Structure Mining für die Wismarer Grundbücher des 16. – 19. Jahrhunderts

Diplomarbeit

Universität Rostock, Institut für Informatik
Lehrstuhl Datenbanken und Informationssysteme

vorgelegt von:	**Manja Nelius**
geboren am:	09.11.1979 in Altenburg
Matrikel-Nr.:	098201630
Betreuender Hochschullehrer:	Prof. Dr. Andreas Heuer
Zweitgutachter:	Prof. Dr. Peter Forbrig
Betreuerin:	Dr. Meike Klettke
Abgabedatum:	29. Oktober 2004

Zusammenfassung

Im Rahmen dieser Arbeit ist ein flexibles Verfahren entstanden, das aus den Wismarer Grundbüchern so viele Informationen wie möglich extrahiert und diese in einer Datenbank ablegt. Dabei wurde ein iteratives Vorgehen realisiert, welches die gewonnenen Daten aus einem Teil der Ausgangsdokumente zur Auswertung des Inhaltes anderer Dateien nutzt.

Den Schwerpunkt des Verfahrens stellt die sukzessive Analyse der Dokumentinhalte dar. Dafür werden die Eingabedokumente zunächst in besser auswertbare Formate transformiert, wonach die Interpretation von Layout- und Strukturinformationen erfolgt. Nach der Normalisierung der Dokumente werden Parser zur Analyse der strukturierten Dokumentanteile genutzt. Für die Auswertung der Volltextanteile kommen hingegen Wörterbücher, buchstaben- und sprachorientierte Verfahren zum Einsatz. Nutzerdefinierbare semantische Regeln werden zur Identifizierung mehrdeutig und nicht erkannter Inhalte angewandt. Zum Abschluss erfolgt die Strukturierung von logisch zusammengehörigen Informationen, wonach die analysierten Inhalte auf relationale Datenbanktabellen abgebildet werden.

Durch den Einsatz von nutzerdefinierbaren Parametern, beliebig erweiterbaren Regeln und Wörterbuchbegriffen kann der Anwender auf die Ergebnisse des Verarbeitungsprozesses Einfluss nehmen, womit ein anpassungsfähiges Verfahren zur Extraktion von Informationen aus semi- und unstrukturierten Inhalten realisiert ist.

Abstract

Within the scope of this work a flexible method was developed which extracts as much information as possible from Wismar's land registers and stores it into a database. Therefor, an iterative proceeding was implemented which uses the gained data from some source documents for the evaluation of the remaining ones.

The emphasis of the method is placed on the successive analysis of the document contents. For this purpose, the source documents are initially transformed into better evaluable formats, after what the interpretation of layout and structure information is performed. After the normalization of the source files parsers are used to analyze the structured document fragments, whereas dictionaries, letter- and speech-oriented techniques are applied to unstructured contents. User-definable semantic rules are used to identify ambiguous and not recognized data. Finally, information logically belonging together are structured and all analyzed data are mapped to relational database tables.

By applying user-definable parameters, expandable rules and dictionaries the user is able to exert influence on the processing results, whereby an adjustable method for information extraction from semi- and unstructured contents is realized.

CR-Classification

H.3.1: Content Analysis and Indexing
- Dictionaries
- Linguistic Processing

Key Words

Structure Mining, Information Extraction, Wrapper, Parser, Dictionary-based Proceedings, Letter- and Speech-oriented Methods

Inhaltsverzeichnis

Zusammenfassung **2**

Inhaltsverzeichnis **3**

Aufgabenstellung **6**

1 Einführung **8**
1.1 Motivation 8
1.2 Der Mining-Begriff 9

2 Die Ausgangsdokumente **14**
2.1 Das Abkürzungsverzeichnis 14
2.2 Das Personenverzeichnis 15
2.3 Das Grundbuch 15
2.4 Sprachliche Besonderheiten 18

3 Konzeption **23**
3.1 Gesamtarchitektur 23
3.2 Umwandlung des Eingabeformats 23
3.2.1 Die Ausgangsformate DOC und RTF 23
3.2.2 Die Zielformate TXT und HTML 26
3.2.3 Gewählte Zielformate für die einzelnen Dateien 27
3.3 Layout- und Strukturanalyse 28
3.3.1 Merkmale für die Layout- bzw. Strukturanalyse 28
3.3.2 XML als Zielformat 29
3.3.3 Ablauf der Analyse und XML-Generierung 31
3.4 Logische Strukturierung 34
3.5 Normalisierung 34
3.5.1 Ermittlung der Abkürzungsvarianten bei Abkürzungen von Wortgruppen . 34
3.5.2 Normalisierung der Leerzeichen 36
3.5.3 Normalisierung von einzelnen Worten 36
3.5.4 Splitten der Grundbuchdateien 40
3.6 Auswertung der regulären Anteile 40
3.6.1 Umsetzung des abstrakten Syntaxbaums auf eine XML-Struktur 40
3.6.2 Einpassung des XML-Baums in die Quellstruktur 41

3.7 Auswertung der Volltextanteile . . . 41
3.7.1 Entfernung der Satzzeichen . . . 45
3.7.2 Zerlegung in Token . . . 45
3.7.3 Untersuchung der Token auf Standardeigenschaften . . . 45
3.7.4 Überprüfung der Token mit Hilfe der Wörterbücher . . . 46
3.7.5 Aufbau der Wörterbücher . . . 47
3.7.6 Berechnung des Wortabstandes . . . 51
3.7.7 Wichtung des Wortabstandes . . . 53
3.7.8 Zuordnung des Tokens zu einem Wörterbuch . . . 53
3.7.9 Normalisierung der Rechtschreibung . . . 54
3.7.10 Nutzerdefinierbare Einstellungen . . . 57
3.8 Semantische Analyse . . . 58
3.8.1 Ablauf der Regelanwendung . . . 58
3.8.2 Definition der semantischen Regeln . . . 60
3.8.3 Beispiele für semantische Regeln . . . 61
3.9 Informationsstrukturierung . . . 62
3.9.1 Aufsplitten von reduzierten Begriffen . . . 62
3.9.2 Zusammenfügen von mehrteiligen Begriffen . . . 62
3.9.3 Informationsgruppierung . . . 63
3.10 Abbildung auf Datenbanktabellen . . . 63
3.10.1 Grundlegende Ansätze zur Speicherung der Informationen . . . 63
3.10.2 Globale ID-Vergabe . . . 64
3.10.3 ER-Modelle . . . 65
3.11 Prüfmechanismus . . . 65
3.11.1 Integration eines Logbuches . . . 66
3.11.2 Validierung der Ergebnisdokumente mit DTDs . . . 70
3.11.3 Überprüfung der Informationsstrukturen anhand von Regeln . . . 70

4 Umsetzung & Implementierung 72
4.1 Konvertierung von RTF bzw. DOC in TXT . . . 72
4.2 Konvertierung von RTF in HTML . . . 72
4.3 Generierung von XML mit Wrappern . . . 77
4.4 Auswertung der regulären Anteile mit Parsern . . . 81
4.5 Umsetzung des Logbuches mit log4j . . . 84
4.6 DOM-Manipulation mit dom4j . . . 85
4.7 Speicherung der Daten in DB2 . . . 85
4.8 Aufbau der Java-Klassenstruktur . . . 87

5 Beurteilung der Ergebnisse 92
5.1 Layout- und Strukturanalyse . . . 92
5.2 Auswertung der regulären Anteile . . . 92
5.3 Analyse der Volltextanteile . . . 93
5.4 Semantische Analyse . . . 94
5.5 Informationsstrukturierung . . . 96

6 Zusammenfassung & Ausblick **100**

Literaturverzeichnis **102**

Abbildungsverzeichnis **106**

Tabellenverzeichnis **108**

A Semantische Regeln **109**

B Strukturierungsregeln **111**

C Regeln zur Überprüfung der Informationsstrukturen **113**

D Struktur der Datenbank **114**
D.1 Relationenschemata für das Abkürzungsverzeichnis 114
D.2 Relationenschemata für das Personenverzeichnis 115
D.3 Relationenschemata für das Grundbuch 115

E Laufendes Beispiel **118**
E.1 Quelldokument 118
E.2 Ergebnis nach der Normalisierung 119
E.3 Ergebnis nach der Auswertung der Volltextanteile 119
E.4 Ergebnis nach der semantischen Analyse 122
E.5 Ergebnis nach der Informationsstrukturierung 122

Aufgabenstellung

Bearbeiter

- Manja Nelius
  manel@informatik.uni-rostock.de

Betreuender Hochschullehrer

- Prof. Dr. Andreas Heuer

Zweitgutachter

- Prof. Dr. Peter Forbrig

Betreuerin

- Dr. Meike Klettke

Literatur

- Literatur zum Themenkomplex Structure Mining im Projekt Mefis (Abschlussbericht zur ersten Projektphase)
- Studienarbeit von Andreas Schulz
- Literatur zu Wrappern
- Studienarbeit von Rolf Gohla
- Vorwort der Wismarer Grundbücher zum Verständnis des Aufbaus

Beginn der Bearbeitung

- 23. März 2004

Abgabe der Diplomarbeit

- 29. Oktober 2004

Structure Mining für die Wismarer Grundbücher des 16. – 19. Jahrhunderts

Charakterisierung

Implementierung/Konzeption

Kurzbeschreibung

Aufgabe dieser Diplomarbeit ist es, ein Verfahren zu entwickeln und zu implementieren, das aus den Wismarer Grundbüchern so viele Informationen wie möglich extrahiert und in Form einer Datenbank (DB2) für andere Informationssysteme bereitstellt.

