformatstandards sollten nicht zu weit auseinander gehen. Zusammenfassend ist die Philosophie von *OOXML*, die Daten, die Microsoft Office 2007 nutzt, zu repräsentieren, die Dateigröße klein zu halten und die Performance zu verbessern. Der Schwerpunkt beim *ODF* liegt dagegen auf der Interoperabilität, die die Implementierung und den Zugriff auf die Daten vereinfacht. [Macnaghten 2007, S. 24]

Auswertung

Der Vergleich der beiden ISO-Standards *ISO/IEC 29500* (*OOXML*) und *ISO/IEC 26300* (*ODF*) hat gezeigt, dass das *OOXML*-Format einige Schwächen und Inkonsistenzen aufweist. Hinzu kommt, dass dieses Format hauptsächlich mit der Microsoft-Umgebung operiert. *ODF* steht im Gegensatz dazu für Interoperabilität und benutzt viele etablierte Standards. Das Ausgabeformat sollte auf einem ISO-Standard basieren, der möglichst weit anerkannt ist und genutzt werden kann. Zurzeit ist *ODF* das Format, das bereits in vielen Ländern und Institutionen verwendet wird. Aus diesem Vergleich der Spezifikationen ergibt sich die Nutzung des *ODF* Formats – *ISO/IEC 26300* – für das Ausgabedokument des neuen Systems.

3.3 Auslesen bzw. Konvertierung von T_EX-Dokumenten

Da das Programm zur automatischen Erstellung des Jahresberichts auf der Basis des *ISO/IEC-Standards 26300* (*Open Document Format*) implementiert wird, sollen alle T_EX-Dokumente des Jahresberichts zu Open Office-Dokumenten konvertiert werden. Die Konvertierung von T_EX-Dokumenten ist mittels mehrerer Programme auf der Basis verschiedener Programmiersprachen und Plattformen möglich. Diese CASE-Tools (Computer-Aided Software Engineering) zur Unterstützung des Programms zur Erstellung des Jahresberichts werden, nachdem im folgenden Abschnitt die notwendigen Anforderungen an diese Tools definiert wurden, analysiert.

3.3.1 Anforderungen an die Konvertierung

Bei der Konvertierung von T_EX- in Open Office-Dokumente müssen einige Faktoren beachtet werden. Dafür haben sich, wie in Tabelle 3.11 zu sehen ist, folgende

funktionale, qualitative und systembezogene Anforderungen an die CASE-Tools ergeben:

Tabelle 3.11: Anforderungen an das Konvertierungsprogramm

Funktionale Anforderungen	Konvertieren von T_EX- in Open Office-Dokumente
Qualitative Anforderungen	Plattformunabhängigkeit
	Auslesen von nicht vollständigen T_EX-Dokumenten
	Leichte Installation und Bedienung der Software (*EN ISO 9241*)
	Gute Dokumentation
Systembezogene Anforderungen	Java-Bibliothek
	Möglichst wenig zusätzliche Programme
	Open Source

Konvertieren von T_EX- in Open Office-Dokumente (funktionale Anforderung)

Die Anforderung an das CASE-Tool besteht darin, ohne Informationsverlust T_EX- in Open Office-Dokumente zu konvertieren. Das entstehende Dokument soll für ein effektives Auslesen der Daten möglichst strukturiert sein.

Plattformunabhängigkeit (qualitative Anforderung)

Plattformunabhängigkeit bedeutet, dass ein Programm auf verschiedenen Computersystemen, wie z. B. Microsoft Windows oder Linux, mit Unterschieden in der Architektur, dem Prozessor, dem Compiler und dem Betriebssystem, lauffähig ist. Der Grad der Plattformunabhängigkeit wird Portabilität genannt. Zusätzlich zu der bestehenden Plattformunabhängigkeit wird hierbei der geschätzte Aufwand behandelt, der notwendig ist, um das CASE-Tool in ein vollständig plattformunabhängiges zu transformieren. Da das Tool auf verschiedenen Plattformen funktionieren soll, ist es wichtig, dass die Programme, die die T_EX-Dokumente konvertieren, unabhängig von der Plattform sind.

Auslesen von nicht vollständigen T_EX-Dokumenten (qualitative Anforderung)

Da T_EX-Dokumente durch das neue System in dieser Arbeit von jedem Mitarbeiter abgegeben werden und nicht wie bisher in einem Masterdokument der Institute zusammengefasst werden, soll das Programm zusätzlich Teil-T_EX-Dokumente konvertieren können. Ein *vollständiges* T_EX-Dokument muss alle obligatorischen Bestandteile enthalten. Obligatorisch sind *\documentclass[parameter]{klasse}*, *\begin{document}* und *\end{document}*. Diese Bestandteile könnten durch die Software zur Erstellung des Jahresberichts hinzugefügt werden, besser wäre jedoch, wenn das

Konvertierungstool auch die Dokumente behandelt, die diese Teile nicht beinhalten. In Abbildung 3.36 ist ein Beispielaufbau eines vollständigen TEX-Dokuments dargestellt.

```
\documentclass[a4paper]{book}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
%%
\begin{document}
\chapter{…Kapitel-Überschrift}
\subsection{ … }
%% Hier Text
\end{document}
```

Abbildung 3.36: Beispiel für den Aufbau eines TEX-Dokuments

Leichte Installation und Bedienung der Software (qualitative Anforderung)

Die Installation eines Programms soll intuitiv und minimal komplex sein. Eine komplizierte Installation kann zu Fehlern und Irritationen beim Anwender führen. Die nachträgliche Einrichtung des Programms soll einfach und verständlich sein, um eine effektive Nutzung von diesem zu ermöglichen. Auch die Nutzung des Programms soll möglichst intuitiv sein. Eine übersichtliche GUI (Graphical User Interface) und Tooltipps bei den Funktionen erhöhen die Bedienerfreundlichkeit des Programms. Gemäß ISO-Standard *EN ISO 9241* muss die GUI folgende Merkmale aufweisen:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Fehlertoleranz
- Individualisierbarkeit
- Lernförderlichkeit

Gute Dokumentation (qualitative Anforderung)

Eine gute Dokumentation eines Programms, in der alle Funktionen und mögliche Probleme beschrieben werden, kann bei der Nutzung helfen und langes Suchen vermeiden. Zu der Dokumentation gehören Handbücher, Hilfefunktionen im Programm aber auch Online-Dokumentationen, die mehrsprachig umgesetzt sein sollten. Fachbegriffe sollten zum besseren Verständnis in einem Glossar erklärt werden.

Java-Bibliothek (systembezogene Anforderung)

Bibliotheken sind Sammlungen von Programmfunktionalitäten, jedoch keine eigenständigen Einheiten, sondern Hilfsmodule, die einem Nutzer zur Verfügung stehen, um beispielsweise Operationen auszuführen und Programme anzusprechen. Da die Software für die Erstellung des Jahresberichts in Java implementiert wird, ist es hilfreich, wenn Funktionen für Java, beispielsweise Methoden von OpenOffice.org, um Open Office-Dokumente anzusprechen, bereitgestellt werden.

Möglichst wenig zusätzliche Programme (systembezogene Anforderung)

Für das CASE-Tool sollen möglichst wenig bis keine zusätzlichen Programme benötigt werden, da die meisten zusätzlichen Programme nicht vorhanden bzw. nicht auf den Arbeitsplätzen installiert sind.

Open Source (systembezogene Anforderung)

Um die Kosten für das Tool zu minimieren, sollen alle notwendigen CASE-Tools und die Programme, die die Tools zum Ausführen benötigen, *Open Source* sein. Dies sind Lizenzen für Software, von denen die Quelltexte zugänglich und Weiterentwicklungen gewünscht sind.

3.3.2 Analyse der Anforderungen an den Beispielprogrammen

Tabelle 3.12: Vergleich verschiedener Programme zur Konvertierung

	TeX4ht	**TeX2RTF**	**LaTeX2HTML**	**LaTeX2hyp**	**Hyperlatex**
Konvertieren von T_EX- in Open Office-Dokumente	++	+	+	+	+
Leichte Installation und Bedienung der Software	-	+	-	-	-
Gute Dokumentation	+	+	++	-	++
Plattformunabhängigkeit	+	++	++	++	-
Java-Bibliothek	--	--	--	--	--
Auslesen nicht vollständiger T_EX-Dokumente	--	--	--	--	--
Keine zusätzliche Programme	++	+	--	++	++
Open Source	++	++	++	++	++

(++: voll erfüllt, +: weitestgehend erfüllt, -: teilweise nicht erfüllt, --: überhaupt nicht erfüllt)

Im Folgenden werden ein paar ausgewählte Programme zum Konvertieren von T_EX- in Open Office-Dokumente nach den eben erläuterten Anforderungen analysiert und

bewertet. Tabelle 3.12 enthält eine Übersicht der behandelten Konvertierungsprogramme.

TeX4ht

Tex4ht ist ein Open Source-Programm zur Konvertierung von T_EX- in HTML- und Open Office-Dokumente, das einen großen Grad an Plattformunabhängigkeit besitzt. Das Programm kann auf den Betriebssystemen MS Windows und UNIX gestartet werden und unterstützt somit zwei der größten, zurzeit auf dem Markt befindlichen, Systeme. Die Bedienung ist wenig intuitiv, da es keine GUI für dieses Programm gibt. Um die Konvertierung, die bei diesem Programm nur bei einem vollständiges T_EX-Dokument möglich ist, durchzuführen, muss in einer Batchdatei ein komplexer Befehl mit mehreren Parametern eingegeben werden. In der Dokumentation zu dem Programm sind nicht alle Parameter und deren Varianten erklärt. Zum Ansprechen des Programms über Java existieren keine Java-Bibliotheken. Weitere Programme werden zur Ausführung des Programms nicht benötigt.

TeX2RTF

Dieses Open Source-Programm konvertiert L^AT_EX- in HTML- und RTF-Dokumente. Aus den RTF-Dokumenten können anschließend PostScript-, PDF- oder Open Office-Dokumente generiert werden. *TeX2RTF* hat einen sehr hohen Grad an Plattformunabhängigkeit, lässt sich intuitiv über die GUI bedienen und es werden nur kleine zusätzliche Programme, wie *make* und ein C-Compiler, benötigt, um mit *Tex2RTF* arbeiten zu können. [Partosch 1997] Allerdings wird ein vollständiges T_EX-Dokument benötigt und es stehen keine Java-Bibliotheken zur Verfügung.

LaTeX2HTML

LaTeX2HTML ist ein Open Source-Konvertierungstool, das L^AT_EX- in HTML-Dokumente konvertiert. Die Ausführung des Programms erfolgt nicht über eine GUI, was die Bedienung erschwert. Zur Erklärung des Programms existiert eine ausführliche Dokumentation. Der Grad der Plattformunabhängigkeit ist hoch, da die beiden großen Systeme Windows und UNIX unterstützt werden. Zum Ausführen werden jedoch diverse andere Programme, wie z. B. *Ghostscript*, *Perl*, *DBM*, *dvips* und T_EX, benötigt. [Partosch 1997] Da zu *LaTeX2HTML* keine Java-Bibliotheken zur Verfügung stehen und nur vollständige T_EX-Dokumente konvertiert werden können, ist dieses Programm keine optimale Lösung zur Konvertierung der T_EX-Dokumente.

LaTeX2hyp

Das Open Source-Programm *LaTeX2hyp* ist ein C-Programm und konvertiert L^{A}T$_{E}$X- in Text-, HTML- und RTF-Dokumente. Es läuft auf den Systemen UNIX und MS-DOS und hat somit einen sehr hohen Grad an Plattformunabhängigkeit. Eine übersichtliche GUI zur einfachen Bedienung existiert nicht und die Dokumentation zum Programm ist zwar ausführlich, aber unstrukturiert. Es sind keine Java-Bibliotheken vorhanden und vollständige T$_{E}$X-Dokumente für das Konvertieren notwendig. Zum Ausführen benötigt das Programm weder T$_{E}$X oder L^{A}T$_{E}$X noch andere zusätzliche Programme.

Hyperlatex

Hyperlatex ist eine Open Source-Software, konvertiert L^{A}T$_{E}$X- in HTML-Dokumente und kann sowohl unter UNIX als auch VMS (Virtual Memory System) ausgeführt werden. Dadurch ist der Grad der Plattformunabhängigkeit gering und nur mit hohem Aufwand kann das Programm auf Windows portiert werden. Die Dokumentation ist sehr ausführlich und gut strukturiert. Eine GUI zur Bedienung steht nicht zur Verfügung. Dieses Programm benötigt keine zusätzlichen Programme zum Ausführen, jedoch vollständige T$_{E}$X-Dokumente. Es existieren keine Java-Bibliotheken zur Unterstützung.

3.3.3 Analyse und Bewertung

In Kapitel 3.3.2 wurden verschiedene Beispielprogramme auf die Anforderungen aus Kapitel 3.3.1 hin untersucht und bewertet. Viele der Programme sind zwar plattformunabhängig, jedoch können einige nicht genutzt werden, da sie zusätzliche Programme benötigen. Programme wie z. B. *Perl*, *Ghostscript* oder T$_{E}$X sind nicht zwangsläufig auf jedem Rechner installiert und das müsste dementsprechend erst nachgeholt werden. Zusätzlich gibt es zu keinem der Programme Java-Bibliotheken. Alle im Kapitel 3.3.2 vorgestellten Programme benötigen vollständige T$_{E}$X-Dokumente zur Konvertierung. Da die T$_{E}$X-Dokumente in der Mehrzahl unvollständig sind, würde dies zu Problemen und einem hohen Editierungsaufwand bei den Masken während der Ausführung des Konvertierungsprogramms führen. Keines der vorgestellten Programme erfüllt die vorher definierten Anforderungen in ausreichendem Maße. Aus diesem Grund muss auf die Konvertierung von T$_{E}$X-Dokumenten durch die vorgestellten CASE-Tools in Open Office-Dokumente verzichtet und ein anderer Weg gefunden werden.