Bislang liegen diese Informationen in Form von Word-Dokumenten elektronisch vor, in diesen Dokumenten ist eine bestimmte Struktur durch das Layout der einzelnen Einträge erkennbar.

Zu Beginn der Arbeit soll der Aufbau und der Inhalt der einzelnen Grundbucheinträge und des Personenindexes untersucht werden. Dabei muss auch eine Einarbeitung in die sprachlichen Besonderheiten des 17. und 18. Jahrhunderts und in die speziellen Schreibweisen in den Grundbüchern (verwendete Abkürzungen etc.) erfolgen. Ebenfalls zu Beginn muss eine Methode entwickelt werden, um die Word-Dateien in andere (besser auswertbare) Formate zu übertragen.

Ziel der Arbeit ist die Entwicklung eines Algorithmus, der die Informationen, die durch das Layout der Grundbucheinträge erkennbar sind, auswertet. Darüber hinaus soll die Auswertung der Texte durch eine Analyse der Satzstrukturen und wörterbuchbasierte Verfahren erfolgen. Es ist zu überprüfen, ob eine iterative Vorgehensweise Vorteile bringt, zum Beispiel ist vorstellbar, dass im Algorithmus das Personenregister zuerst ausgewertet wird und dieses dann zur Ermittlung der Namen in den einzelnen Grundbucheinträgen herangezogen wird.

Ebenfalls soll der Algorithmus einen Prüfmechanismus beinhalten, der beim Parsen der einzelnen Einträge feststellt, ob der eingesetzte Algorithmus die richtigen Ergebnisse ableitete. Dadurch soll es möglich sein, Fehlerfälle zu erkennen und entsprechend eine Erweiterung der Wörterbücher oder eine manuelle Nachbehandlung vorzunehmen.

Zum Abschluss der Arbeit soll eine Untersuchung erfolgen, welche Informationen durch den Algorithmus abzuleiten sind. Dazu soll die entwickelte Methode auf den ersten Band der vorliegenden Grundbücher angewendet werden. Die Fälle, in denen keine automatischen Verfahren zur Extraktion der Informationen eingesetzt werden können, sind zu analysieren.

Kapitel 1

Einführung

1.1 Motivation

Mit dem Wismarer Grundbuch von 1677/1680 liegt eine historische Quelle mit fundamentaler Bedeutung für die Wismarer Stadtgeschichte des Mittelalters und der frühen Neuzeit vor. Aufgrund der damaligen erheblichen Veränderungen des Gebäudebestandes und der Grundeigentumsverhältnisse im Verlauf der kriegerischen Jahrzehnte des 17. Jahrhunderts wurde das Wismarer Stadtbuch ursprünglich als Hilfsmittel für Fragen der innerstädtischen Grundstücksgeschäfte vom damaligen Bürgermeister Antonius Scheffel angelegt. Heute steht damit eine Quelle zur Verfügung, die wertvolle Erkenntnisse zu den unterschiedlichsten Fragestellungen liefern kann. Zum einen wird dadurch die flächendeckende Rekonstruktion des historischen Stadtbildes Wismars mit seinen Straßen, Grundstücken und Gebäuden möglich, zum anderen lassen sich heutige Eigentumsverhältnisse in frühere Jahrhunderte systematisch zurückverfolgen.

Neben personengeschichtlichen Nachforschungen spielt auch die Informationsgewinnung über die Entwicklung des Immobilienmarktes und die Baukonjunktur eine große Rolle. Die Veränderungen in den Grundstücks- und Gebäudenutzungen stellen einen wichtigen Indikator für die wirtschaftliche Entwicklung Wismars im Spätmittelalter und der frühen Neuzeit dar. Ebenso lassen sich daraus Erkenntnisse über den Stellenwert Wismars in der Geschichte und im Vergleich der Hansestädte gewinnen. [Mue02]

Wegen dieser vielfältigen Nutzungsmöglichkeiten zählte das „Alte Stadtbuch“ in den vergangenen Jahren im Archiv der Hansestadt Wismar zu den am häufigsten genutzten Quellen. Durch die Historische Kommission für Mecklenburg wurde das handschriftliche Grundbuch nun elektronisch aufbereitet und veröffentlicht [Mue02]. Dennoch ist die Informationsgewinnung aus dem Stadtbuch in den meisten Fällen keineswegs einfach, da benötigte Daten immer noch manuell in den Einträgen gesucht und daraus extrahiert werden müssen. Zur Erleichterung der Recherche ist es daher notwendig, Informationen automatisch aus dem Grundbuch zu gewinnen und für effektive Anfragen in einer Datenbank abzulegen. Diese Aspekte sollen Thema dieser Diplomarbeit sein.

1.2 Der Mining-Begriff

Ursprünglich lässt sich der Begriff des Mining auf den Bergbau zurückführen. Dort bezeichnet er den Prozess der Entdeckung von kleinen, wertvollen Klumpen — den so genannten Nuggets — in einer riesigen Menge Rohmaterial. Bezogen auf den IT-Bereich bedeutet Mining die Wissensentdeckung bzw. -extraktion aus einer großen Menge von Daten [HK01]. Basierend darauf, aus welcher Art von Daten Wissen gewonnen werden soll, bildeten sich in den letzten Jahrzehnten mehrere Anwendungsbereiche des Mining heraus, wovon die wichtigsten in den nächsten Abschnitten kurz vorgestellt werden sollen.

Data Mining

Data Mining gehört wahrscheinlich zu den bekanntesten Teilgebieten des Mining. Obwohl der Begriff häufig als Synonym für Knowlegde Discovery in Databases (KDD) verwendet wird, lassen sich diese beiden Begriffe nach [Sch02] wie folgt abgrenzen:

- **KDD** bezeichnet den Prozess der Identifikation von interessanten — nicht trivialen, impliziten, vorher unbekannten und potentiell nützlichen — Informationen oder Mustern aus Daten in großen Datenbanken.
- **Data Mining** ist ein Teilschritt des KDD-Prozesses, der aus Algorithmen besteht, welche in akzeptabler Rechenzeit aus einer vorgegebenen Datenbasis eine Menge von Mustern liefern.

Der Prozess der Wissensentdeckung in Datenbanken setzt sich nach [HK01] somit aus folgenden Abschnitten zusammen:

1. **Datenbereinigung**
   Entfernen von Fehlern und inkonsistenten Daten.
2. **Datenintegration**
   Zusammenführung von mehreren Datenquellen.
3. **Datenselektion**
   Holen von relevanten Informationen — bezüglich der Analyseaufgabe — aus der Datenbank.
4. **Datentransformation**
   Umwandlung der Daten in eine für das Mining angemessene Form, zum Beispiel durch Aggregationsoperationen.
5. **Data Mining**
   Anwendung von intelligenten Methoden, um Datenmuster zu extrahieren.
6. **Pattern Evaluation**
   Identifikation der wirklich interessierenden, wissensrepräsentierenden Patterns basierend auf bestimmten Interessantheitsmaßen.
7. **Wissensrepräsentation**
   Präsentation des gewonnenen Wissens durch Anwendung von Visualisierungs- und Wissensdarstellungsmethoden.

Der Data-Mining-Schritt kann dabei mit dem Nutzer oder einer Wissensbasis interagieren und konzentriert sich auf die Wissensentdeckung in strukturierten Daten.

Image Mining

Image Mining befasst sich im Gegensatz zu Data Mining nicht mit der Wissensgewinnung aus Datenbanken, sondern mit der Extraktion von Informationen aus Bildern. Dabei spielen beispielsweise Merkmale wie Größe, Farbe, Form, Textur, Ausrichtung, relative Positionen und Strukturen von Objekten oder Regionen des Bildes eine Rolle. Ein interessantes Anwendungsgebiet ist zum Beispiel die Analyse von Internetbildern zum automatischen Sperren von pornographischen Inhalten für Kinder.

Spatial Data Mining

Beim Spatial Data Mining dienen raumbezogene Daten als Quelle für Analysen und für die Gewinnung von neuen Informationen. Ziele des Spatial Mining sind dabei

- die Identifizierung räumlicher Cluster[1],
- die Erkennung solche Cluster verursachender Objekte und
- die Erklärung der für die räumliche Clusterbildung verantwortlichen Faktoren.

Spatial Data Mining koppelt dabei Data Mining mit Geo-Informationssystemen[2] [GG]. Ein Beispiel dafür ist das Projekt SPIN! des Centre for Computational Geography (CCG) der University of Leeds [CCG], das sich mit neuen Möglichkeiten für die Analyse von geobezogenen Daten beschäftigt und dafür ein Spatial-Data-Mining-System entwickelt.

Web Mining

Web Mining befasst sich mit der Gewinnung von Informationen aus dem Web und wendet dafür Data-Mining-Techniken auf unterschiedliche Web-Datenressourcen an. Besonders im Bereich des E-Commerce findet das Web Mining rege Anwendung, zum Beispiel um neue potentielle Kunden zu finden, zum Beobachten und Analysieren der Konkurrenz oder zum Zusammentragen von Informationen aus verschiedenen Web-Seiten für einen Produktkatalog oder Newsletter. Nach [Com] kann man Web Mining in folgende Hauptabschnitte unterteilen (siehe auch Abbildung 1.1):

- **Web Content Mining**
  Web Content Mining beschreibt das Entdecken nützlicher Informationen in Web-Inhalten, -Daten und –Dokumenten. Web-Inhalte sind dabei Texte, Bilder, Audio, Video, Metadaten und Hyperlinks. Dokumente können unstrukturiert oder semistrukturiert (HTML-Dateien) sein. Zur Analyse des textuellen Inhalts kommen unter anderem Text-Mining-Methoden zum Einsatz.