3.3.4 Alternative Lösung

Da das Ziel, die T_EX-Dokumente direkt in Open Office-Dokumente zu konvertieren, aufgrund der nicht erfüllten Anforderungen der zur Verfügung stehenden CASE-Tools nicht möglich ist, werden die T_EX-Dokumente über das Programm zur Erstellung des Jahresberichts ausgelesen, verarbeitet und anschließend in einem Open Office-Dokument ausgegeben. T_EX-Dokumente können mit Hilfe eines Dateilesers[2] effektiv ausgelesen und ausgewertet werden. Eine genaue Beschreibung des Systems wird im fünften Kapitel gegeben.

[2] In diesem Fall wird der Typ *java.io.FileInputStream* zum Einlesen des T_EX-Dokuments verwendet.

4 Anwendungsbeispiel

Im dritten Kapitel wurden die Möglichkeiten der Textanalyse, unter anderem *Information Retrieval*, *Pattern Matching* und *Information Extraction,* erörtert. In diesem Abschnitt wird die Erstellung des Jahresberichts der Fakultät für Informatik an der Otto-von-Guericke-Universität als Anwendungsbeispiel vorgestellt. Bei der Erläuterung des Anwendungsobjekts wird auf den Aufbau und den aktuellen Ablauf der Erstellung des Jahresberichts eingegangen. Anschließend werden die Vorteile der aktuellen Erstellung betrachtet. Zum Schluss werden eine Schwachstellen- und diesbezüglich eine Anforderungsanalyse durchgeführt.

4.1 Aufbau des Jahresberichts

Für jedes Jahr wird ein Jahresbericht erstellt. Dieser dient einer umfassenden Darstellung der Aktivitäten, Erfolge und Leistungen der Fakultät für Informatik, die im Folgenden FIN genannt wird, und deren Institute innerhalb des entsprechenden Jahres. Zu Beginn werden die Lehrkörper, in dem Fall die Hochschullehrer, der FIN aufgeführt, die im Laufe des Jahres an der Fakultät tätig waren. Im Anschluss werden gegebenenfalls neue Professoren vorgestellt oder Antrittsvorlesungen angekündigt. Zusätzlich existiert eine Übersicht über die Verwaltungsorgane der FIN. Es werden beispielsweise die Mitglieder mit Posten des Dekanats, des Fakultätsrats, der Fachschaft oder des Senats aufgeführt. Es werden alle Studiengänge dargestellt und erläutert. Weiterhin werden Themen wie Kooperationsbeziehungen der FIN, Ehrenpromotionen oder Forschungsschwerpunkte behandelt. Nach den allgemeinen Informationen zu der Fakultät folgen Informationen zu den einzelnen vier Instituten. In jedem Institutskapitel werden, wie in Tabelle 4.1 zu sehen ist, die Themen *personelle Besetzung*, *Forschungsgebiete und –projekte*, *Vorträge und Teilnahme an Veranstaltungen*, *Veröffentlichungen*, *Lehrveranstaltungen*, *studentische Arbeiten* und *Sonstiges* behandelt. Es sind jedoch nicht alle Eingabefelder obligatorisch.

Tabelle 4.1: Themen und Eingabefelder im Jahresbericht

Thema	**Eingabefelder**	
Personelle Besetzung	Name	Titel
Forschungsgebiete und –projekte	Bezeichnung*	Projektpartner
	Mission-Statement	Fördersumme
	Projekttitel*	Laufzeit
	Projektträger*	Bearbeitung
	Förderkennzeichen	Kurzbeschreibung*
	Projektleitung*	
Veröffentlichungen (Bücher und Publikationen)	Dokument*	Organisation
	Begutachtet*	Herausgeber
	Kürzel*	Jahr
	Autor*	Monat
	Titel*	Schlüssel
	Querverweis	Hinweis
	Buchtitel	Kommentar
	Seiten	Sprache
	Redakteur	Volume
	Band	Ort
	Ausgabe	ISBN
	Reihe	Edition
	Adresse	URL
Vorträge und Teilnahme an Veranstaltungen	Betreff*	Veranstaltung*
	Titel* (nur bei Vorträge)	Ort*
	Person*	Zeit*
Lehrveranstaltungen	Titel*	Lehrbeauftragter*
	Stunden Vorlesung*	Stunden Übung*
	Stunden Praktikum*	Zielgruppe*
Studentische Arbeiten	Ersteller*	Art*
	Betreuer*	Thema*
Thema	**Unterthemen**	
Sonstiges	Gremientätigkeiten	Mitgliedschaften
	Gastaufenthalte	Gutachtertätigkeiten
	Mitarbeit in Programmkomitees	Was sonst noch wichtig war

(* Pflichtfelder)

4.2 Interviews

Im Zuge der Analyse der aktuellen Erstellung des Jahresberichts wurden Interviews mit an dem Jahresbericht beteiligten Mitarbeitern verschiedener Arbeitsgruppen bzw.

Institute der FIN und dem Jahresberichtsverantwortlichen durchgeführt. Folgende Punkte standen bei den Interviews im Mittelpunkt:

- Wie werden die Informationen für den Jahresbericht zusammengetragen?
- Welche Probleme ergeben sich beim Zusammentragen der Informationen?
- Welche Vorteile existieren bei der aktuellen Lösung?
- Welche Verbesserungen könnten beim Zusammentragen der Informationen und bei der Erstellung des Jahresberichts hilfreich sein?
- Ist die aktuelle Lösung mit den T_EX-Masken effektiv und intuitiv? Falls nein, existieren bessere Formate als T_EX?

Anhand dieser Diskussionspunkte kann der Ablauf der Erstellung (Kapitel 4.3) des bestehenden Systems analysiert und beschrieben werden. In den Kapiteln 4.4, 4.5 und 4.6 werden auf der Grundlage der Interviews die Vorteile und Schwachstellen der aktuellen Erstellung des Jahresberichts hervorgehoben und analysiert. Aus diesen können anschließend die Anforderungen für die zu implementierende Software zur Erstellung des Jahresberichts abgeleitet werden.

4.3 Aktueller Ablauf der Erstellung

In den letzten Jahren ist der Prozess der Erstellung des Jahresberichts sehr komplex, zeitintensiv und unterliegt vielen Fehlerquellen. An der Erstellung arbeiten verschiedene Personen, die in Abbildung 4.1 in einem Organigramm dargestellt sind, mit unterschiedlichen Methoden und Werkzeugen, um Informationen für den Jahresbericht zu sammeln. Für die Erstellung gibt es eine zentrale Person, den Jahresberichts-verantwortlichen, der alles organisiert und bei dem alle „Teilberichte“ zusammengefasst werden. In Abbildung 4.2 wird u. a. der Arbeitsablauf des Jahresberichts-verantwortlichen in einer ereignisgesteuerten Prozesskette (EPK) dargestellt. An der FIN übt diese Tätigkeit zurzeit Herr Dr. Bernd Reichel aus. Er erstellt die T_EX-Masken für die Themengebiete des Jahresberichts, die zusammenfassenden T_EX-Masken für die Institute und die Haupt-T_EX-Datei, in der zum Schluss alles zusammengefügt wird. In der Haupt-T_EX-Datei stehen beispielsweise die Style- und andere Eigenschaften des Jahresberichts**.**

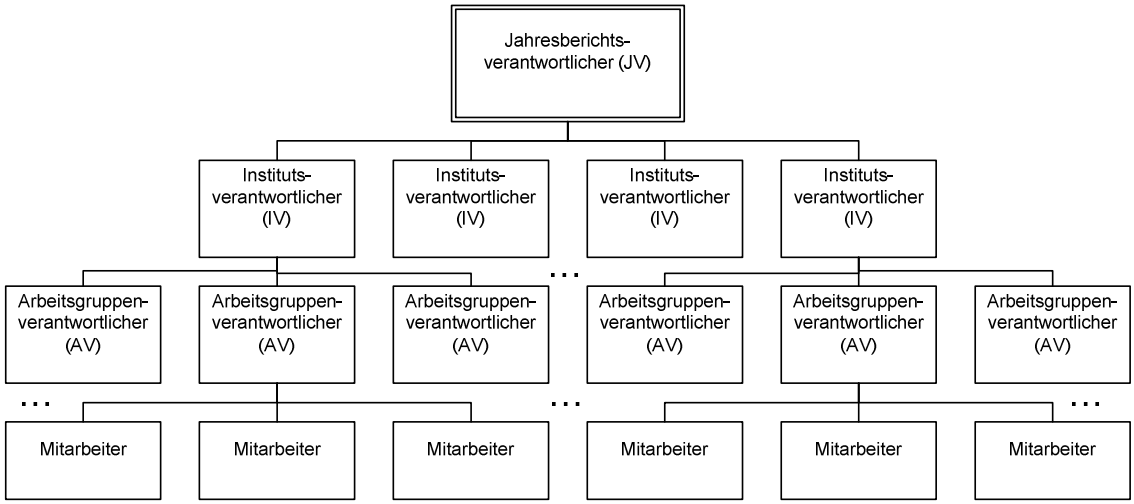


Abbildung 4.1: Organigramm der Erstellung

Zur Unterstützung des Jahresberichtsverantwortlichen gibt es je einen Instituts-verantwortlichen, der wiederum von Arbeitsgruppenverantwortlichen, im Folgenden Institut und Arbeitsgruppe genannt, unterstützt wird. Die Verantwortlichen müssen vorher von den Instituten bzw. Arbeitsgruppen bestimmt und dem Jahresberichts- bzw. Institutsverantwortlichen gemeldet werden.

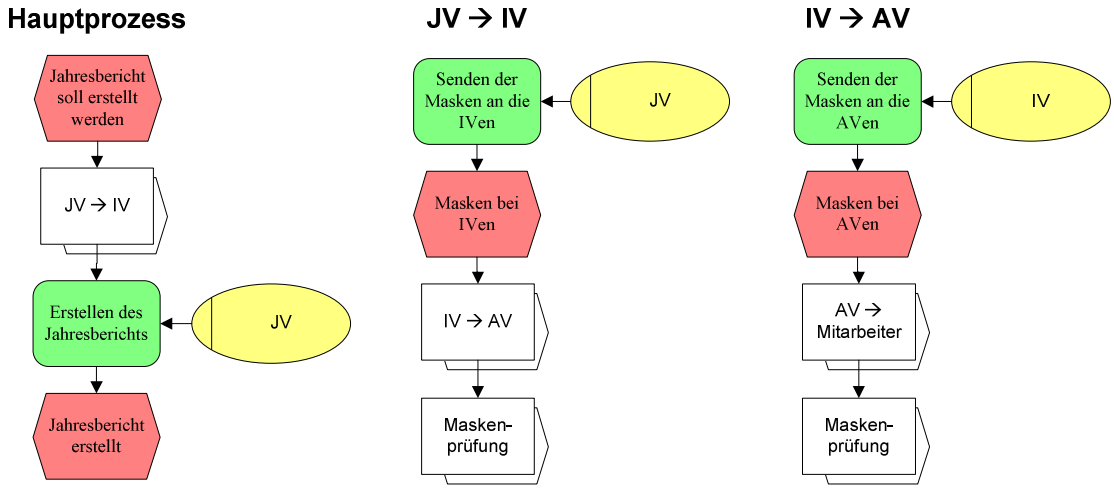


Abbildung 4.2: Hauptprozess, JV → IV und IV → AV (JV – Jahresberichtsverantwortlicher, IV – Institutsverantwortlicher, AV – Arbeitsgruppenverantwortlicher)

Der Jahresberichtsverantwortliche verteilt die vorher erstellten TEX-Masken der Themengebiete an alle Institute. In Abbildung 4.3 ist eine TEX-Maske des Themengebiets *Forschungsgebiete und –projekte* zu sehen. Für jedes Forschungsprojekt wird das gleiche Muster, das in der Abbildung zu sehen ist, verwendet. Hierbei werden die einzelnen Attribute bei jedem Projekt in der gleichen Reihenfolge angegeben. Jedes Forschungsprojekt beginnt mit einem *\projekt* und beinhaltet im Folgenden neun Einträge in geschweiften Klammern, in denen Informationen wie z. B. Projekttitel, Bearbeitung oder eine Kurzbeschreibung zu dem Projekt einzutragen sind. Die anderen

T_EX-Masken sind ähnlich der Maske des Themengebiets *Forschungsgebiete und -projekte* aufgebaut. In Abbildung 4.2 ist der Ablaufplan für einen Instituts-verantwortlichen zu sehen. Die Institute versuchen die notwendigen Informationen zusammen zu stellen, indem sie jeder Arbeitsgruppe des Instituts die jeweiligen Masken zum Ausfüllen zur Verfügung stellen. Die Arbeitsgruppen fügen die Informationen mit Hilfe der Mitarbeiter zusammen. Sobald die Arbeitsgruppen ihre Masken vollständig ausgefüllt und auf Korrektheit überprüft haben, schicken sie diese an das zuständige Institut. Dieses überprüft seinerseits ebenfalls die Korrektheit und Vollständigkeit der T_EX-Masken und schickt diese bei Auftreten größerer Fehler an die betroffene Arbeitsgruppe zur Behebung zurück. Kleine Fehler, u. a. Rechtschreibfehler, korrigiert das Institut aus Effizienzgründen oft selbst.

```
\projekt[
  {}, % Projekttitel
  {}, % Projektträger
  {}, % Förderkennzeichen
  {}, % Projektleitung (Vorname Name)
  {}, % Projektpartner
  {}, % Fördersumme (gesamt / 2003)
  {}, % Laufzeit in der Form 'Monat Jahr -- Monat Jahr'
  {}, % Bearbeitung (Vorname Name, etc)
  {}  % Kurzbeschreibung
]
```

Abbildung 4.3: Beispiel einer T_EX-Maske (*Forschungsgebiete und –projekte*)

Bei Korrektheit und Vollständigkeit der Masken schicken die Institute diese an den Jahresberichtsverantwortlichen. Er kontrolliert die Masken und fügt sie dann zu dem Jahresbericht zusammen. Im Zuge dessen kontrolliert er noch einmal die Korrektheit der Informationen. Diese Überprüfung hat eine Schlüsselposition, da danach der Jahresbericht erstellt und zum Drucken weitergegeben wird.