[1] Das Cluster selbst ist das Grundelement der Clusteranalyse, einer statistischen Methode zur Aufdeckung von Raummustern. [GG]

[2] Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden. [GG]

- **Web Structure Mining**
  Beim Web Structure Mining richtet sich das Augenmerk auf die Struktur einer Web-Seite — vor allem ihre Verlinkung — mit dem Ziel, die interne Navigation zu verbessern und äußere Verweise, die in der Regel von inhaltlich verwandten Seiten kommen, zu identifizieren.

- **Web Usage Mining**
  Beim Web Usage Mining wird das Verhalten der Seitenbesucher mit Hilfe von Web-Server-Logdateien ausgewertet, um anhand der dort protokollierten Daten wichtige Informationen für verschiedene Fragestellungen zu gewinnen. Beispiele für Anwendungsmöglichkeiten wären die Einteilung der Seitenbesucher in Benutzergruppen nach dem Navigationsverhalten oder die Optimierung des Web-Auftritts durch die Vorhersage von Seitenaufrufen. Außerdem spielen Marketing-Aspekte eine wichtige Rolle.

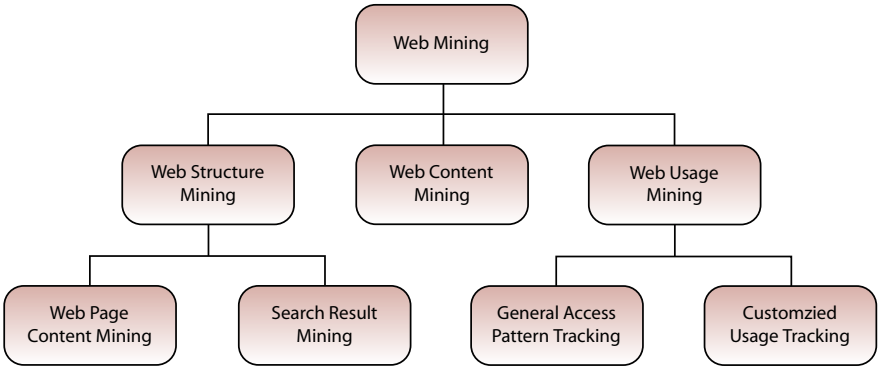


Abbildung 1.1: *Taxonomie des Web Mining nach [Sch02]*

Text Mining

Text stellt nach wie vor den wichtigsten Träger von Information dar. Da textuelle Inhalte jedoch meist unstrukturiert vorliegen, sind deren Informationen nur schwer zugänglich. Um eine effiziente und zielgerichtete Nutzung von Textdaten zu ermöglichen, sind daher Techniken und Algorithmen zur automatischen Analyse (Typisierung, Organisation, Strukturierung) von unstrukturierten Daten nötig, was Gegenstand des Text Mining ist. Text Mining kann dabei als interdisziplinäres Gebiet von

- Textanalyse,
- Informationsextraktion,
- Textzusammenfassung,
- Clustering,
- Kategorisierung,
- Visualisierung,

- Datenbanktechnologie und
- Data Mining

verstanden werden. [Ver02]

Bei der Nutzung von Texten als Wissensquelle ist die Informationsextraktion (IE) ein essentieller Aspekt und stellt inzwischen ein eigenes Forschungsgebiet dar. Während dieser Phase werden gezielt domänenspezifische Informationen aus freien Texten extrahiert und strukturiert, wobei irrelevante Informationen überlesen werden. Diese Phase ist dabei stark von Techniken und Theorien der Computerlinguistik beeinflusst, wie an der verallgemeinerten Architektur eines IE-Systems in Abbildung 1.2 zu sehen ist.

Abbildung 1.2: *Verallgemeinerte Architektur eines IE-Systems nach [HG03]*

Eine spezielle Variante der Informationsextraktion ist die Named Entity Recognition, die sich mit der Erkennung und Klassifikation bestimmter Wörter oder Wortsequenzen hinsichtlich vorgegebener sematischer Typen (zum Beispiel Personennamen, Datums- oder Mengenangaben, Firmenbezeichnungen etc.) befasst. Dabei kommen musterbasierte (`\d\d`), kontextuelle (zum Beispiel Trigger „Mr.“) und lexikonbasierte Verfahren zum Einsatz. [Kle04]

Im Vergleich zum Information Retrieval, das relevante Dokumente liefert, selektiert die Informationsextraktion bestimmte Informationen aus relevanten Texten. Da diese Relevanz domänenspezifisch ist, sind IE-Systeme hinsichtlich ihres Anwendungsbereiches stark eingeschränkt. Je umfangreicher jedoch die Textmengen sind, desto effektiver ist die Informationsextraktion gegenüber dem Information Retrieval.

Die besonderen Herausforderungen beim Text Mining sind auf die Komplexität der natürlichen Sprache zurückzuführen. Der deutsche Wortschatz umfasst zum Beispiel ca. 10^4–10^5 Wörter. Neben Ambiguitäten[3], Multilingualismus[4] und Anaphern[5] erschweren Umlaute und Komposita[6] die Definition effizienter formaler Grammatiken. Auch reicht im Deutschen bei der Namenser-

[3]Lexikalische oder syntaktische Mehrdeutigkeit. [Dud01]
[4]Mehrsprachigkeit, Vielsprachigkeit. [Dud01]
[5]Zurückverweisendes Element eines Textes (zum Beispiel: Die Frau... *Sie* war sehr elegant). [Dud01]
[6]Zusammengesetzte Wörter, Zusammensetzungen. [Dud01]

kennung die Unterscheidung zwischen Groß- und Kleinschreibung allein nicht aus [HG03]. Zur Unterstützung der automatischen Textanalyse wird daher eine Vorstufe zum Parsen[7], das so genannte Tagging, eingefügt, während derer bestimmten Daten im Text Etiketten zur Charakterisierung der Art der Information zugewiesen werden. Beim Part-Of-Speech-Tagging (POS-Tagging) wird zum Beispiel jeder Wortform eine eindeutige grammatische Kategorie zugeordnet. Dieser Vorgang kann von so genannten Taggern mit einer geringen Fehlerquote vollautomatisch durchgeführt werden und die Möglichkeiten der effektiven Suche in Texten besonders für Sprachen mit einer hohen Anzahl von Homographen[8] extrem steigern [Zie97].

Neben der Informationsextraktion, die auch in dieser Arbeit einen wichtigen Aspekt darstellt, beinhaltet das Text Mining weiterhin Aufgaben wie die Erstellung von Textzusammenfassungen und die Klassifizierung von Texten, worauf hier aber nicht näher eingegangen werden soll.

Structure Mining

Der Begriff des Structure Mining ist im Gegensatz zu den vorangegangenen Beispielen ein sehr neuer Begriff und deswegen noch nicht wissenschaftlich definiert. Grundsätzlich beschäftigt sich dieses Teilgebiet nicht mit der Wissensgewinnung aus schon vorhandenen strukturierten Informationen wie das Data Mining, sondern mit der Datengewinnung aus Struktur- und Layoutinformationen von Dokumenten verschiedenster Art. Dazu müssen zunächst relevante Strukturen bestimmt werden, wohingegen beim Web Structure Mining bereits bekannte Strukturen zur Informationsgewinnung eingesetzt werden. Die identifizierten Strukturen — beispielsweise Textauszeichnungen wie Überschriften, Absätze oder Tabellenstrukturen — werden dann unterstützend zur Interpretation von unstrukturierten Textanteilen und deren Bedeutung für den Gesamtkontext herangezogen.

Structure Mining befasst sich demnach mit der Analyse von unstrukturierten Informationen und besitzt in der Anwendung auf textuelle Daten starke Ähnlichkeiten zum Text Mining. Der Unterschied liegt allerdings in der Tatsache, dass beim Text Mining für die Analyse des Inhaltes dessen Layout unbeachtet bleibt. Gerade diese Strukturinformationen können jedoch wertvolle Erkenntnisse über die Bedeutung von einzelnen Textabschnitten liefern, da gerade das die Aufgabe logischer Textauszeichnung ist.

[7] Während des Parsens erfolgt eine syntaktische Überprüfung der Eingangsdaten mit Hilfe eines Parsers. Ein Parser (von englisch *to parse* – analysieren) ist ein Programm, das entscheidet, ob eine Eingabe zur Sprache einer bestimmten Grammatik gehört. [Wik]

[8] Ein Wort, das sich in der Aussprache von einem anderen, gleich geschriebenen unterscheidet (zum Beispiel: Tenor „Haltung“ neben Tenor „hohe Männerstimme“). [Dud01]

Kapitel 2

Die Ausgangsdokumente

Für ein besseres Verständnis der Aufgabenstellung und den damit einhergehenden Herausforderungen werden in diesem Abschnitt zunächst die einzelnen Ausgangsdateien mit ihrer Struktur und ihren Merkmalen vorgestellt. Für die Grundbuchdokumente wird zusätzlich eine Evaluierung der in den Volltextanteilen enthaltenen Informationstypen angeführt. Darüber hinaus werden sprachliche Besonderheiten und Konventionen seitens des Herausgebers bei der Erstellung der Quellensicherung angesprochen.

2.1 Das Abkürzungsverzeichnis

Das Abkürzungsverzeichnis listet alle im Grundbuch und Personenindex verwendeten Abkürzungen mit den jeweiligen Erklärungen auf. In einigen Fällen existieren mehrere Abkürzungsvarianten, die hintereinander durch Komma getrennt aufgeführt werden. Sind ganze Wortgruppen abgekürzt, werden die einzelnen Wörter mit ihren Varianten nacheinander aufgelistet. Beispiele dafür sind in Abbildung 2.1 ersichtlich. Da die einzelnen Einträge sehr regulär aufgebaut sind und einem bestimmten Muster folgen, bietet sich in der späteren Informationsanalyse die Auswertung mit grammatikalischen Regeln an.

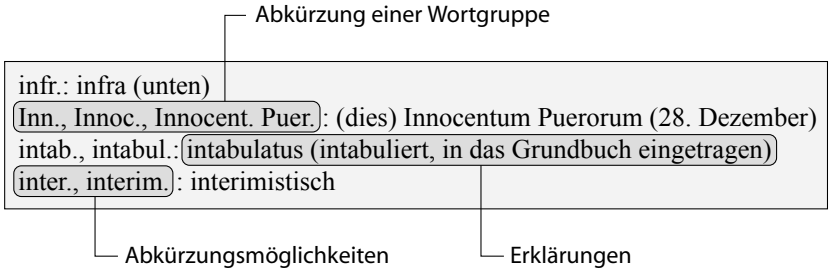


Abbildung 2.1: *Aufbau des Abkürzungsverzeichnisses*

2.2 Das Personenverzeichnis

Das Personenverzeichnis besteht aus insgesamt 22 DOC[9]-Dateien, wobei jede davon die Nachnamen zu einem oder mehreren bestimmten Buchstaben des Alphabets beinhaltet. Dieser Anfangsbuchstabe ist als Überschrift in jeder Datei in der ersten Zeile aufgeführt. Die darauf folgenden Einträge kann man — zeilenweise betrachtet — in vier verschiedene Kategorien unterteilen:

1. Nennung eines Familiennamens, gegebenenfalls mit Varianten der Schreibung,
2. Nennung eines Familiennamens, gegebenenfalls mit Varianten der Schreibung, gefolgt von den Angaben zu einer Person (Vorname, Informationen, Grundbuchnummern),
3. Angaben zu einer Person (Vorname, Berufsangaben, weitere Informationen, Grundbuchnummern),
4. Verweise von Familiennamenvarianten auf die Hauptschreibweise.

Dabei folgen die Einträge der Kategorie 3 immer einem Eintrag der Kategorie 1. Die Angabe des Vornamens zu einer Person ist optional. In Klammern aufgeführte Rufnamen sind aus anderen Quellen ergänzt. Bei Varianten eines Vornamens wird nur die moderne Form angeführt (zum Beispiel „Hans“ und „Hanß“). In machen Fällen jedoch werden verschiedene Schreibweisen von Namen mit Klammern gekennzeichnet (beispielsweise „Brand(anus)“). Namen können ebenfalls durch einen Punkt abgekürzt sein.

Berufsangaben und andere Angaben zur Person (wie zum Beispiel Ortsangaben) werden, soweit vorhanden, direkt hinter dem Vornamen in Klammern angeführt. Ehefrauen und Witwen werden, wenn ausdrücklich im Grundbuch als solche genannt, bei den Ehemännern aufgelistet. Zu den Grundbuchnummern können ebenfalls erklärende Informationen angeführt sein, welche dann in vorangehenden Klammern erfasst sind. Diese beziehen sich auf alle darauf folgenden Grundbuchnummern.

Der Aufbau des Personenverzeichnisses ist in Abbildung 2.2 anhand eines Auszugs zur Verdeutlichung aufgeschlüsselt. Ebenso wie beim Abkürzungsverzeichnis sind auch die Einträge des Personenindexes sehr regulär aufgebaut, was die spätere Analyse der Informationen mit Hilfe von grammatikalischen Regeln bedingt.

2.3 Das Grundbuch

Das Grundbuch lag zur Bearbeitung dieser Arbeit aufgeteilt in zehn RTF[10]-Dokumente vor. Darin sind die Grundstücke mit ihren Eintragungen straßenweise aufgeführt, wobei jeder Grundbucheintrag in mehrere, nachfolgend erläuterte Rubriken unterteilt ist:

1. Angaben zur Grundbuchnummer (die neu vorgenommene durchgängige Zählung für das gesamte Grundbuch, in Klammern die Zählung Scheffels, die Seitenangabe, auf der sich die Eintragung in der Reinschrift von 1680 befindet, und die neue Stadtbuchnummer ab 1838),

[9] Abkürzung für *Document*. Verbreitetes Dateiformat für Textdokumente, das in Microsoft Word verwendet wird. [Wik]

[10] Abkürzung für *Rich Text Format*. Dateiformat für Texte, das von Microsoft eingeführt wurde und zum Datenaustausch zwischen Textverarbeitungsprogrammen verschiedener Hersteller dient. [Wik]

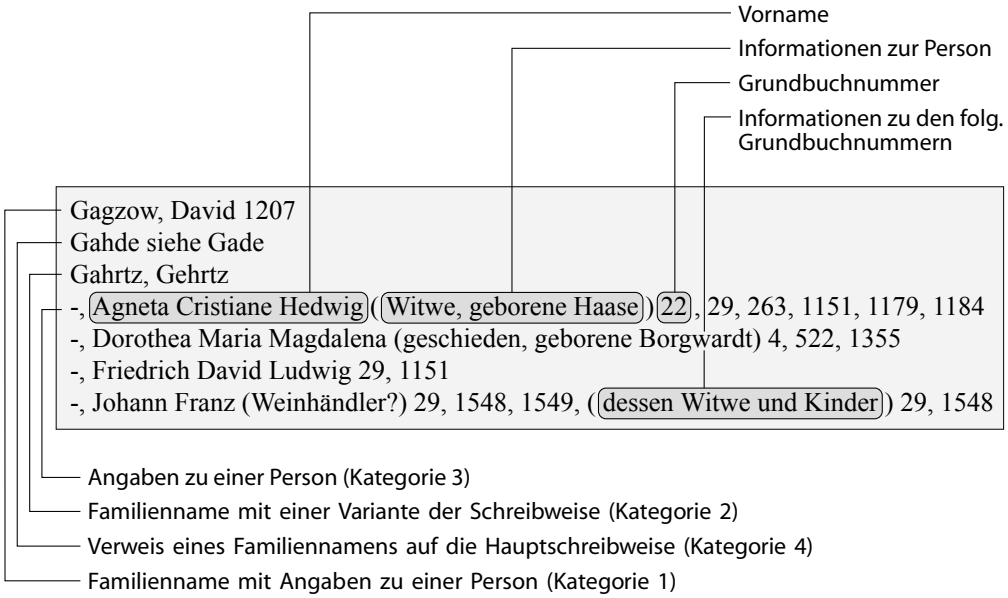


Abbildung 2.2: *Aufbau des Personenverzeichnisses*

2. die frühere Qualität des jeweiligen Gebäudes bzw. Grundstückes,
3. die Qualität für die Gegenwart Scheffels,
4. Bezüge zwischen dem Gebäude bzw. Grundstück und seinem Zubehör, Hinweise über Wasserläufe oder die Tilgung von Braugerechtigkeit bei ehemaligen Brauhäusern,
5. chronologische Abfolge der Eigentümer des Objektes (in der Regel Angaben dazu über die Art und den Zeitpunkt der Erwerbung), das Datum der Übertragung in das neue Stadtbuch,
6. Angaben über auf dem Grundstück lastende Renten und Kapitalien (Empfänger, Zeitpunkt der Einrichtung, eventuell Zeitpunkt ihrer Zahlung, Tilgung bzw. Übertragung auf andere Personen).

Dabei muss nicht jede Rubrik für jeden Grundbucheintrag zwingend vorhanden sein. Auch gibt es Angaben über einige Objekte, die aufgenommen, aber nicht gezählt wurden. Dazu gehören frühere oder spätere Grundstücksteilungen oder -ergänzungen und Grundstücke, die zwischen zwei anderen liegen. Demgegenüber wurde die konsequente Einzelzählung auch von zusammengehörigen Grundstücken bzw. Gebäuden (zum Beispiel „Haus“ und „Torweg“) durchgeführt. Zu jeder Eintragung sowie zu den Straßenangaben können darüber hinaus weitere Anmerkungen vorhanden sein, die den Charakter eines Volltextes haben und somit schwer zu differenzieren sind.

Bei Rubriken mit Angabe der Qualität des Erwerbs von Grundstücken sowie mit Datierungen wurde vor und zwischen diesen Angaben stets ein Punkt gesetzt. Ebenso beendet ein Punkt konsequent den jeweiligen Eintrag in den Rubriken 5 und 6.

Kursive Angaben deuten auf Ergänzungen des Herausgebers hin, wobei *„Entwurf 1677“* erheblichere Abweichungen zwischen Entwurf und Reinschrift kenntlich macht. In Klammern an-

geführte Angaben bezeichnen irrtümliche und später veränderte Eintragungen. Ein Beispiel für einen Grundbucheintrag zeigt Abbildung 2.3.