4.4 Vorteile der bisherigen Erstellung

Die bisherige Erstellung des Jahresberichts hat zwei Vorteile. Einer besteht darin, dass ein einheitliches Format benutzt wird. Dadurch müssen diejenigen, die die ausgefüllten Masken zusammenführen, sich nicht auf viele verschiedene Werkzeuge einstellen. Das heißt, aktuell benutzen alle Mitarbeiter, die an der Erstellung des Jahresberichts beteiligt sind, die T_EX-Masken, so dass keine Konflikte zwischen den Quellen entstehen können. Es existieren ein paar Ausnahmen wie Word oder Textdateien. Die Anzahl derer ist jedoch sehr gering. Als weiterer Vorteil wurde genannt, dass der Jahresbericht in T_EX immer das gleiche Layout hat, da Dr. Bernd Reichel dieses vorher definiert hat. Eine Begründung für das Beibehalten des bestehenden Erstellungssystems, die des Öfteren in

den Interviews genannt wurde, ist, dass das bestehende System funktioniert, getreu dem Motto: *„Never change a running system.“*. Daher sehen nicht alle Mitarbeiter eine Notwendigkeit das bestehende System zu ändern.

4.5 Schwachstellenanalyse

Eine Schwachstellenanalyse ist ein Qualitätsmanagement-Werkzeug zur Untersuchung eines Prozesses. Bei der Analyse werden Schwachstellen und Fehler aufgedeckt, um die untersuchten Prozesse zu optimieren. Abbildung 4.4 zeigt ein Ursache-Wirkung-Diagramm zur Steigerung der Effektivität. Aus den Interviews haben sich die im Folgenden näher beschriebenen Schwachstellen der aktuellen Erstellung heraus kristallisiert.

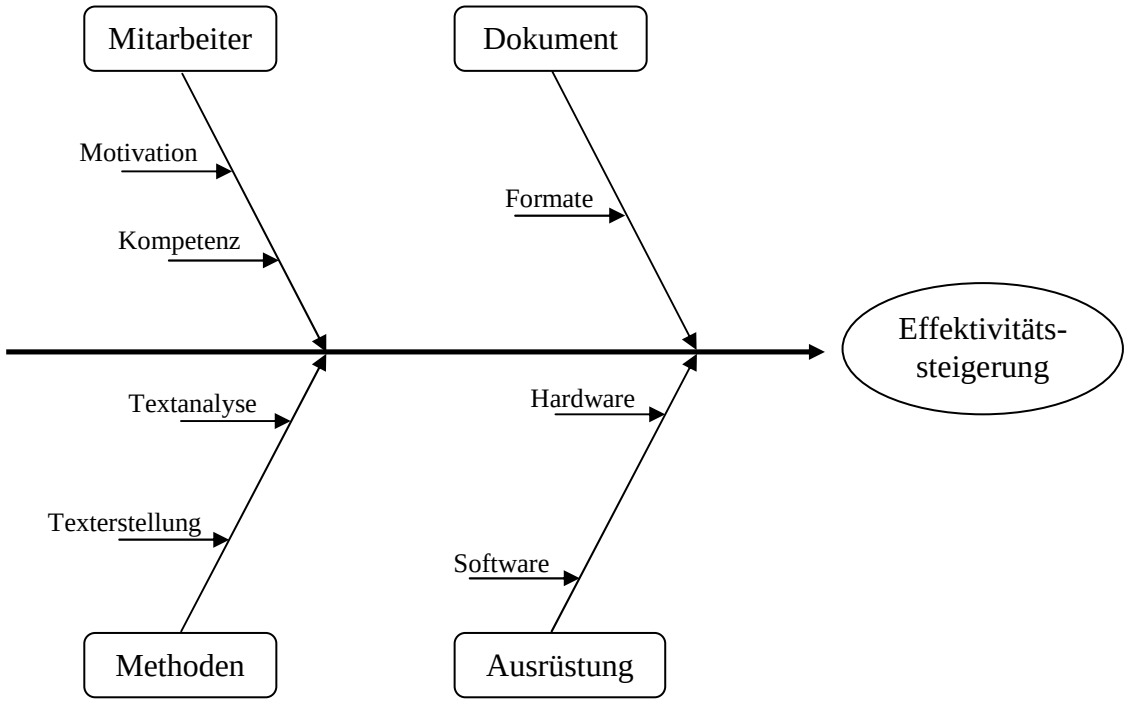


Abbildung 4.4: Effektivitätssteigerung bei der Erstellung des Jahresberichts

Arbeitsumgebung T_EX

Zur Eingabe der Informationen stehen die T_EX-Masken und nur in Ausnahmefällen Word- oder Textdokumente zur Verfügung. Da nicht zwangsläufig an allen Arbeitsplätzen T_EX installiert ist, muss die Installation gegebenenfalls nachgeholt werden. Dies erfordert nicht nur den Zeitaufwand zur Installation der Umgebung, sondern auch einen Zeitaufwand der Mitarbeiter zur Einarbeitung in das Programm. Aufgrund von

Überforderung durch die neue Software oder fehlender Motivation der Mitarbeiter kann es zu einer Unzufriedenheit mit dieser Lösung kommen**.**

Nutzung von Textanalysemethoden

Bei der Auswertung der TEX-Masken, die der Jahresberichtsverantwortliche erhält, werden bisher keine automatischen Textanalysemethoden verwendet. Zurzeit werden die TEX-Masken entweder in andere TEX-Dateien hineinkopiert oder eingebunden. Die Kontrolle der Korrektheit wird beim aktuellen Ablauf manuell durch die Verantwortlichen durchgeführt. Hierbei können Fehler, beispielsweise Übersehen von fehlenden Pflichteinträgen, auftreten, was Zeit für nachträgliches Anpassen kostet.

Manuelles Einfügen der Daten

TEX-Masken müssen manuell gefüllt werden. Das heißt, die Daten müssen von einer Quelle, beispielsweise Word oder Editoren wie z. B. *Editor* oder *Notepad*, in die TEX-Maske kopiert oder eingetragen werden. Das Ausfüllen per Hand ist sehr zeitaufwändig, die Fehleranfälligkeit (Rechtschreib- und Tippfehler, falsche Darstellung von Umlauten, etc.) ist hoch und damit einhergehend der Korrekturaufwand. Die Mitarbeiter, die die TEX-Dateien erstellen, könnten unabsichtlich die Inhalte an den falschen Stellen in der TEX-Datei platzieren, da sie mit der Arbeitsumgebung nicht ausreichend vertraut sind.

Kompilieren der TEX-Dateien

Ein weiterer Nachteil besteht darin, dass die TEX-Dateien von den Instituts-verantwortlichen so lange kompiliert und gegebenenfalls korrigiert werden müssen, bis keine Fehler mehr auftreten. Für die Institute existieren vollständige TEX-Dokumente, in der alle Daten des Instituts zusammengefasst werden müssen. Da viele der Mitarbeiter sich nicht ausreichend mit der Software und der Arbeitsumgebung auskennen, kann es länger dauern, bis die Datei fehlerfrei kompiliert wird. Der Mitarbeiter muss die Fehlermeldung richtig interpretieren und den Fehler dann korrigieren. Versteht er jedoch die Fehlermeldung nicht, braucht er für die Behebung des Fehlers fachkundige Hilfe. Das kostet wiederum ihm und dem Mitarbeiter, der ihm dann helfen muss, Zeit.

Bib-Files

Wenn Veröffentlichungen, wie in Abbildung 4.5 zu sehen ist, im T_EX-Code abgegeben werden sollen, müssen dazu Bib-Files[3] erstellt werden. Für die Syntax werden zusätzliche Einarbeitungs- und Erstellungszeit benötigt.

```
@Book{Kuerzel,
  author = {},
  title = {},
  publisher = {},
  year = {},
  month = {},
}
```

Abbildung 4.5: Beispiel eines Eintrags im Bib-File (am Beispiel eines Buchs)

Schwierig so viele Daten auf einmal zusammen zu tragen

Alle Ereignisse wie z. B. *Veröffentlichungen* oder *studentische Arbeiten* müssen von den Mitarbeitern immer von dem vergangenen Jahr zusammengetragen werden. Es hat sich herausgestellt, dass es schwierig für die Mitarbeiter ist, Ereignisse aus dem Zeitraum eines Jahres zusammen zu tragen, Daten könnten dabei verloren gehen oder in Vergessenheit geraten. Dies kann vermieden werden, indem die Informationen zum Zeitpunkt des Ereignisses eingegeben werden.

Verzögerung der Erstellung durch Schnittstellen

In Abbildung 4.6 sind die bisher vorhandenen Schnittstellen im Erstellungsprozess des Jahresberichts dargestellt. Schnittstellen sind in diesem Zusammenhang die Verbindungsstellen zwischen den einzelnen Verantwortlichen. Dadurch, dass es drei Schnittstellen gibt, eine zwischen dem Jahresberichtsverantwortlichen und den Instituten, eine weitere zwischen den Instituten und den Arbeitsgruppen und eine zwischen den Arbeitsgruppenverantwortlichen und den jeweiligen Mitarbeitern, besteht die Gefahr, dass Verzögerungen bei der Erstellung aufgrund von Verspätungen und Fehlern in den Masken auftreten. Ebenfalls können bei den Schnittstellen Kommunikationsdefizite zwischen den Beteiligten, lückenhafte Informationsübermittlung oder mangelnde Abstimmung den Prozess verlangsamen. Je mehr

[3] *BibT_EX* ist ein Programm zur Erstellung von Literaturverzeichnissen in T_EX- oder L^AT_EX-Dokumenten.

Schnittstellen in einem Vorgang existieren, desto höher ist das Fehlerpotential. Deshalb sollte die Anzahl der Schnittstellen minimal sein, um Fehlerquellen zu reduzieren.

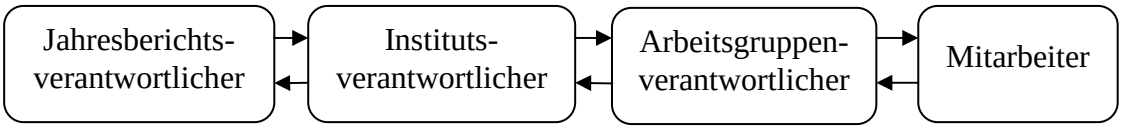


Abbildung 4.6: Schnittstellen im bisherigen Erstellungsprozess

Die identifizierten Schwachstellen, die sich im Wesentlichen auf fehlendes Know How zurückführen lassen, zeigen die Komplexität der Eingabe und machen deutlich, dass die Sekretärinnen der Arbeitsgruppen bzw. Institute die TEX-Dateien oft nicht ordnungsgemäß ausfüllen können, da ihnen dieses Wissen eventuell fehlt. Dadurch müssen die Mitarbeiter der Arbeitsgruppen, die sich mit dem Programm auskennen, dies tun. Aus diesen Schwachstellen und weiteren gewünschten Eigenschaften werden im folgenden Kapitel anhand einer Anforderungsanalyse die Anforderungen für die zu implementierende Software aufgestellt.

4.6 Anforderungsanalyse

Die Anforderungen werden in funktionale, qualitative, systembezogene und prozessbezogene unterteilt. Um keine wichtigen Eigenschaften der zukünftigen Software zu vergessen, müssen diese detailliert beschrieben werden. [Dumke 2003, S. 25] Tabelle 4.2 enthält die Übersicht der Anforderungen an die im Zuge dieser Arbeit zu entwickelnde Software.

Funktionale Anforderungen

Funktionale Anforderungen geben eine „Beschreibung der Arbeitsweise und der grundlegenden Eigenschaften der Problembezogenen Funktionalität“ [Dumke 2003, S. 25] Mit der zu entwickelnden Software sollen Dokumente verschiedenen Typs – bereits existierende TEX-Masken (z. B. Abbildung 4.3), semi-strukturierte Word-Dokumente und Word-Dokumente aus einer Vorlage – eingelesen werden. Nach der Verarbeitung und Speicherung der Daten in einer Datenbank soll die Software ein Open Office-Dokument (*ISO/IEC 26300*) erstellen, das die Informationen der eingelesenen Dateien beinhaltet.

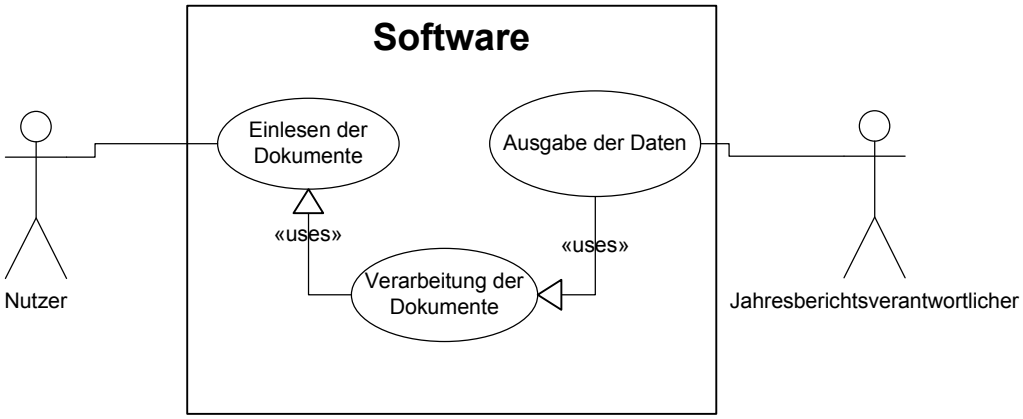


Abbildung 4.7: Use-Case-Diagramm der Jahresberichtserstellung

Qualitative Anforderungen

Bei den qualitativen Anforderungen wird die Produkt-Qualität in den verschiedensten Ausprägungen dargestellt. [Dumke 2003, S. 25] Die Technologie des Ausgabemediums soll auf den weltweit anerkannten Standard *ISO/IEC 26300* basieren. Dadurch werden die Wiederverwendung und die Universalität der Software erleichtert. Ein weiterer wichtiger Aspekt bei der Entwicklung der Software ist deren einfache und intuitive Bedienung, um lange Einarbeitungszeiten zu vermeiden. Hilfestellungen bei auftretenden Fehlern während der Anwendung der Software sollen eine möglichst gute Hilfe für Lösungen der Probleme sein. In der Bedienung sollen wenig bis keine Fehler der implementierten Software auftreten. Für die Nutzer ist es wichtig, dass die Software transparent und in einer angemessenen Zeit ausführbar ist. Weiterhin soll die Software ohne großen Aufwand erweiterbar und editierbar sein, um beispielsweise weitere Typen von Quell-Dokumenten hinzuzufügen. Dabei sollen neue Funktionen hinzugefügt und vorhandene verändert werden können.