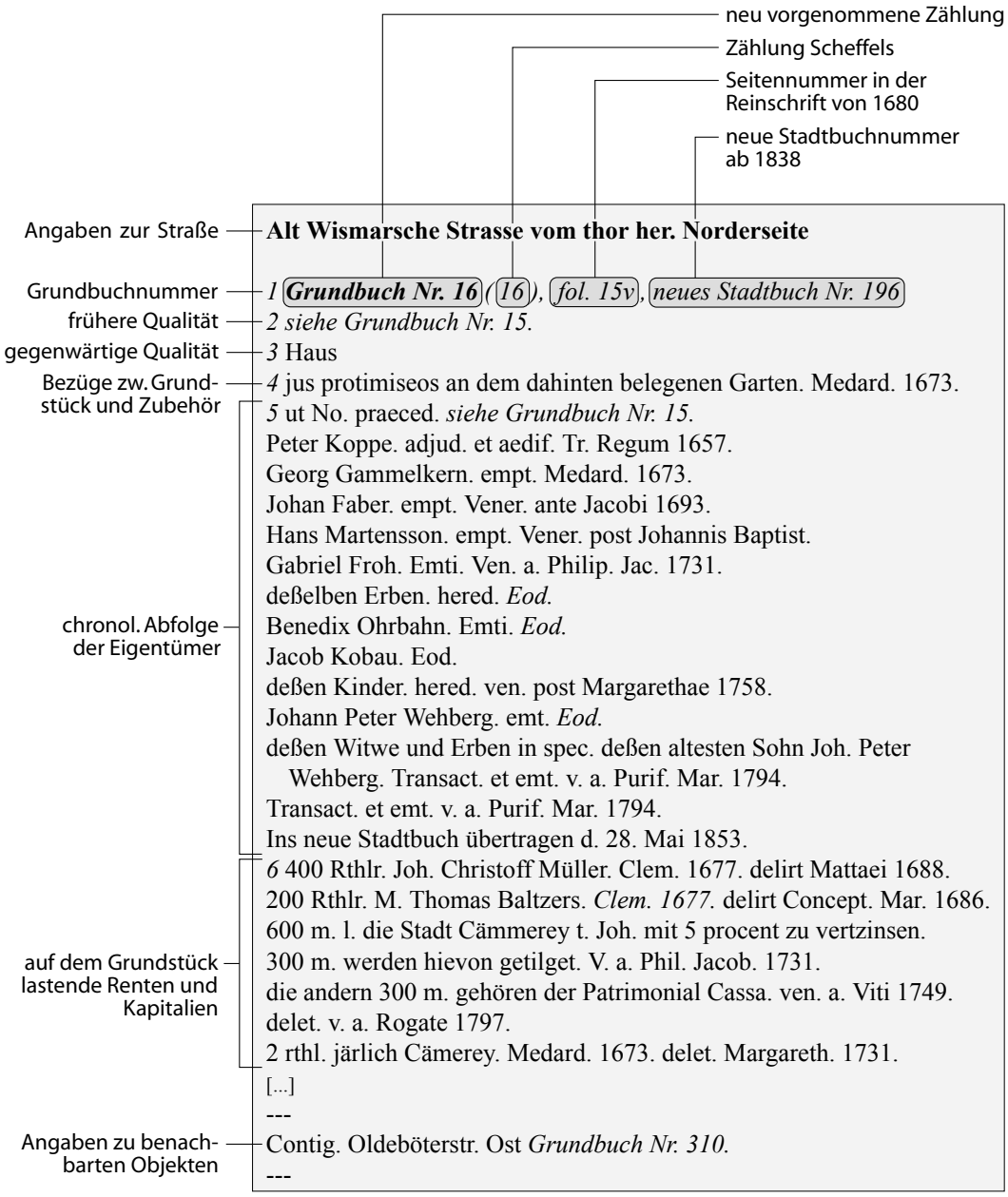


Abbildung 2.3: *Aufbau der Grundbucheinträge*

Im Gegensatz zum Abkürzungs- und Personenverzeichnis enthalten die Grundbuchdokumente größtenteils Volltextanteile. Nur die Angaben zur ersten Rubrik sind regulär aufgebaut und lassen sich durch grammatikalische Regeln beschreiben. Die restlichen Einträge sind hingegen irregulären Charakters. Um die darin enthaltenen Informationstypen zu identifizieren, wurde eine Evaluie-

rung des Volltextes anhand von Stichproben[11] durchgeführt. Dafür wurden die einzelnen Wörter — soweit möglich — bestimmten Kategorien zugeordnet, die sich später auch in der Wahl der Wörterbücher widerspiegeln. Darüber hinaus wurden auch Zahlen in Bezug auf ihre Bedeutung evaluiert. Worte, deren Bedeutung nicht eindeutig aus dem Kontext ersichtlich war, wurden bei der Auswertung nicht berücksichtigt, ebenso häufig auftretende Worte, die für den Inhalt irrelevant sind (zum Beispiel „der“). Wie in den statistischen Auswertungen in Abbildung 2.4 bis 2.6 zu sehen ist, ist die Häufigkeit der auftretenden Begriffe einer Kategorie stark abhängig von der Art des Eintrags. So sind zum Beispiel Angaben zu Himmelsrichtungen besonders oft bei den Straßeninformationen zu finden. Feiertage treten hingegen bei allen Einträgen häufig auf. Auf die Ergebnisse dieser Auswertung stützen sich ebenfalls die erstellten Regeln für die späteren Phasen der semantischen Analyse und Informationsstrukturierung.

2.4 Sprachliche Besonderheiten und Konventionen des Herausgebers

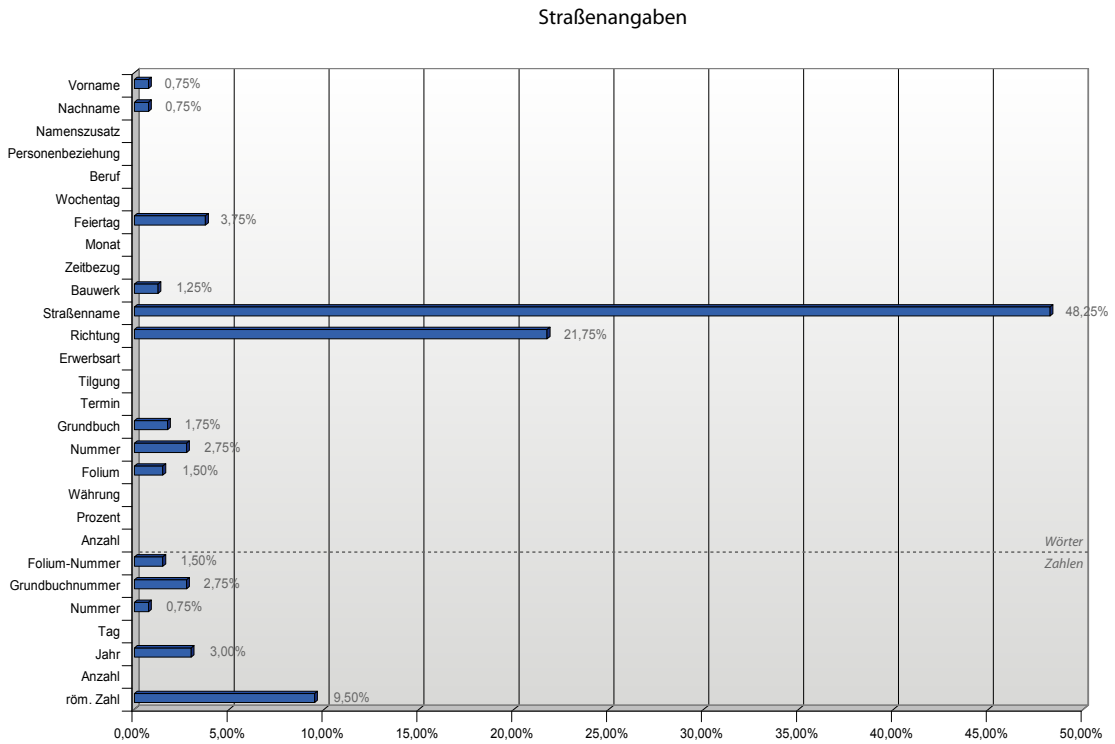
Die Rechtschreibung im 17. Jahrhundert ist vor allem dadurch charakterisiert, dass individuell so geschrieben wurde, wie man es für richtig hielt. Außerdem begründete die politische und geographische Aufteilung gerade des deutschen Sprachgebiets etliche Schreibvarianten [Mar]. So konnte die Rechtschreibung sogar zwischen zwei benachbarten Orten stark variieren. Darüber hinaus stand Wismar unter anderem unter schwedischem Einfluss, was sich ebenfalls auf die Sprache auswirkte. Weiterhin sind die Quellen dadurch charakterisiert, dass unter anderem für Erwerbungstitel, Feste und Heiligentage die lateinischen Entsprechungen anstelle der deutschen Begriffe verwendet wurden.

Bei der Erstellung dieser Quellensicherung wurden der Originaltext sowie die Schreibungen beibehalten. Dadurch treten Schwankungen bei der Groß- und Kleinschreibung auf, wobei für die Heiligen- und Feiertage, die als klassische Termine bei der Angabe von Zeitpunkten verwendet wurden, die Großschreibung vereinheitlicht wurde. Unsicherheiten in der Quellensicherung bezüglich Namen und Informationen, die in der teilweise schlechten Lesbarkeit der handschriftlichen Quellen begründet sind, wurden mit „?“ bzw. „...“ gekennzeichnet.

Weiterhin tauchen im Originaltext mehrere Varianten für Personennamen auf. Im Personenverzeichnis wurden jedoch nur für die Familiennamen die verschiedenen Schreibweisen aufgeführt, während für Vornamen lediglich die moderne Form angegeben wurde. Diese Tatsache erschwert bei der späteren Textanalyse die Identifikation von Wörtern als Vornamen. Vorteilhaft für die spätere Auswertung des Inhalts wirkt sich jedoch die Tatsache aus, dass jeweils eine Zeile — beendet durch einen Zeilenumbruch — Informationen zu einem bestimmten Bedeutungsschwerpunkt beinhaltet. Die einzelnen Zeilen können somit inhaltlich und strukturell voneinander isoliert betrachtet werden.