Von den im Kapitel 3.1 analysierten Methoden und Verfahren sind alle meist nur für eine bestimmte Gruppe von Problemen relevant und auf diese anwendbar. Methoden wie z. B. *Information Extraction* oder *Information Retrieval* sind darauf spezialisiert, Dokumente zu filtern, die relevant für den jeweiligen Nutzen sein könnten. Das *Pattern Matching* ist die Methode, die bei dem neuen System aufgrund der Notwendigkeit eines Musters angewendet werden soll. Beim *Pattern Matching* werden Muster verwendet, nach denen im entsprechenden Text gesucht wird. Durch das *Pattern Matching* können die einzelnen Einträge in den T_EX-Masken und Dokumenten der Word-Vorlage identifiziert und ausgelesen werden. Auf die Anwendung der Verfahren des *Pattern Matching* auf die jeweiligen Dokumente wird im fünften Kapitel, in dem das neue System der Erstellung vorgestellt wird, näher eingegangen und beschrieben.

Systembezogene Anforderungen

Zur „Charakterisierung der erforderlichen Hardware und damit verbundenen bzw. notwendigen Software“ [Dumke 2003, S. 25] werden systembezogene Anforderungen benannt. Um auf den verschiedensten Rechnern bzw. Systemen ausgeführt werden zu können, soll die Software zur Erstellung des Jahresberichts plattformunabhängig und mindestens auf Microsoft Windows und Linux betriebsfähig sein. Die Software soll in Java entwickelt werden, da diese Programmiersprache objektorientiert ist, Java-Bibliotheken bzw. APIs (Application Programming Interface) bietet und der Großteil der Mitarbeiter der Fakultät für Informatik damit vertraut ist. Zum Speichern der Daten des Jahresberichts soll das Datenbankmanagementsystem (DBMS) *MySQL* verwendet werden. Zur Nutzung der zu implementierenden Software ist auch die Bereitstellung eines Webbrowsers nötig.

Prozessbezogene Anforderungen

Um „projektspezifische Merkmale, wie Entwicklungszeit, […] personelle und finanzielle Ressourcen“ [Dumke 2003, S. 25] zu beschreiben, werden prozessbezogene Anforderungen charakterisiert. Ziel ist die möglichst schnelle Fertigstellung der Software, damit sie in absehbarer Zeit in der Praxis getestet und verwendet werden kann.

Tabelle 4.2: Anforderungen an das Programm zur Erstellung des Jahresberichts

Funktionale Anforderungen	Einlesen von Dokumenten verschiedenen Typs (T_EX, Word)
	Speicherung der Daten in einer Datenbank
	Erstellung eines Open Office-Dokuments
Qualitative Anforderungen	Basierend auf dem Standard *ISO/IEC 26300*
	Verwenden von Textanalysemethoden – *Pattern Matching*
	Vorlage möglichst intuitiv
	Möglichst gut erweiterbar
	Hilfestellungen
	Fehlertoleranz
	Transparenz
	Performance
Systembezogene Anforderungen	Plattformunabhängigkeit
	In Java entwickelt
	Verwenden des Datenbankmanagementsystem *MySQL*
	Web Browser zur Nutzung der Webmaske
Prozessbezogene Anforderungen	Möglichst schnelle Fertigstellung

5 Konzeptionierung

In diesem Kapitel werden der grundlegende Aufbau und die Funktionsweise des neuen Systems zur Erstellung des Jahresberichts beschrieben. Im Zuge dessen werden der Ablauf und die Technologien des neuen Systems erläutert, die Vor- und Nachteile gegenübergestellt und anschließend mit dem alten System verglichen.

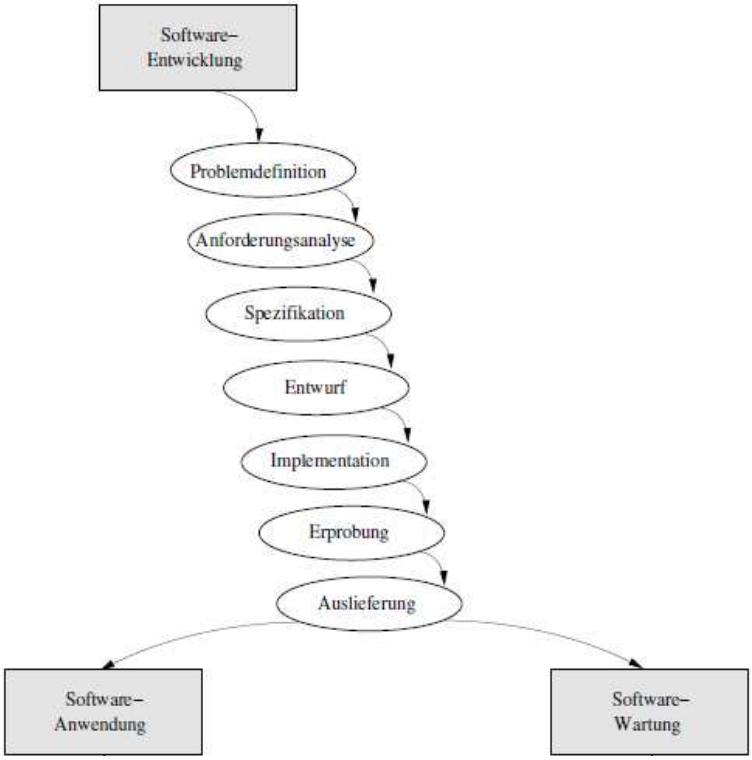


Abbildung 5.1: Phasen im Softwarelebenszyklus [Dumke 2003, S. 20]

5.1 Grundlegender Aufbau

In Kapitel 4.6 wurden die unterschiedlichen Anforderungen an die Erstellung des Jahresberichts in der Anforderungsanalyse entwickelt. Die Beschreibung des grundlegenden Aufbaus entspricht der nachfolgenden Spezifikationsphase im Softwarelebenszyklus, der in Abbildung 5.1 dargestellt ist. Die „Spezifikation ist die Formulierung aller funktionaler und einiger qualitativer Anforderungen in einem Modell, welche die Computerbezogenen und organisatorischen Systemkomponenten beschreibt“ [Dumke 2003, S. 51]. Das Modell für das neue System, das in Abbildung 5.2 dargestellt ist, wurde zu Beginn der Konzeptionierung entwickelt, um einen Überblick über die zu erfüllenden funktionalen Anforderungen für das zu lösende Problem zu erhalten.

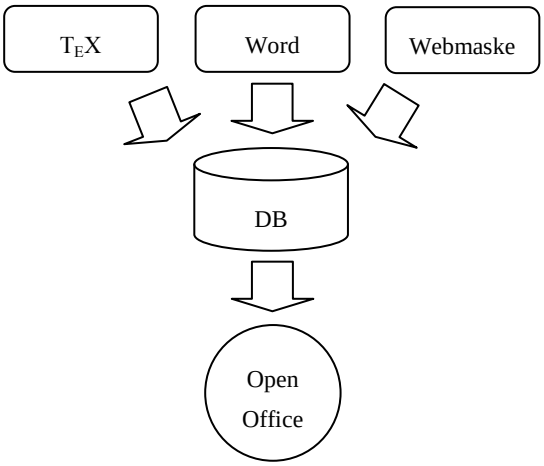


Abbildung 5.2: Modell des neuen Systems zur Erstellung des Jahresberichts

Um die Methoden, die die Nutzer als effektiv und intuitiv ansehen, zu ermitteln, wurden Interviews (Kapitel 4.2) mit Mitarbeitern der FIN durchgeführt. Die daraus herauskristallisierten Typen von Quellmedien, die in der Anforderungsanalyse (Kapitel 4.6) aufgestellt wurden, sind T_EX, Word und Webmaske. Bei den T_EX-Masken werden weiterhin die alten Masken genutzt, die schon bei der bisherigen Erstellung verwendet wurden, da viele Mitarbeiter der Fakultät während ihrer Arbeit T_EX benutzen, um beispielsweise Paper für Tagungen oder Zeitschriften zu schreiben. Einige Mitarbeiter haben in der Vergangenheit jedoch kaum oder gar nicht mit T_EX gearbeitet und würden bei der Erstellung des Jahresberichts Word-Dokumente bevorzugen. Word konnte schon als Ausnahme bei der bisher angewandten Erstellung des Jahresberichts verwendet werden, jedoch mussten die Informationen aus dem Word-Dokument manuell in eine T_EX-Maske übertragen werden. Als drittes Medium, das in diesem Modell zur Dateneingabe für den Jahresbericht genutzt werden kann, ist eine Webmaske, die in dem Seminar „Grand Management Information Design“ an der FIN entwickelt wurde. Die drei Eingabemöglichkeiten werden in den Kapiteln 5.1.2, 5.1.3 und 5.1.4 näher erläutert. Eine weitere in dem Modell umgesetzte funktionale Anforderung ist die Speicherung der Daten aus den T_EX- und Word-Dokumenten in einer Datenbank als zentralen Speicherort. Zum Schluss des Prozesses werden die Daten in ein erzeugtes Open Office-Dokument (*ISO/IEC 26300*) geschrieben. Damit sind die funktionalen Anforderungen der Anforderungsanalyse in dem Modell abgebildet. Dadurch verändert sich der Ablauf der Eingabe von Informationen in den Jahresbericht für die Nutzer. In Abbildung 5.3 wird der geplante neue Ablauf für den Nutzer mit den drei möglichen Eingabemöglichkeiten des neuen Systems beschrieben.

Abbildung 5.3: Geplanter Ablauf in dem neuen System

5.1.1 Technologien

Dieser Abschnitt beschreibt die Technologien, die bei der Entwicklung der Software zur Erstellung des Jahresberichts angewendet wurden. Tabelle 5.1 zeigt eine Übersicht der verwendeten Technologien. Zur Speicherung der Daten des Jahresberichts steht das Datenbankmanagementsystem (DBMS) *MySQL* zur Verfügung. Dieses System wurde ausgewählt, da es Open Source ist und alle erforderlichen Anforderungen an ein professionelles DBMS erfüllt. Als Sprache für die Implementierung der Software wird Java mit der *Java Runtime Environment* (*JRE*) verwendet. Java wird aufgrund seiner Objektorientierung und Plattformunabhängigkeit genutzt. Zusätzlich existieren zu dieser Sprache Bibliotheken bzw. APIs, mit denen Funktionen, wie beispielsweise das Einlesen der Dokumente oder das Generieren des Ausgabedokuments, der Software zur Erstellung des Jahresberichts unterstützt werden. Zum Einlesen von Word-Dokumenten

wird beispielsweise die API *HWPF* (*Horrible Word Processor Format*)[4] verwendet. Das Ausgabedokument wird mit Hilfe der *Uno Runtime Environment* (*URE*)[5] generiert. Die Verbindung zur *MySQL*-Datenbank wird mit Hilfe des *MySQL Connector/J*[6] aufgebaut.

Tabelle 5.1: Verwendete Technologien

Technologietyp	Technologie
Datenbank	MySQL
Programmiersprache	Java
API	JRE, HWPF, URE, MySQL Connector/J
Programme	Microsoft Word, Open Office

5.1.2 Eingabe über die Webmaske

Als zentralen Ort zur Speicherung aller Daten des Jahresberichts wurde eine Webmaske in einem Seminar entwickelt. Der Aufbau dieser Maske ist in Abbildung 5.4 dargestellt. In dieser können zu jeder Zeit des Jahres Daten eingegeben werden. Damit ist es nicht mehr notwendig, zum Zeitpunkt der Erstellung des Jahresberichts alle Informationen zusammen zu tragen. Für den Zugriff auf die Webmaske ist eine Anmeldung des Nutzers notwendig. Anschließend hat er Zugriff auf die entsprechenden Masken, in denen er seine Informationen einpflegen kann. Die Masken sind so aufgebaut, dass für jedes Themengebiet ein Formular im Web-Browser zur Verfügung steht. In den Formularen existiert für jedes Attribut des Themengebiets ein Feld zur Eingabe. Mit dem Betätigen des Buttons *Speichern* im unteren Teil des jeweiligen Formulars wird der aktuelle Eintrag in die Datenbank gespeichert. Zusätzlich zu der Eingabe werden alle weiteren Quellmedien – die ausgefüllten T_EX-Masken und Word-Dokumente – auf der Webmaske hochgeladen. Dafür wird ein Formular bereitgestellt, in das die Datei mit den Informationen zum Jahresbericht hochgeladen wird. Bei den T_EX-Masken muss der Nutzer zur besseren Zuordnung der Informationen das Jahr und das Institut angeben. Bei der ausgefüllten Word-Vorlage brauchen keine zusätzlichen Angaben getätigt werden, da das Jahr und das Institut bereits in dem Dokument selbst enthalten sind. Beim Hochladen der T_EX-Masken wird die Korrektheit des Dokuments geprüft. Anschließend wird der Inhalt der T_EX- und Word-Dokumente eingelesen und in der Datenbank gespeichert.

[4] *HWPF* ist die Schnittstelle zwischen dem Microsoft Word 97(-2007)-Dateiformat und Java.

[5] *URE* ist die Schnittstelle zwischen dem Open Office-Dateiformat und verschiedenen Programmiersprachen (z. B. Java).

[6] *MySQL Connector/J* ist ein Java-Treiber, der *JDBC* (*Java Database Connectivity*)-Aufrufe in das Netzwerkprotokoll, das von der *MySQL*-Datenbank genutzt wird, konvertiert.