Die angesprochenen sprachlichen Besonderheiten haben große Auswirkungen auf die anwendbaren Verfahren zur Verarbeitung der Grundbuchtexte. Insbesondere der mehrsprachige Wortschatz sowie die variierende Rechtschreibung und Grammatik, für die im 17. Jahrhundert noch

[11] Der Stichprobenumfang betrug dabei 1,79% aller Inhalte (das heißt, 517 von insgesamt 28.926 Rubrikeneinträgen wurden in die Auswertung mit einbezogen). Eine umfangreichere Stichprobe war im zeitlichen Rahmen dieser Arbeit nicht realisierbar.



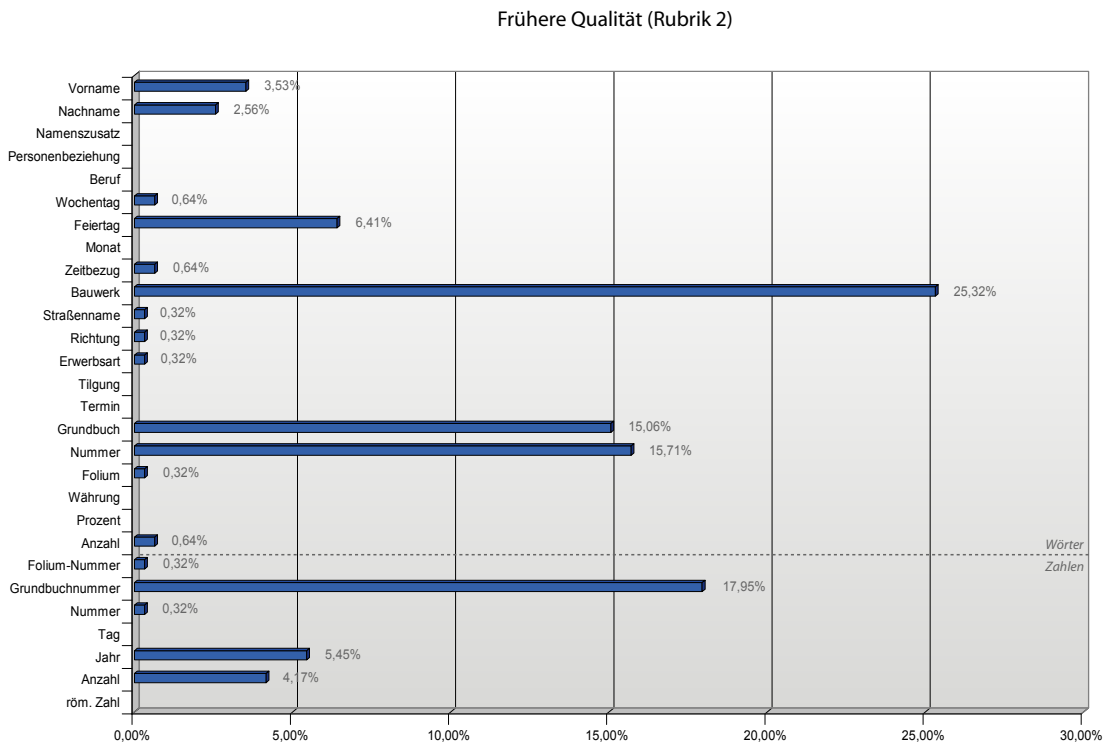
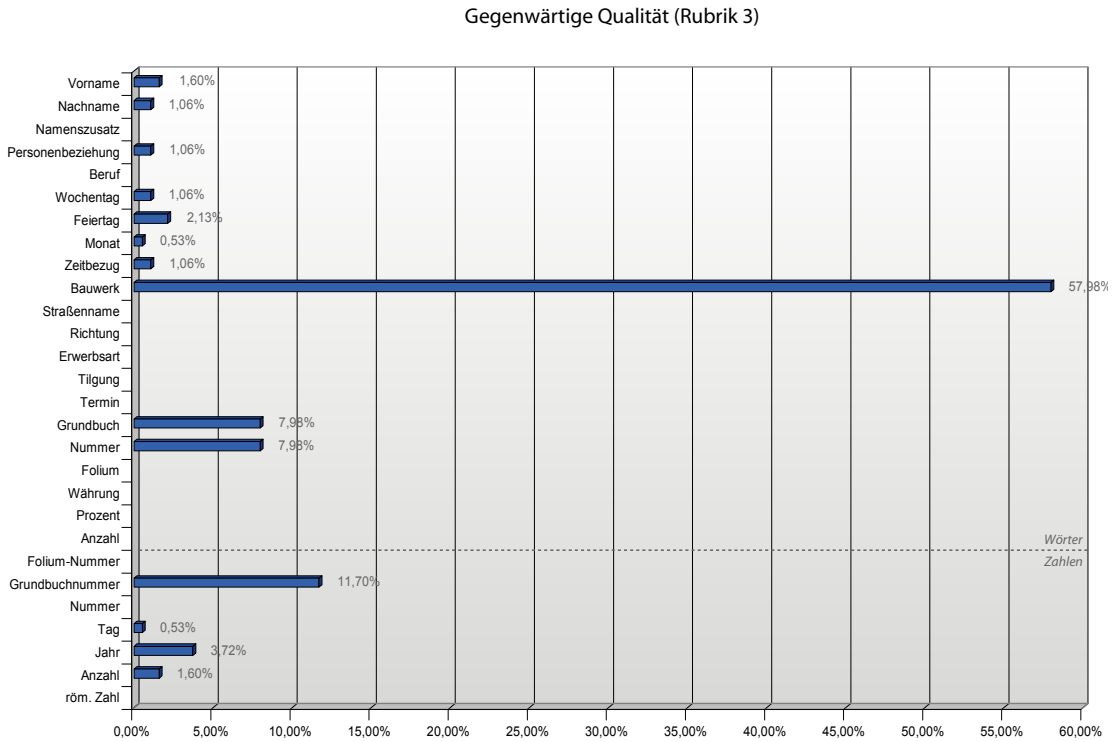


Abbildung 2.4: *Evaluierung der Informationstypen im Volltextanteil — Teil 1*



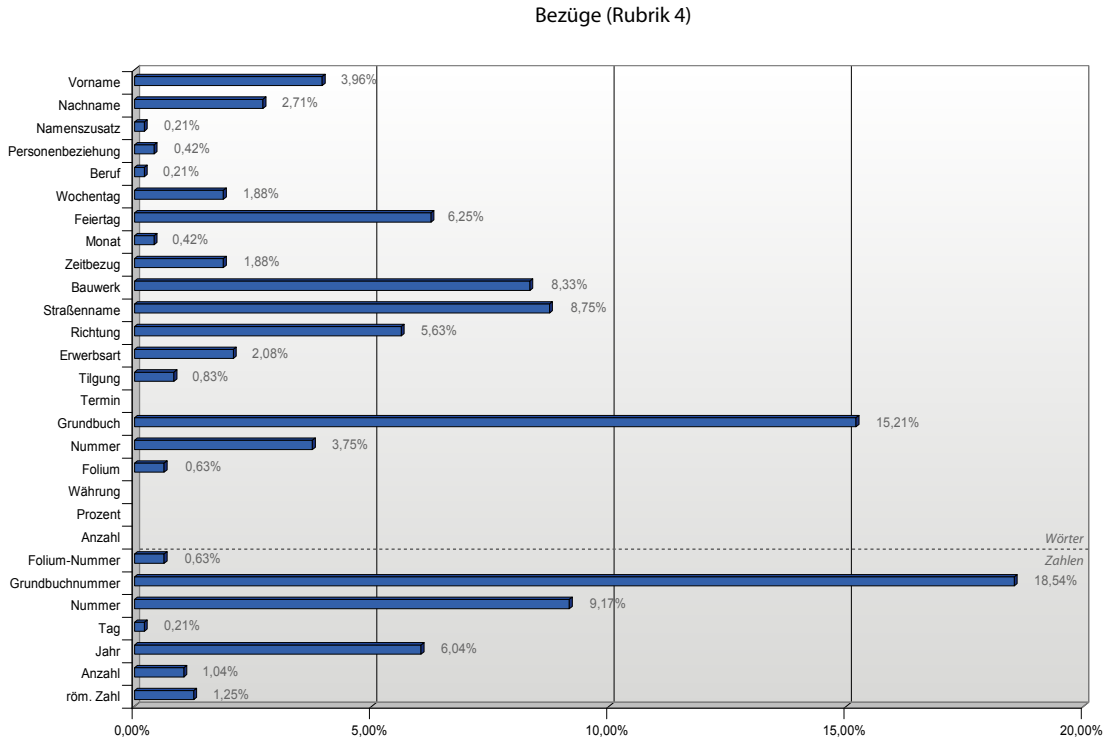
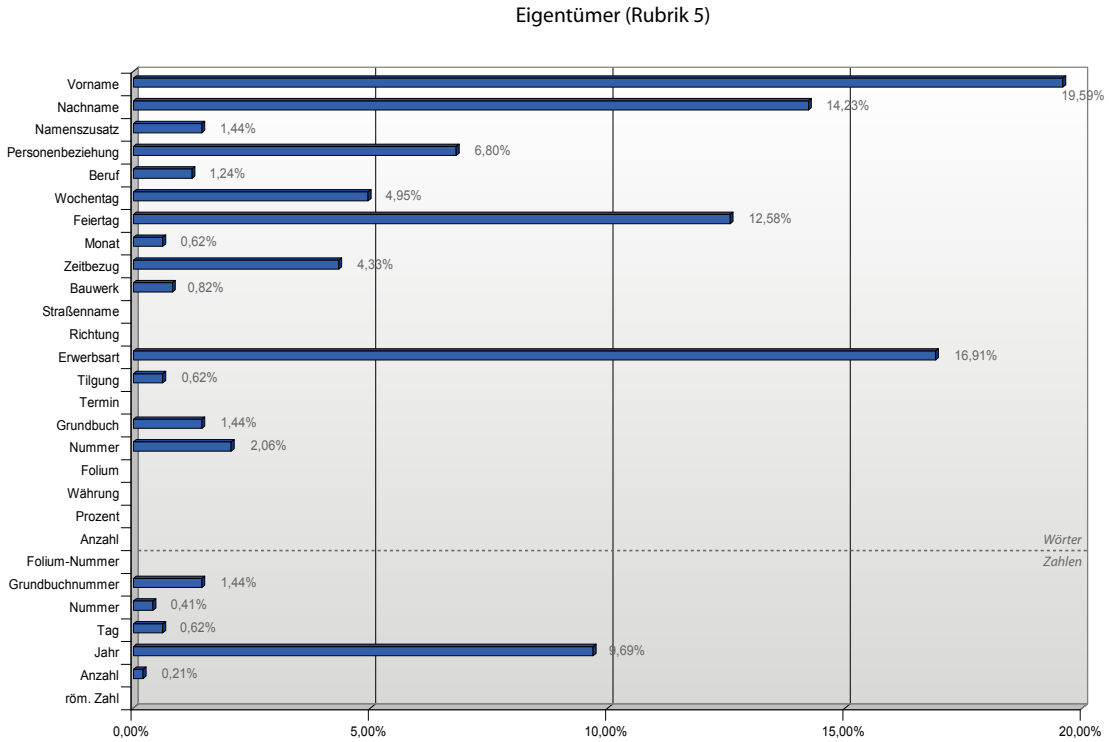


Abbildung 2.5: *Evaluierung der Informationstypen im Volltextanteil — Teil 2*



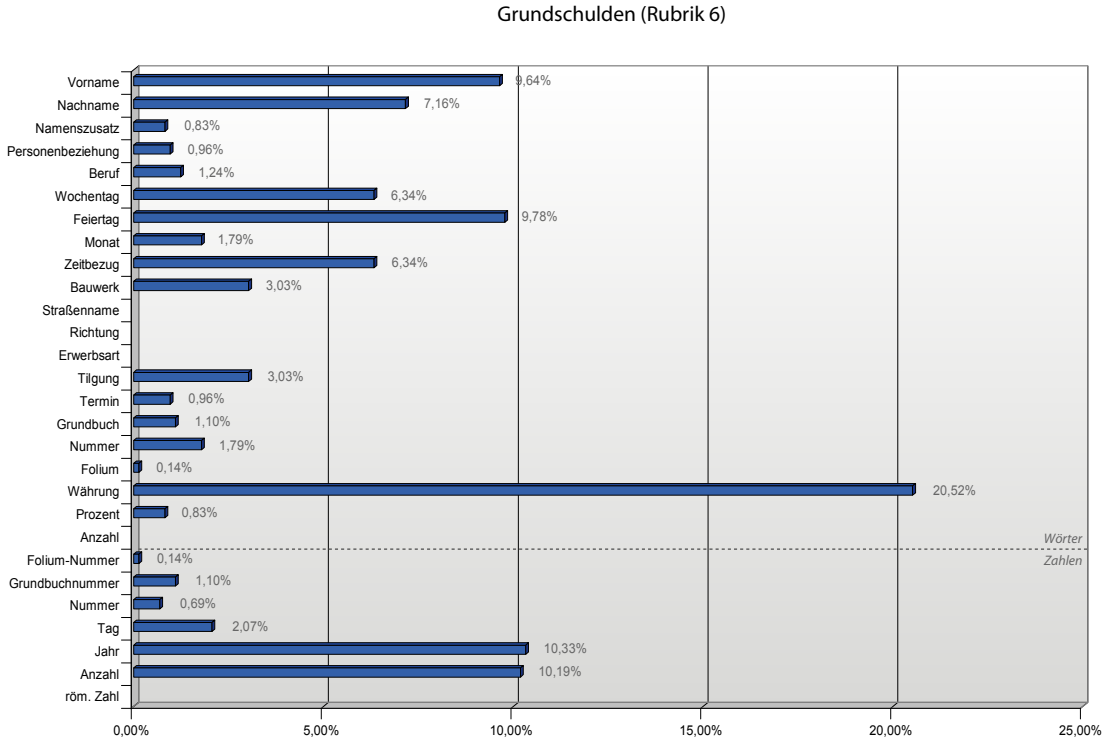


Abbildung 2.6: *Evaluierung der Informationstypen im Volltextanteil — Teil 3*

keine einheitlichen Regeln existierten, schließen zum Beispiel den Einsatz bekannter Verfahren zur Stammwortreduktion oder zum POS-Tagging aus und führen dazu, dass ein eigener Ansatz zur Identifikation der interessierenden Informationen gefunden werden muss.

Kapitel 3

Konzeption

In diesem Kapitel wird das Konzept zur Auswertung der Ausgangsdokumente vorgestellt, wobei zunächst ein Überblick über die Gesamtarchitektur gegeben wird und danach die Erläuterung der einzelnen Teilschritte erfolgt. Konkrete Beispiele zu den verschiedenen Phasen sind in Anhang E ab Seite 118 dargestellt. Nähere Details zur Umsetzung und Implementierung der Abläufe folgen im nächsten Kapitel.

3.1 Gesamtarchitektur

In Abbildung 3.1 ist ein Überblick über die Gesamtarchitektur dargestellt, deren wesentliches Hauptmerkmal die iterative Vorgehensweise bei der Auswertung der Wismarer Grundbuchdateien ist. Diese setzt die zuvor extrahierten Informationen aus einem Teil der Ausgangsdokumente zur Analyse des Inhaltes anderer Dateien ein und unterstützt damit das wörterbuchbasierte Verfahren.

Wie in der Abbildung erkennbar ist, werden die Personenverzeichnisse vor den Grundbuchdateien analysiert. Die dadurch gewonnenen Personennamen und dazugehörigen Grundbuchnummern können dann bei der Auswertung der Grundbucheinträge helfen, Namen in den Informationen zu identifizieren, und somit die Extraktion relevanter Daten unterstützen.

3.2 Umwandlung des Eingabeformats

Den ersten Schritt bei der Verarbeitung der Eingabedokumente stellt die Umwandlung der RTF- bzw. DOC-Dokumente in ein anderes, besser auswertbares Format dar. Warum diese Transformation vorgenommen werden muss und welche Zielformate für diese Phase gewählt wurden, soll in den nächsten Abschnitten erläutert werden.

3.2.1 Die Ausgangsformate DOC und RTF

MS Word Document (DOC)

Bei diesem von Microsoft Word verwendeten Dateiformat handelt es sich nicht um ein Austauschformat zwischen verschiedenen Textverarbeitungsprogrammen, sondern solche Dateien lassen sich einzig mit diesem Programm betrachten und ändern. Auch ist dieses Format weder öffentlich dokumentiert, noch lässt es sich im Klartext nachvollziehen. Das DOC-Format stellt somit ein