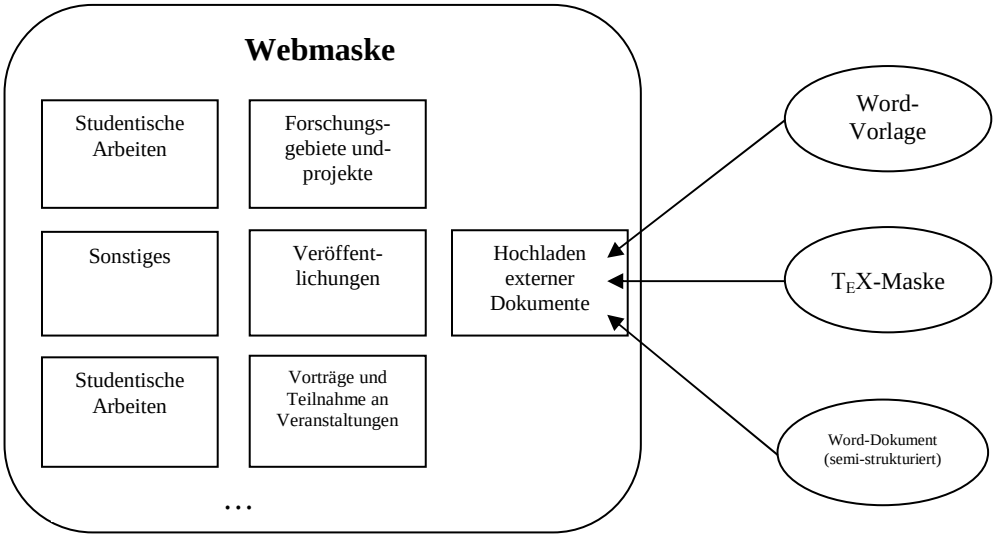


Abbildung 5.4: Aufbau der Webmaske

Die Analyse und das Einlesen der Dokumente werden in dem Kapitel 5.2 näher beschrieben. Wird ein Fehler gefunden, zeigt das System dem Nutzer eine Fehlermeldung zur Verbesserung an. Sobald der Jahresbericht als T_EX-Datei erstellt werden soll, kann über die Webmaske auf der Basis der Daten in der Datenbank eine T_EX-Datei, die den endgültigen Jahresbericht enthält, erstellt werden.

5.1.3 Eingabe über Word

Mit Hilfe einer Vorlage wird die Möglichkeit gegeben, strukturierte Word-Dokumente mit Daten zum Jahresbericht abzugeben. Die Vorlage selbst bleibt bei der Nutzung unberührt, da dafür eine Arbeitsdatei generiert wird. In der aktuellen Erstellung können nur in Ausnahmefällen semi-strukturierte Word-Dokumente oder Text-Dateien mit Daten abgegeben werden. Im Nachhinein müssen diese jedoch manuell in die L^AT_EX-Masken eingetragen oder kopiert werden. Bei der Anfertigung der Vorlage ergaben sich ein paar Hindernisse.

Tabelle 5.2: Probleme und Regeln bei der Autotext-Lösung

- *Problem:* Nutzer kann an allen des Dokuments Stellen editieren • *Regel:* Nutzer darf die Bereiche, die die Struktur darstellen, nicht verändern bzw. löschen
- *Regel:* Nutzer muss bei einem neuen Eintrag mit dem Cursor am Ende des Dokuments sein • *Problem:* Gefahr, dass neue Einträge mitten im Dokument eingefügt werden und andere durcheinander bringen (wenn der Cursor nicht am Ende des Dokuments ist) • *Problem:* Parser könnte dadurch nicht alle oder die falschen Informationen finden

Im ersten Ansatz wurde das Einfügen eines neuen Eintrags zu einem Themengebiet über Autotextelemente in Betracht gezogen. Diese Lösung hat jedoch gewisse Schwachstellen und der Nutzer müsste hierbei einige Regeln einhalten, die in Tabelle

5.2 dargestellt sind. Um das falsche Editieren der Vorlage zu verhindern, kristallisierte sich die Lösung, Formularelemente wie Schaltflächen oder Textfelder zu integrieren, heraus. Auf diese Art und Weise wird sichergestellt, dass der Nutzer nur an den Stellen etwas eintragen kann, an denen Informationen benötigt werden.

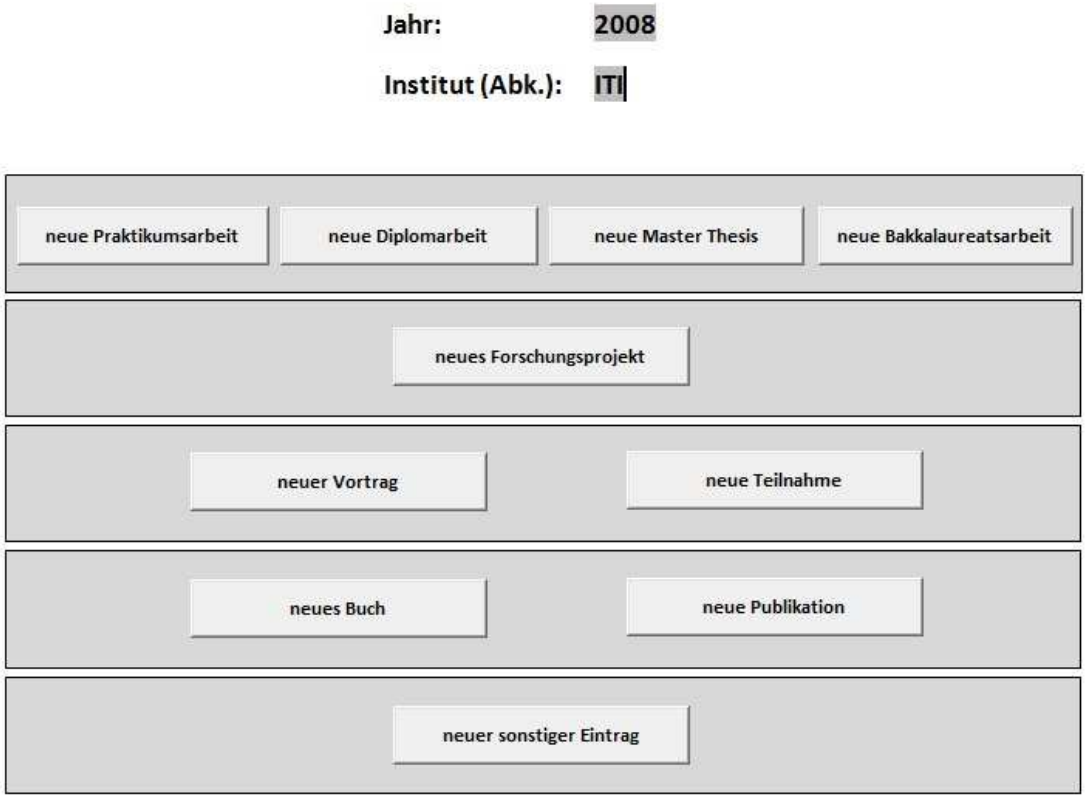


Abbildung 5.5: Word-Vorlage für den Jahresbericht - Hauptansicht

In einigen Formularfeldern wird schon während der Eingabe der Inhalt kontrolliert, um eindeutige Falscheingaben zu vermeiden. Neue Eingaben werden, wie in Abbildung 5.5 zu sehen ist, mit themenbezogenen Schaltflächen durchgeführt. Im Formularmodus, in dem sich das Dokument die ganze Zeit befindet, kann kein Text außerhalb der Eingabefelder hinzugefügt und bearbeitet werden, da nur die Formularelemente aktiviert sind. Beim Betätigen der einzelnen Schaltflächen wird der Formularschutz kurzfristig, für den Nutzer jedoch nicht sichtbar, aufgehoben, um den neuen Eintrag einzufügen. Dadurch wird am Ende des Dokuments eine neue Seite angefügt und das Themengebiet zur Eingabe vorbereitet, wie beispielsweise in Abbildung 5.6 am Beispiel einer Diplomarbeit als *studentische Arbeit* zu sehen ist. Auf dieser Seite wird für jedes Attribut ein Eingabefeld bereit gestellt. Bevor die Vorlage auf der Webmaske hochgeladen wird, muss auf der ersten Seite des Dokuments das Jahr des aktuellen Berichts und das Institut angegeben werden.

Abbildung 5.6: Word-Vorlage am Beispiel einer Diplomarbeit

Anschließend kann der Nutzer die verschiedenen Schaltflächen – für jedes Thema existiert mindestens ein Schaltfläche – betätigen, wodurch ein neuer Eintrag in dem Jahr für das Institut eingefügt wird. Der Nutzer kann für jedes Attribut einen Eintrag vornehmen. Im Beispiel aus der obigen Abbildung sind das *Betreuer*, *Ersteller* oder *Thema* für eine Diplomarbeit. Eine Ausnahme stellt das Themengebiet *Sonstiges* dar. Bei diesem besteht durch Betätigen der Schaltfläche *neuer Unterwert*, wie in Abbildung 5.7 zu sehen ist, die Möglichkeit, mehrere Unterwerte zu einem Eintrag hinzuzufügen. Die Themengebiete können gemischt aneinander gereiht angefügt werden. Somit können z. B. Diplomarbeiten, Veröffentlichungen und andere Themengebiete im Wechsel eingetragen werden, ohne dass der Parser[7] damit Probleme bekommt. Der Parser sucht nach bestimmten Stichwörtern, wie z. B. *Veröffentlichungen* oder *studentische Arbeiten*, und weiß im Anschluss, welches Thema sich auf der Seite befindet. Anschließend wird der dazugehörige Algorithmus für das Auslesen der Daten ausgeführt. Basierend auf der Vorlage ist der Nutzer in der Lage, zu jeder Zeit des Jahres das Dokument mit Einträgen zu füllen. Die entstehenden Word-Dateien werden zum Schluss auf der Webmaske hochgeladen und nach dem im Kapitel 5.2 beschriebenen Algorithmus in die Datenbank gespeichert.

[7] In der Volltextverarbeitung bedeutet „Parsen“, dass Wörter aus einem Text oder Anfrage-String anhand von Regeln extrahiert werden, die definieren, aus welcher Zeichenfolge ein Wort besteht und wo die Wortgrenzen liegen.

Abbildung 5.7: Word-Vorlage am Beispiel des Themengebiets *Sonstiges*

Es ist außerdem möglich, ein Word-Dokument ohne Nutzung der Word-Vorlage abzugeben. Die darin enthaltenen Informationen können teilweise nur semi-strukturiert ausgewertet werden, da die Analyse und Extraktion der Informationen zu aufwendig und komplex ist. Die Informationen werden nur grob abgespeichert, so dass sie im Nachhinein noch bearbeitet werden müssen. Die angesprochene Art der Eingabe von Informationen sollte jedoch zukünftig nicht mehr erfolgen, da für Word die bereits erwähnte Vorlage erstellt wurde. Durch diese wird es dem Nutzer einfacher gemacht, die Informationen einzugeben und anderen Personen wird das Auswerten der nur grob gespeicherten Informationen erspart.

5.1.4 Eingabe über T_EX

Es soll weiterhin möglich sein, über die schon vorhandenen und genutzten T_EX-Masken für die einzelnen Themengebiete Daten für den Jahresbericht abzugeben. Damit können die Mitarbeiter, die gern die T_EX-Masken ausgefüllt haben, auch zukünftig mit ihnen arbeiten. Damit haben die Nutzer die Möglichkeit, die T_EX-Masken am eigenen Arbeitsplatz zu einem selbst gewählten Zeitpunkt auszufüllen. Die T_EX-Dateien werden eingelesen und anschließend analysiert, um sie danach zu speichern. Die Daten müssen, wie in Abbildung 5.8 zu sehen ist, in der richtigen Reihenfolge eingegeben werden und die richtige Anzahl an Attributen aufweisen.

```
\begin{ListeVortraege}
…
\vortrag { } % Initial. Name
         { } % Titel
         { } % Veranstaltung, Ort, Zeit
…
\end{ListeVortraege}
```

Abbildung 5.8: Beispiel einer T_EX-Maske am Beispiel der Vorträge

Die Informationen werden in die geschweiften Klammern an der passenden Position geschrieben. Um die korrekte Funktionsweise des Parsers zu gewährleisten, müssen von den Nutzern die in jeder Maske erläuterten Regeln, wie in Abbildung 5.9 zu sehen ist, eingehalten werden. In dem Fall können die Regeln sich auf den Aufbau des Dokuments oder die Schreibweise bestimmter Inhalte beziehen. Wenn die Informationen an der falschen Position in der Maske eingetragen werden, kann es passieren, dass die Daten in der falschen Spalte einer Tabelle der Datenbank gespeichert werden. Die ausgefüllte Maske wird am Ende auf der Webmaske hochgeladen und anschließend nach dem im Kapitel 5.2 beschriebenen Algorithmus in der Datenbank gespeichert.

```
%% Vortr"age und Veranstaltungen.
%%
%% Unter der ersten Liste Vortr"age sind alle Vortr"age mit Angabe
%% der jeweiligen Veranstaltung anzugeben.
%%
%% In der zweiten Liste Teilnahmen sind nur die im Unterkapitel
%% Vortr"age noch nicht genannten Veranstaltungsteilnahmen
%% aufzuf"uhren!
…
%% Gro"s- und Kleinschreibung von englischen Titeln wie im Kapitel
%% Ver"offentlichungen.
%% Bei Teilnahmen mehrerer Personen bitte nur einen Eintrag,
%% Personen nur mit abgek"urzten Vornamen bezeichnen.
%% Datumsangaben mit ausgeschriebenem Monat.
%% Bitte die Eintr"age innerhalb der Listen alphabetisch ordnen.
```

Abbildung 5.9: Regeln und Erläuterungen einer T_EX-Maske (*Vorträge und Teilnahmen*)

5.2 Ablaufbeschreibung des Programms

In der Ablaufbeschreibung, die in diesem Abschnitt beschrieben wird, werden die Punkte des Entwurfs des Softwarelebenszyklus behandelt. Es werden „die in der Spezifikation erstellten Modelle“ [Dumke 2003, S. 69] in Diagrammen, Schemata oder Pseudocodes umgesetzt, die die qualitativen Anforderungen einschließen. [Dumke 2003, S. 69] Für die Dateneingabe unter Verwendung von Word und T_EX wird die Verarbeitung in vier Phasen, die in Abbildung 5.10 dargestellt sind, aufgeteilt. Die Webmaske spielt in diesem Zusammenhang keine Rolle, da sie die Daten direkt über

die Maske erhält und dementsprechend keine fremden Dokumente ausgelesen werden müssen. Die TEX- und Word-Dokumente werden lediglich an dieser Stelle hochgeladen, die Verarbeitung der Dokumente wird jedoch unabhängig von der Maske durchgeführt.

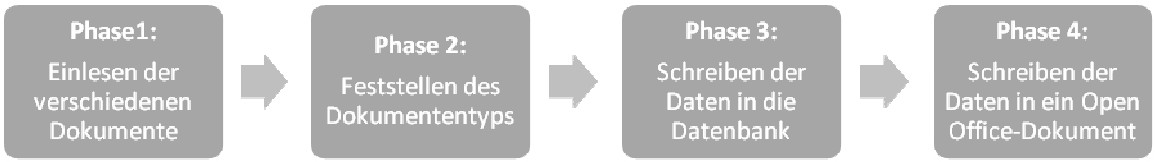


Abbildung 5.10: Ablauf des neuen Systems

Phase 1 – Einlesen der verschiedenen Dokumente

In der ersten Phase werden die verschiedenen TEX- und Worddokumente eingelesen. Während es bei der Benutzung der Vorlage für Word nur ein Dokument für alle Themengebiete gibt, existiert bei TEX je Themengebiet ein Dokument. In dieser Phase werden die Kommentare in den TEX-Dokumenten gelöscht, damit keine Informationen versehentlich aus dem kommentierten Bereich in den Jahresbericht gelangen. Der größte Teil der Kommentare besteht aus den Aufbauregeln (siehe Abbildung 5.9) zu Beginn der Dokumente. Sonderzeichen, wie z. B. „ö“, „ä“ oder „ß“, werden zum Teil mit TEX-Syntax gekennzeichnet. Dort wird ein „ö“ beispielsweise durch die Zeichenfolge „“ö“ oder „{\“ö}“ dargestellt. Diese werden in den TEX-Dokumenten in die gewohnten deutschen Sonderzeichen konvertiert. Bei der Word-Vorlage sind diese Schritte nicht notwendig. Der Nutzer hat im Gegensatz zu den TEX-Dokumenten nicht die Möglichkeit, in dem Dokument Kommentare anzulegen. Außerdem ist es ihm nicht möglich, an nicht gewünschten Stellen zu schreiben, da die Vorlage im Formularmodus und somit geschützt ist. Eine Konvertierung der Sonderzeichen ist nicht notwendig aufgrund des WYSIWYG-Prinzips („What you see is what you get“). Es existiert kein Code in der Vorlage. Der Text wird in der Form eingegeben, wie er in die Datenbank und in den Jahresbericht soll.

```
\begin{ListeTeilnahmen}
…
\teilnahme
{}  % Initial. Name, gegebenenfalls weitere Namen
{}  % Veranstaltung, Ort, Zeit
…
\end{ListeTeilnahmen}
```

Abbildung 5.11: Muster einer TEX-Maske am Beispiel der *Teilnahme an Veranstaltungen*

Phase 2 – Feststellen des Dokumententyps

Während der zweiten Phase wird der Typ und damit das Themengebiet des Dokuments festgestellt. Bei einem Dokument basierend auf der Word-Vorlage ist dies nicht notwendig, da alle Themengebiete in diesem Dokument behandelt werden. In diesem Fall wird das Dokument ohne Vorverarbeitung eingelesen.

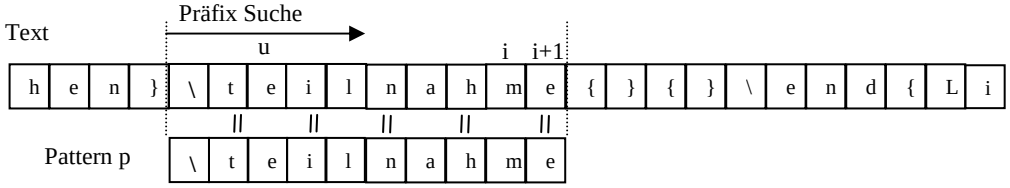


Abbildung 5.12: Pattern Matching für Schlüsselwortsuche

Bei den TEX-Dokumenten wird zuerst der Typ des Dokuments festgestellt. Dies erfolgt mittels Suche nach bestimmten Schlüsselwörtern, die für die jeweiligen Themengebiete typisch bzw. eindeutig sind.

Tabelle 5.3: Schlüsselwörter für die Themengebiete in TEX-Dokumenten

Studentische Arbeiten	\arbeit	\work	
	\begin{Praktikumsarbeiten}	\begin{Diplomarbeiten}	
	\begin{Master´s Theses}	\begin{Bakkalaureatsarbeiten}	
Forschungsgebiete und –projekte	\projekt	\project	
Veröffentlichungen	@article	@unpublished	@techreport
	@inproceedings	@incollection	@inbook
	@proceedings	@phdthesis	@book
Vorträge und Teilnahme an Veranstaltungen	\vortrag	/begin{ListeVortraege}	
	\teilnahme	/begin{ListeTeilnahmen}	
Sonstiges	\subsection{Eigene Veranstaltungen}	\subsection{Herausgeberschaften von Periodika, Editortätigkeiten}	
	\subsection{Gastaufenthalte von Mitgliedern des Instituts}	\subsection{Mitarbeit in Programmkomitees}	
	\subsection{Mitgliedschaften}	\subsection{Lehraufträge an anderen Einrichtungen}	
	\subsection{Gremientätigkeiten}	\subsection{Was sonst noch wichtig war}	
	\subsection{Gutachtertätigkeiten}		

Ist beispielsweise die Zeichenfolge „\teilnahme“, wie in Abbildung 5.11 zu sehen ist, in einem Dokument vorhanden, so ist die Wahrscheinlichkeit sehr hoch, dass es sich um ein Dokument für *Vorträge und Teilnahme an Veranstaltungen* handelt. Die Suche wird, wie in Abbildung 5.12 zu sehen ist, durch den Präfix-basierten Ansatz des

Pattern Matching in Kapitel 3.1.3 durchgeführt. In dem Text *u* (z. B. *vortrag.tex*) wird nach einem bestimmten Pattern *p* („\teilnahme“) gesucht. In Tabelle 5.3 wird eine Übersicht über die Schlüsselwörter für die verschiedenen Themengebiete gegeben. Dabei werden Zeichenfolgen, die aus einer Mischung von TEX-Syntax und Jahresberichtsvokabular, verwendet. Nachdem der Typ des Dokuments festgestellt wurde, wird das Dokument entsprechend dem Typ validiert. Analog wie bei der Typermittlung, wird noch einmal nach den Schlüsselwörtern durch das *Pattern Matching* gesucht. Bei einem Treffer wird kontrolliert, ob ausreichend Attribute, in den TEX-Masken durch geschweifte Klammern dargestellt, zu dem Schlüsselwort existieren. Es existiert zu jedem Schlüsselwort ein Pattern, das mit Inhalt gefüllt werden muss. In Abbildung 5.13 ist ein solches gefülltes Pattern am Beispiel der *studentischen Arbeiten* dargestellt, bei dem dem Schlüsselwort „\arbeit“ genau drei Attribute (*Ersteller*, *Betreuer* und *Thema*) folgen müssen.

```
\begin{Diplomarbeiten}
…
\arbeit{Mathias Kant}
       {Hans-Knud Arndt}
       {Aufbau und Realisierung einer Software
         zur Konvertierung von LATEX-Dokumenten,
         in einem XML-Standard}
…
\end{Diplomarbeiten}
```

Abbildung 5.13: Pattern der studentischen Arbeiten

Zusätzlich werden alle geschweiften Klammern untersucht. Die Validation ist nur dann erfolgreich, wenn genügend Attribute zu den Schlüsselwörtern vorhanden sind und alle geschweiften Klammern wieder geschlossen werden. Ist die Validation nicht erfolgreich, wird dem Nutzer der Grund ausgegeben. Um ein Dokument erfolgreich validieren und anschließend einlesen zu lassen, muss der Nutzer gegebenenfalls das Dokument entsprechend der Fehlermeldung anpassen und erneut hochladen.

Phase 3 – Schreiben der Daten in die Datenbank

Nachdem der Typ des TEX-Dokuments in der zweiten Phase festgestellt wurde, können die entsprechenden Informationen ausgelesen und anschließend in die Datenbank geschrieben werden. Die dritte Phase ist die bedeutendste von allen, da hier die Informationen aus dem Dokument extrahiert werden. Dafür wird das Muster des vorher validierten Themengebiets auf das Dokument angewendet. Das heißt die Informationen werden dem Muster entsprechend aus dem TEX-Dokument ausgelesen und in die Datenbank geschrieben. In dem Dokument wird nach Schlüsselwörtern des Themengebiets gesucht, beispielsweise „\arbeit“ bei den *studentischen Arbeiten*. Wie in

Phase 2 wird der Präfix-basierte Ansatz des *Pattern Matching* aus Kapitel 3.1.3 angewendet. In Abbildung 5.14 wird dies am Beispiel der *studentischen Arbeiten* dargestellt. Findet der *Pattern Matching*-Algorithmus ein Vorkommen des Patterns wird eine weitere Suche gestartet. Bei den *studentischen Arbeiten* wird dann beispielsweise nach den drei Einträgen *Ersteller*, *Betreuer* und *Thema* gemäß der Reihenfolge, in der diese in den geschweiften Klammern aufgelistet sein müssen, gesucht. Typische Rechtschreibfehler wie beispielsweise „teilname“ oder englische Begriffe wie z. B. „work“ werden berücksichtigt und nicht aufgrund fehlender Übereinstimmung mit dem Pattern übersprungen. In dem eben erwähnten Beispiel würde der Parser die Zeichenfolge „teilname“ dementsprechend als „teilnahme“ interpretieren und die dazugehörigen Attribute einlesen. Anschließend werden die Attribute eingelesen und in eine Schablone, ein sogenanntes Prepared Statement, geschrieben. Unter einem Prepared Statement wird eine vorbereitete Anweisung für ein Datenbanksystem ohne Parameterwerte verstanden. Mit diesen Statements kann eine Anweisung schneller und mit unterschiedlichen Parameterwerte mehrfach ausgeführt werden. Ein Beispiel dafür ist `SELECT id FROM personen WHERE (vorname = ?)`. Um am Ende in der Datenbank keine TEX-Notationen zu haben, werden vor der Übertragung in die Datenbank alle noch vorhandenen geschweiften Klammern, Backslashes und sonstige TEX-Ausdrücke aus dem jeweiligen Eintrag entfernt. Um unabsichtliche Falscheingaben zu vermeiden, wird bei bestimmten Einträgen die Länge kontrolliert. Zu einer solchen fehlerhaften Eingabe kann es kommen, wenn der Anwender z. B. anstatt eines Namens einen längeren Titel eingibt, der wiederum zu lang für die Spalte des Namens in der Datenbank ist. Um Redundanz in der Datenbank zu vermeiden, wird beim Speichern in der Datenbank noch geprüft, ob der Eintrag schon vorhanden ist. Wenn ein Eintrag bereits eingetragen wurde, wird der Speichervorgang in die Datenbank unterbrochen und zum nächsten Eintrag im Dokument gesprungen.

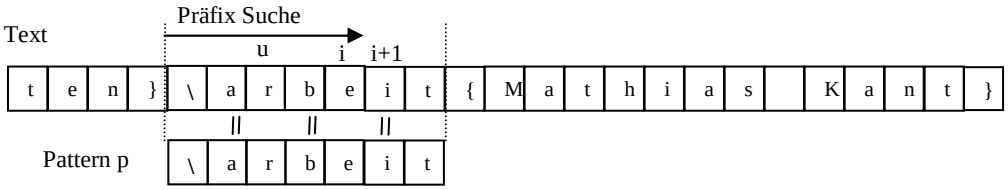


Abbildung 5.14: *Pattern Matching* bei den TEX-Dokumenten am Beispiel der *studentischen Arbeiten*

Liegt ein Dokument auf der Basis der erstellten Word-Vorlage vor, werden zu Beginn das Institut und das Jahr, die auf der ersten Seite vom Nutzer eingegeben wurden, ausgelesen. Anschließend wird das Dokument nach Schlüsselwörtern, die die Themengebiete des Jahresberichts repräsentieren, durchsucht. Wird ein Schlüsselwort gefunden, wird ähnlich dem Verfahren bei den TEX-Dokumenten, ein Muster

angewandt. Das heißt, die Attribute des jeweiligen Themengebiets werden nach einem Pattern ausgelesen und in die Datenbank gespeichert.

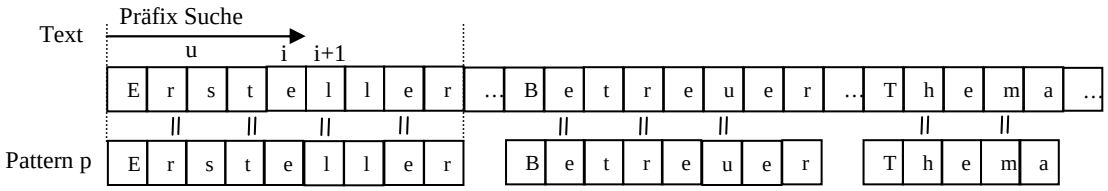


Abbildung 5.15: *Pattern Matching* bei der Word-Vorlage am Beispiel der *studentischen Arbeiten*

Findet der Algorithmus beispielsweise das Schlüsselwort *studentische Arbeiten* sucht er anschließend, wie in Abbildung 5.15 zu sehen ist, nach den Einträgen *Ersteller*, *Betreuer* und *Thema*. Zur Speicherung in der Datenbank werden die Zeichenfolgen, die den eben erwähnten Einträgen folgen, bis zum Ende der Zeile verwendet. Bevor der komplette Eintrag in die Datenbank gespeichert wird, wird kontrolliert, ob er bereits existiert. Dieser Algorithmus wird im gesamten Dokument der Vorlage für jedes gefundene Schlüsselwort bzw. Themengebiet durchgeführt.

Bei den semi-strukturierten Word-Dokumenten werden die Daten aufgrund des unstrukturierten Aufbaus teilweise nur grob abgespeichert. Dies bedeutet, dass viele Daten nicht den richtigen Spalten in den Tabellen der Datenbank zugeordnet werden können. Da bei diesen Dokumenten verschiedene Begriffe für ein Themengebiet verwendet werden können, wurde eine Liste mit Begriffen, die in Tabelle 5.4 dargestellt sind, erstellt. Die Begriffe werden den entsprechenden Themengebieten zugeordnet und bei jedem einzelnen festgelegt, ob es ein Haupt- oder Unterbegriff des Themengebiets ist. Unter einem Hauptbegriff sind die Themengebiete des Jahresberichts zu verstehen. Unterbegriffe hingegen sind Begriffe zu den Themengebieten bzw. den Hauptbegriffen. Demzufolge wäre beispielsweise *Praktikumsarbeiten* ein Unterbegriff des Hauptbegriffs *studentische Arbeiten*. Somit können viele Informationen anhand dieser Begriffe gefunden und, soweit möglich, in den richtigen Spalten der Tabellen der Datenbank gespeichert werden. Jedoch werden viele Informationen nur in der Spalte „undefiniert“ für unstrukturierte Daten gespeichert. Um sie in den Jahresbericht einfließen zu lassen, müssen diese im Nachhinein manuell durch Mitarbeiter der FIN aufbereitet werden.

Tabelle 5.4: Beispiele der Zuordnung der Begriffe zu den Themengebieten

Begriff	Themengebiet	Typ
arbeiten	Studentische Arbeiten	HB
forschung	Forschungsgebiete und –projekte	HB
buch	Veröffentlichungen	HB
publikation	Veröffentlichungen	HB
vortrag	Vorträge und Teilnahmen	HB
teilnahme	Vorträge und Teilnahmen	HB
sonstiges	Sonstiges	HB
monographie	Sonstiges	UB
mitgliedschaften in programmkomitees	Sonstiges	UB
lehraufträge an anderen einrichtungen	Sonstiges	UB
mitgliedschaften	Sonstiges	UB
gäste der arbeitsgruppe	Sonstiges	UB
praktikumsarbeit	Studentische Arbeiten	UB
diplomarbeit	Studentische Arbeiten	UB
master theses	Studentische Arbeiten	UB
referierte publikation	Veröffentlichungen	UB
mitherausgeberschaften	Veröffentlichungen	UB

HB – Hauptbegriff, UB - Unterbegriff

Phase 4 – Schreiben der Daten in ein Open Office-Dokument

Nachdem alle Dokumente die ersten drei Phasen erfolgreich durchlaufen haben, befinden sich die erforderlichen Daten in der Datenbank. Zu diesem Zeitpunkt sind die Daten aus den verschiedenen Quellen in der Datenbank gespeichert. Aus dieser Datenbank werden nun die Daten ausgelesen und in ein Open Office-Dokument geschrieben, wobei der *ISO/IEC-Standard 26300* zur Anwendung kommt. In der *content.xml*, die in Abbildung 5.16 in einem Auszug dargestellt ist, sind der Inhalt und die Formatierung des Jahresberichts enthalten.

Tabelle 5.5: Formatvorlagen des Open Office-Dokuments

Name	Schriftgröße	fett	kursiv	Seitenumbruch	Item
thema	16	X		X	
institut	14	X			
überschrift	12	X			
text	8				
kursiv	8		X		
item	8				X

Die Einträge in das Open Office-Dokument werden durch die implementierte Software so formatiert, dass sie eindeutig – ähnlich dem originalen Jahresbericht – erkennbar

sind. Für die Bezeichnung der Themengebiete wurde beispielsweise die Formatvorlage *thema* generiert. Sie beinhaltet die Schriftgröße (16), ist fettgedruckt und fügt einen Seitenumbruch vor dem Eintrag ein. Diese und andere Formatvorlagen für Untergebiete und Attribute werden in Tabelle 5.5 dargestellt.

```
<text:p text:style-name="thema">Studentische Arbeiten</text:p>
<text:p text:style-name="institut">ITI</text:p>
<text:p text:style-name="überschrift">Diplomarbeiten</text:p>
<text:p text:style-name="kursiv">Name (Betreuer/in)</text:p>
<text:p text:style-name="text">Mathias Kant (Hans-Knud
      Arndt)</text:p>
<text:p text:style-name="kursiv">Thema</text:p>
<text:p text:style-name="text"> Aufbau und Realisierung einer
      Software zur Konvertierung von LATEX-Dokumenten, in einem XML-
      Standard </text:p>
<text:p text:style-name="text" />
```

Abbildung 5.16: Auszug aus der *content.xml* für den Jahresbericht

5.3 Vor- und Nachteile des neuen Systems

Das zuvor vorgestellte System wurde umgesetzt, um die Erstellung des Jahresberichts effektiver, d. h. schneller und weniger fehlerbelastet, zu gestalten. Ein weiteres Ziel des neuen Systems war es, den Ansprüchen mehrerer Nutzergruppen gerecht zu werden. Im Folgenden werden die Vorteile des neuen Systems dargestellt, aber auch kritisch auf mögliche Nachteile geschaut.

5.3.1 Vorteile

Mehrere potentielle Nutzergruppen werden angesprochen

Ein großer Vorteil des neuen Systems besteht darin, dass fast alle potentiell mit dem Jahresbericht befassten Gruppen angesprochen werden. Im alten Erstellungsprozess bestand nur die Möglichkeit, die T_EX-Masken auszufüllen. Word wurde nur als Ausnahme anerkannt, da sich die semi-strukturierten Dokumente nicht in die T_EX-Dokumente integrieren ließen, sondern manuell übertragen werden mussten. Mit dem neuen System und der daraus resultierenden Word-Vorlage haben potentiell mehr Nutzer die Möglichkeit, mit dem präferierten Programm Daten für den Jahresbericht einzugeben. Die T_EX-Nutzer können weiterhin die bekannten Masken ausfüllen. Für die Nutzer, die keines der beiden Programme nutzen wollen, besteht die Möglichkeit, die Daten in der Webmaske einzugeben. Damit stehen drei Möglichkeiten zur Verfügung, um Daten in den Jahresbericht einzutragen. Durch die Nutzung des präferierten

Programms kann die Motivation der Mitarbeiter, Daten in den Jahresbericht einzugeben, steigen.

Geringerer Zeitaufwand

Ziel ist es, in der Zukunft alles über die Webmaske zu verwalten. Das bedeutet, dass alle Dokumente dort hochgeladen werden, zusätzlich zu der Möglichkeit in der Webmaske selbst die Daten direkt einzutragen. Dadurch dass jede Arbeitsgruppe bzw. jeder Mitarbeiter selbst die Möglichkeit hat, die Daten in den Jahresbericht über die Webmaske einzupflegen, fällt der Prozess der Weitergabe der Informationen über die Schnittstellen vom Arbeitsgruppenverantwortlichen über den Institutsverantwortlichen bis hin zum Jahresberichtsverantwortlichen weg und es wird eine Menge Zeit eingespart. Natürlich müssen die Daten am Ende noch einmal auf Rechtschreibfehler und Semantik bzw. Inhalt kontrolliert werden. Der Zeitaufwand hat sich trotz der abschließenden Prüfung verringert.

5.3.2 Nachteile

Ein Nachteil des Systems ist das Fehlerpotential bei der Interpretation der Dokumente, d. h. beim Lesen und Speichern in der Datenbank. Es könnten Daten aufgrund von fehlerhaften Eingaben des Nutzers und durch falsche Interpretation der Daten durch die implementierte Software in die falsche Spalte der Tabelle in der Datenbank eingetragen werden. Durch die unterschiedlichen Dokumentenformate kann es zu Differenzen im Format der Eingaben kommen. Beispielsweise existiert bei den TEX-Masken *Vorträge und Teilnahme an Veranstaltungen* für die Veranstaltung, den Ort und die Zeit nur ein gemeinsames Attribut. In der Word-Vorlage und der Webmaske sind für dieses Beispiel drei einzelne Attribute gegeben. In der Datenbank würden das Attribut aus der TEX-Maske in einer Spalte und die Attribute aus den anderen beiden Formaten in drei verschiedene Spalten der Tabelle in der Datenbank gespeichert.

5.4 Vergleich des alten mit dem neuen System

Der Vergleich des alten Systems mit dem neuen System zeigt, dass das neue System dem Nutzer wesentlich mehr Möglichkeiten der Dateneingabe bietet. In Tabelle 5.6 wird ein kurzer Vergleich der beiden Systeme dargestellt.

Tabelle 5.6: Vergleich altem gegenüber neuem System

Thema	Altes System	Neues System
Eingabemöglichkeiten	T_EX-Masken	T_EX-Masken
		Word-Vorlage
	Word (semi-strukturiert)	Word (semi-strukturiert)
		Webmaske
Einarbeitungszeit	Hoch	Niedrig
Zeit zur Bearbeitung	Hoch	Mittel
Zeit der Eingabe	Zum Zeitpunkt der Erstellung des Jahresberichts	Zum Zeitpunkt des Ereignisses möglich

Im Vergleich zum alten System benötigt der Nutzer weniger Einarbeitungszeit, da er sich die Methode, Informationen in den Jahresbericht einzugeben, aussuchen kann und sich somit meistens schon mit dem Programm seiner Wahl auskennt. Dadurch kann er schneller die Informationen in die Masken oder Vorlage eingeben. Durch die Webmaske und der Möglichkeit jederzeit die T_EX- und Word-Dokumente dort hochzuladen, können die Nutzer die Informationen zum Zeitpunkt des Ereignisses eingeben, so dass sie nicht zum Zeitpunkt der Erstellung des Jahresberichts Informationen des letzten Jahres auf einmal zusammen tragen müssen. Das neue System bietet weit mehr Flexibilität, da es noch durch andere Formate wie z. B. eine Open Office-Vorlage ergänzt werden könnte. Das alte System ist ziemlich starr auf die T_EX-Masken festgelegt und lässt keine Alternativen außer semi-strukturierten Word-Dokumenten zu.

6 Zusammenfassung und Ausblick

Die Aufgabe dieser Diplomarbeit war es, die Erstellung des Jahresberichts zu automatisieren, zu vereinfachen bzw. durch mehrere Quellen erstellbar zu machen. Das resultierende Dokument sollte auf einem ISO-Standard basieren. Nach einer Untersuchung möglicher Textanalyse-Methoden (Kapitel 3.1), dem Vergleich zweier ISO-Standards (Kapitel 3.2) und der Einführung des Anwendungsbeispiels (Kapitel 4) – Jahresbericht der FIN – entstand das neue System (Kapitel 5), bei dem die schon existierenden T_EX-Masken weiterhin verwendet werden. Zusätzlich wurde es ermöglicht, Word-Dokumente zu verwenden. Dafür wurde eine Word-Vorlage erstellt, mit der alle Themengebiete in einem Dokument für das jeweilige Jahr zusammengestellt werden können. Weiterhin ist es möglich, semi-strukturiert Word-Dokumente einzulesen. Die T_EX- und Word-Dokumente werden mit einem entsprechenden Algorithmus in eine Datenbank eingelesen. Außerhalb dieser Diplomarbeit wurde eine Webmaske in einem Seminar entwickelt, mit der die Daten zum Zeitpunkt des Ereignisses über das gesamte Jahr verteilt eingegeben werden können. Damit existieren genau drei Eingabetypen, mit denen Daten für den Jahresbericht eingegeben werden können. Anschließend wurden die Daten der Datenbank in ein Open Office-Dokument, das auf dem *ISO/IEC Standard 26300* basiert, geschrieben. Dieses Dokument beinhaltet alle Informationen, gruppiert nach den Themengebieten, zu dem Jahresbericht mit einer ähnlichen Formatierung wie beim bisherigen Bericht.

Für die Erstellung des Jahresberichts in der Zukunft kann zur Unterstützung das neue System angewendet werden. Durch die verschiedenen Quellmedien und die automatisierte Speicherung der Daten wurde der Prozess der Erstellung für den Nutzer vereinfacht.

In der Zukunft sollten die semi-strukturierten und damit schwer interpretierbaren Word-Dokumente nicht verwendet werden. Als effektiver und intuitiver Ersatz wurde die besser einzulesende Word-Vorlage entwickelt. Zukünftig können weitere Programme zur Eingabe von Informationen in den Jahresbericht dem System hinzugefügt werden. Beispielsweise könnte eine Vorlage für Open Office erstellt werden, wobei für das Auslesen der Code für die Word-Vorlage verwendet und bei Bedarf angepasst werden kann.

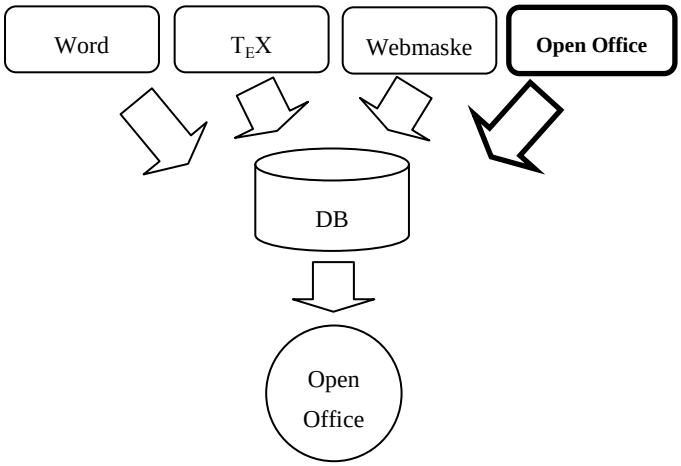


Abbildung 6.1: Beispiel eines neuen möglichen Systems

Durch den übersichtlichen und gut verständlichen Aufbau der Software und ihres Quellcodes sind Anpassungen, die sich aus der Weiterentwicklung des Berichts ergeben können, leicht einzupflegen. Um Verbesserungspotentiale zu erkennen und damit eine kontinuierliche Verbesserung zu ermöglichen, müssen die Nutzer Input über das bestehende System geben. Mit der entwickelten Software existiert eine weniger zeitintensive, erweiterbare und effektivere Möglichkeit, Informationen in den Jahresbericht einzugeben.

A Anhang

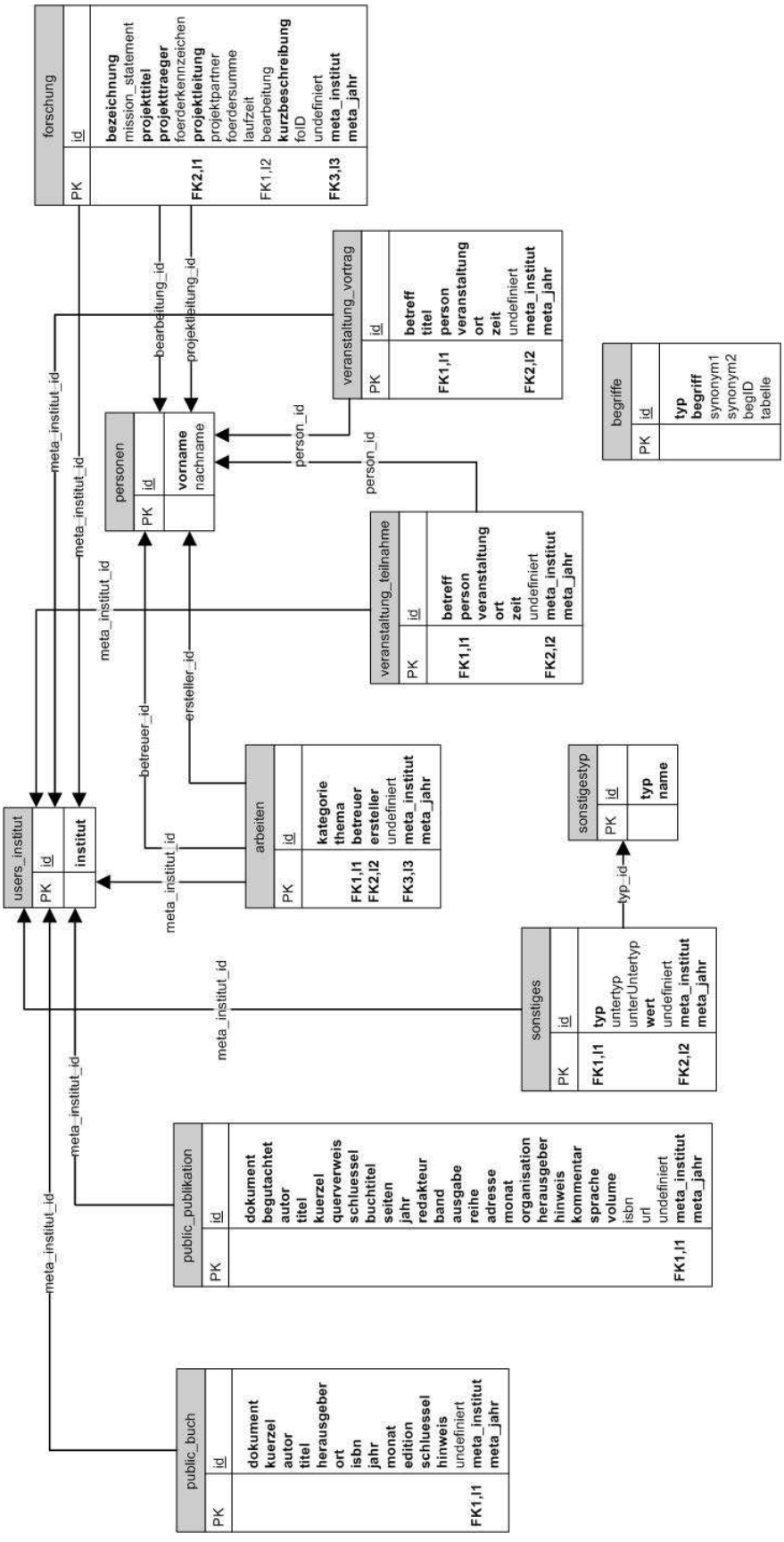


Abbildung A.1: Datenbankschema der Software

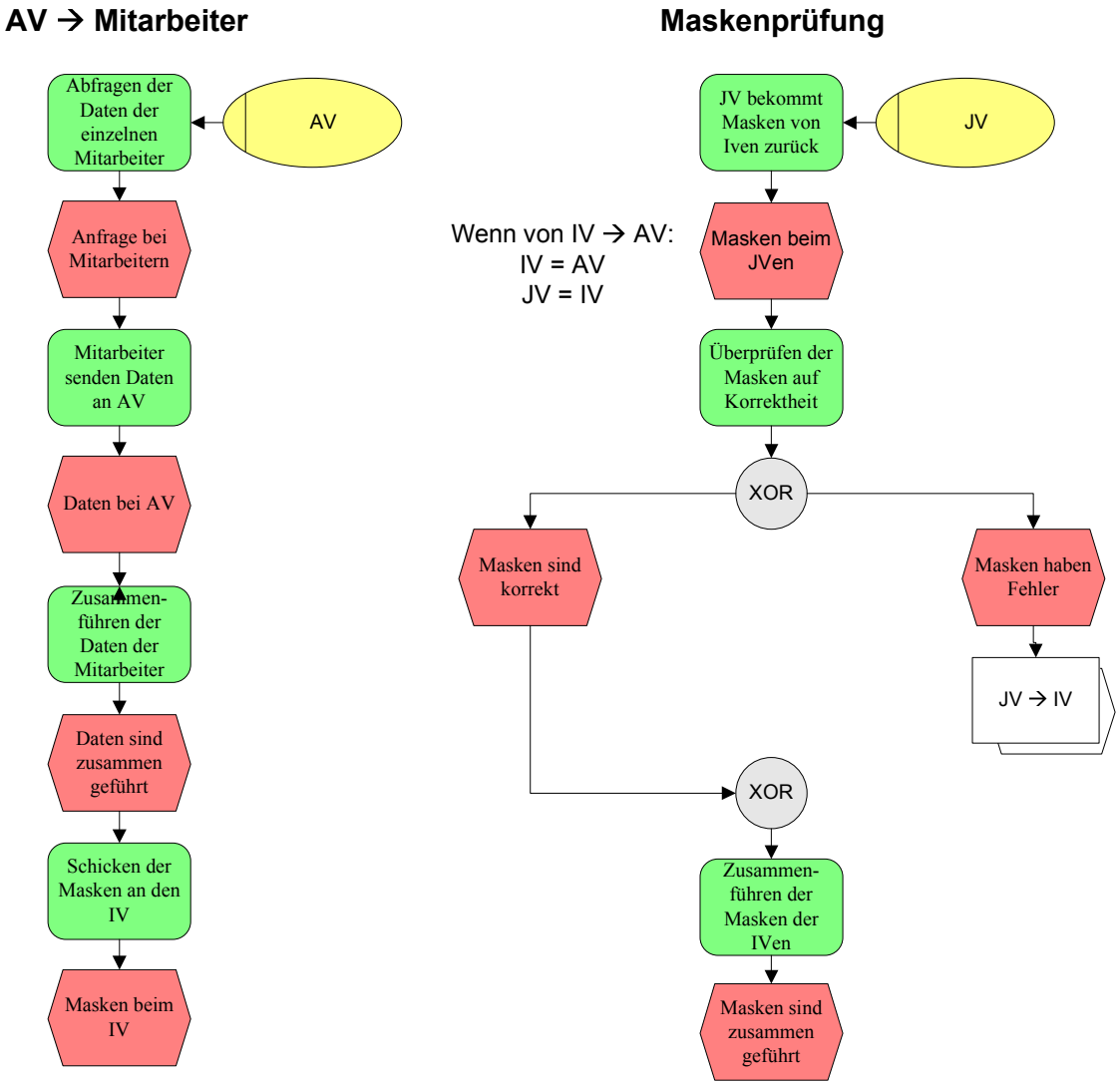


Abbildung A.2: AV → Mitarbeiter und Maskenprüfung (JV – Jahresberichtverantwortlicher, IV – Institutsverantwortlicher, AV – Arbeitsgruppenverantwortlicher)

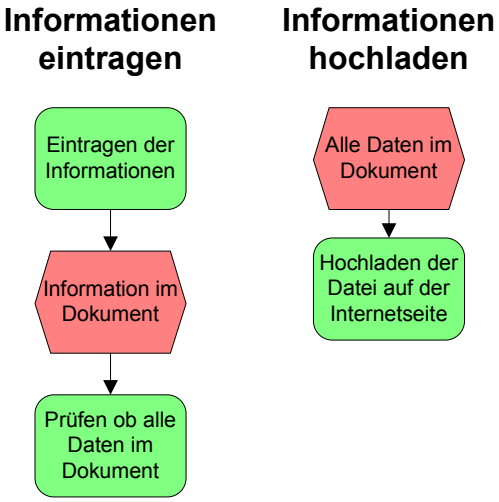


Abbildung A.3: Teilprozesse: *Informationen eintragen*, *Informationen hochladen* aus dem Ablauf des neuen Systems

Literaturverzeichnis

Monographien

Apostolico, A.; Chrochemore, M.; Galil, Z.; Manber, U. (1992): Combinatorial Pattern Matching, Berlin Heidelberg

Apostolico, A; Galil, Z. (1997): Pattern Matching Algorithms, New York Oxford

Baeza-Yates, R.; Ribeiro-Neto, B. (1999): Modern Information Retrieval, Harlow

Bauer, A.; Günzel, H. (2004): Data Warehouse Systeme Architektur, Entwicklung, Anwendung, 2., überarbeitete und aktualisierte Auflage, Heidelberg

Crochemore, M.; Rytter, W. (2003): Jewels of Stringology Text Algorithms, New Jersey et al.

Dengel, A.; Junker, M.; Weisbecker, A. (2004): Reading and Learning Adaptive Content Recognition, Berlin Heidelberg

Dumke, R. (2003): Software Engineering, 4., überarbeitete und erweiterte Auflage, Wiesbaden

Ferber, R. (2003): Information Retrieval Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web, Heidelberg

Frakes; Baeza-Yates (1992): Information Retrieval Data Structures & Algorithms, New Jersey

Frank, E; Witten, I. H. (2001): Data Mining Praktische Werkzeuge und Techniken für das maschinelle Lernen, München u. a.

Grossmann, D. A.; Frieder, O. (2004): Information Retrieval Algorithms and Heuristics, Dordrecht

Han, J.; Kamber, M. (2006): Data Mining Concepts and Techniques, Amsterdam et al.

Hand, D; Mannila, H.; Smyth, P. (2001): Principles of Data Mining, Cambridge Massachusetts, London

Hengartner, U. (1997): Entwurf eines integrierten Informations-, Verwaltungs- und Retrieval Systems für textuelle Daten, Bern

Heuer, A.; Saake, G. (2000): Datenbanken: Konzepte und Sprachen, 2., aktualisierte und erweiterte Auflage, Landsberg

Jones, K. S.; Willett, P. (1997): Readings in Information Retrieval, San Francisco

Moens, M.-F. (2006): Information Extraction: Algorithms and Prospects in a Retrieval Context, Dordrecht

Navarro, G.; Raffinot, M. (2004): Flexible Pattern Matching in Strings, Cambridge

Oberhauser, O. (2005): Automatisches Klassifizieren Entwicklungsstand – Methodik – Anwendungsbereiche, Frankfurt am Main

Petersohn, H. (2005): Data Mining Verfahren, Prozesse, Anwendungsarchitektur, München

Tan, P.-N.; Steinbach, M.; Kumar, V. (2006): Introduction to Data Mining, Boston et al.

Internet-Adressen

Beld, J. (2006): Ecma International approves Office Open XML standard. http://www.ecma-international.org/news/PressReleases/PR_TC45_Dec2006.htm. 16. Dezember 2008.

Durusau, P.; Brauer, M.; Oppermann, L. (2007): Open Document for Applications (OpenDocument) v1.1. http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf. 16. Dezember 2008.

Eikvil, L. (1999): Information Extraction from World Wide Web – A Survey –. http://user.phil-fak.uni-duesseldorf.de/~rumpf/SS2005/Informationsextraktion/Pub/Eik99.pdf. 16. Dezember 2008.

Frost, R.; Brougham, D. (2007): Vote closes on draft ISO/IEC DIS 29500 standard. http://www.iso.org/iso/newsandmedia/pressrelease.htm?refid=Ref1070. 16. Dezember 2008.

Frost, R. (2008): Four national standards bodies appeal against approval of ISO/IEC DIS 29500. http://www.iso.org/iso/pressrelease.htm?refid=Ref1136. 16. Dezember 2008.

Frost, R.; Buck, J. (2008): ISO and IEC members give go ahead on ISO/IEC DIS 29500. http://www.iso.org/iso/pressrelease.htm?refid=Ref1151. 16. Dezember 2008.

Frost, R.; Buck, J. (2008): Publication of ISO/IEC 29500:2008, Information technology - Document description and processing languages - Office Open XML file formats. http://www.iso.org/iso/pressrelease.htm?refid=Ref1181. 16. Dezember 2008.

Krempl, S. (2008): NATO unterstützt offenes Dokumentenformat ODF. http://www.heise.de/newsticker/NATO-unterstuetzt-offenes-Dokumentenformat-ODF--/meldung/113000. 10. Januar 2009.

Macnaghten, E. (2007): Technical Distinctions of ODF and OOXML. http://www.freesoftwaremagazine.com/node/2138/pdf. 16. Dezember 2008.

Ngo, T. (2008): Office Open XML Overview. http://www.ecma-international.org/news/TC45_current_work/OpenXML%20White%20Paper.pdf. 16. Dezember 2008.

o. V. (2007): Objections to JTC-1 Fast-Track Processing of the Ecma 376 Specification v. 0.1. http://www.grokdoc.net/index.php/EOOXML_objections. 16. Dezember 2008.

Partosch, G. (1997): „Gleichzeitiges" Publizieren in HTML und LaTeX. http://www.uni-giessen.de/partosch/TeX/converters.html. 16. Dezember 2008.

Pattison, T; Predeek, C.; van Vugt, W. (2007): Using Office Open XML Formats to Support Electronic Health Records Portability and Health Industry Standards. http://msdn.microsoft.com/de-de/bb879915(en-us).aspx. 16. Dezember 2008.

Zendel, O. (2006): Erstellen und Verarbeiten von Office-Dokumenten mit verschiedenen Programmen. http://www.pro-linux.de/berichte/jpgs/odf-artikel/abbildung3.png. 16. Dezember 2008.

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, den 19. März 2009