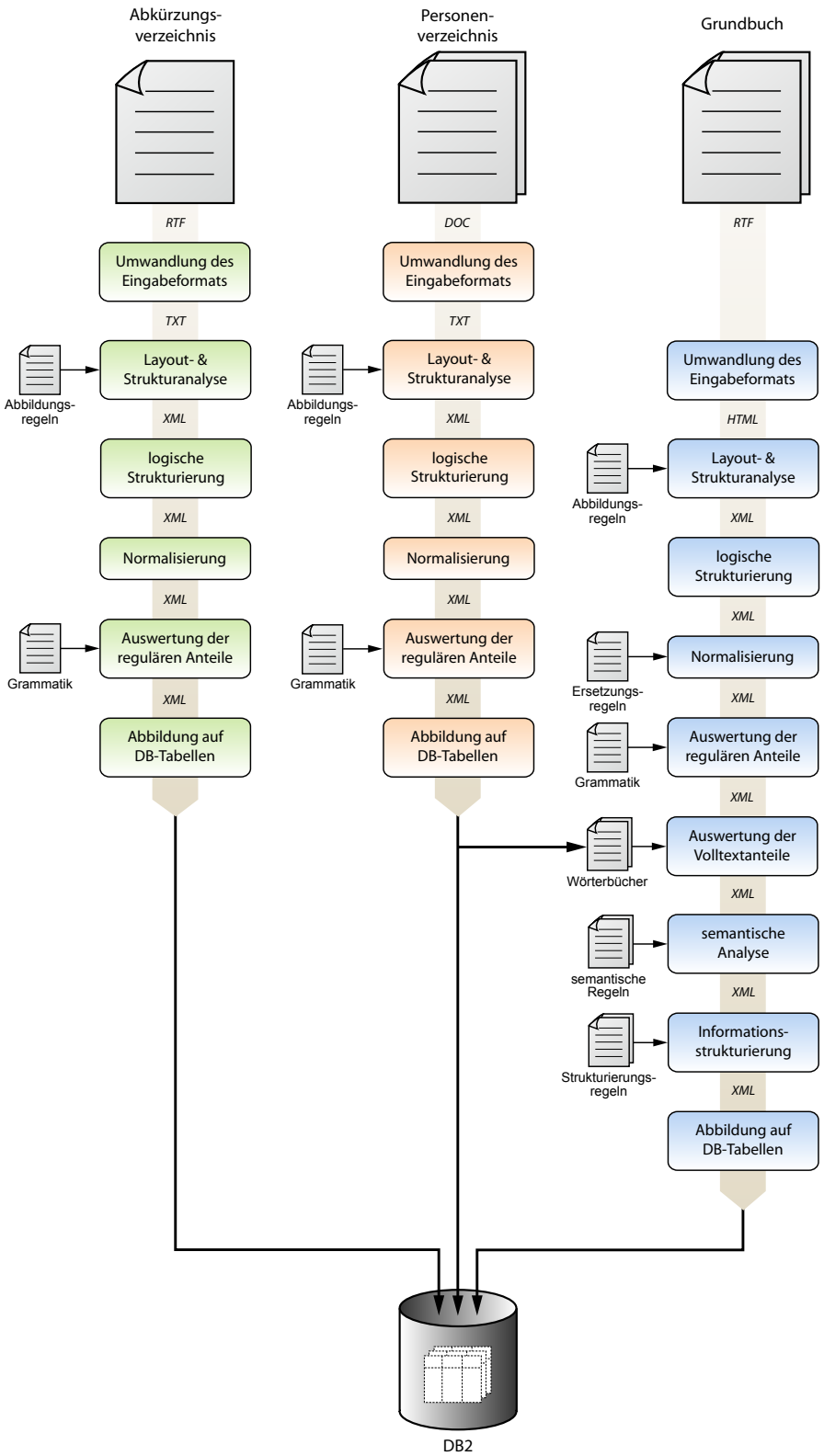


Abbildung 3.1: *Überblick über die Architektur*

proprietäres Dateiformat dar, das heißt, es entspricht keinen „offenen“ oder „freien“ Standards, die mit freier Software implementierbar sind [Wik]. Diese Eigenschaften machen es nötig, DOC-Dateien in ein anderes Format unter Erhaltung der Formatierungen zu übertragen, welches sich dann durch die Analyse des Klartextes auswerten lässt.

Rich Text Format (RTF)

Die RTF-Spezifikation beschreibt eine Methode zur Konvertierung von formatierten Texten und Graphiken zwecks einfachen Austauschs zwischen verschiedenen Anwendungen und wird somit von fast allen Textverarbeitungsprogrammen unterstützt. Dieses Format steht dabei unter der Kontrolle von Microsoft, ist aber im Unterschied zu den verschiedenen DOC-Formaten öffentlich dokumentiert [Mic99]. Danach setzt sich ein RTF-Dokument aus vier verschiedenen Angaben zusammen:

- **Kontrollworte**
  Über Kontrollworte werden Layoutangaben, Angaben zur Druckerkontrolle und andere Informationen spezifiziert. Diese genügen folgender Syntax:

  ```
  LetterSequence<Delimiter>
  ```

  `LetterSequence` ist dabei eine Zeichenfolge aus kleinen Buchstaben[12], die durch ein Begrenzungszeichen abgeschlossen wird. Dieses kann ein Leerzeichen, eine Ziffer, ein Bindestrich mit einem numerischen Parameter oder ein anderes Zeichen als ein Buchstabe und eine Ziffer sein.

- **Kontrollsymbole**
  Kontrollsymbole bezeichnen spezielle Formatierungen, zum Beispiel `\~` für ein geschütztes Leerzeichen, und beginnen wie Kontrollworte mit einem Backslash, dem ein einzelnes nichtalphabetisches Zeichen folgt. Ein Begrenzungszeichen ist in diesem Fall nicht nötig.

- **Gruppen**
  Gruppen enthalten Text und Kontrollworte oder Kontrollsymbole, die in geschweiften Klammern eingeschlossen sind. Sie spezifizieren den betroffenen Text und die darauf anzuwendenden Attribute. Das RTF-Dokument kann außerdem Gruppen für Bilder, Fußnoten, Kommentare, Kopf- und Fußzeilen, Abschnitt-, Absatz- und Zeichenformatierungen enthalten. Den RTF-Datei-Header bilden die Gruppen für schriftart-, stil-, bildschirmfarben- und dokumentformatierende Eigenschaften sowie die Gruppen für Kontrollanmerkungen und Inhaltsinformationen. Gruppen können geschachtelt werden, wobei die innere Gruppe die Eigenschaften der äußeren erbt.

- **Unformatierter Text**
  Unformatierter Text stellt den eigentlichen Inhalt dar, dessen Eigenschaften durch Kontrollworte und Gruppen beschrieben wird.

Ein Beispiel für den Aufbau eines RTF-Dokuments zeigt Abbildung 3.2. Wie man an diesem kurzen Beispiel schon sieht, kann der Quelltext für das Ausgabedokument schnell unübersichtlich und schlecht lesbar werden. Die umfangreichen Angaben zur Formatierung des Inhalts erschweren

[12] Ausnahmen gibt es bei Word 97–2000.

ebenso das Parsen des Dokuments, bei dem nicht benötigte Angaben überlesen, jedoch wichtige Formatierungsbefehle und der eigentliche Inhalt verarbeitet werden müssen. Das Erstellen eines solchen Parsers für RTF-Dokumente würde sich als sehr komplex gestalten, wobei die Anpassung dieses Parsers an sich ändernde Eingabedokumente nur schwer umzusetzen wäre. Diese Aspekte erfordern es, die Eingabedokumente aus dem RTF-Format in ein anderes, besser auswertbares Format zu überführen, wobei die wichtigen Formatierungen, jedoch in kompakterer Form, erhalten bleiben müssen.

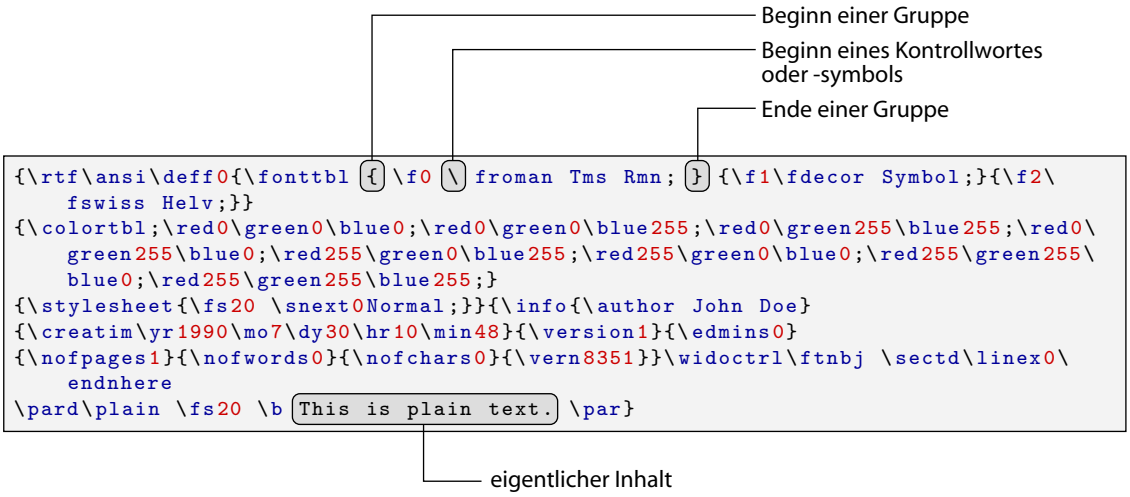


Abbildung 3.2: *Beispiel eines RTF-Dokuments im Klartext*

3.2.2 Die Zielformate TXT und HTML

Textdatei (TXT)

Unter einer Textdatei versteht man eine Datei, die nur Zeichen eines beliebigen Zeichensatzes enthält. Alle diese Zeichen sind als Inhalt der Datei und nicht als Steuerzeichen zu interpretieren. Die einzigen Steuerzeichen, die in Textdateien zum Einsatz kommen, sind die vom Zeichensatz festgelegten — etwa Tabulatoren, Zeilenvorschub oder Wagenrücklauf. Nicht enthalten sind Textformatierungen (beispielsweise Fettschrift, Kursivschrift, Unterstreichungen etc.), Seitenformatierungen (Randabstände etc.), Kopf- und Fußnoten oder Bildinformationen [Wik]. Die Dateien sind also mit simplen Texteditoren lesbar und der Klartext somit durch andere Anwendungen einfach auswertbar. Für diejenigen Ausgangsdateien, welche keine für die Bedeutung des Inhalts relevanten Formatierungen oder Layoutinformationen beinhalten, bietet sich deshalb die Transformation in einfache Textdateien an. Die Zeilenstruktur wird dabei korrekt abgebildet, was für die spätere Analyse der einzelnen Zeilen von Bedeutung sein wird.

Hypertext Markup Language (HTML)

Im Gegensatz zu RTF ist HTML kein Austauschformat für Textverarbeitungsprogramme, sondern eine standardisierte Auszeichnungssprache zur Strukturierung von Texten. Sie wurde von Tim

Berners-Lee[13] entwickelt und wird mit Hilfe von SGML[14] definiert. Inzwischen ist HTML im Zuge des Web-Booms zu einer der erfolgreichsten und verbreitesten Dateiformate geworden.

Als Auszeichnungssprache beschreibt HTML die logischen Bestandteile eines textorientierten Dokuments — wie zum Beispiel Überschriften, Absätze, Listen etc. — in einer hierarchischen Struktur. Darüber hinaus ist die Einbindung von Graphiken und multimedialen Inhalten in Form von Referenzen realisierbar. Weiterhin erlaubt das Konzept der Verweise Verknüpfungen zu beliebigen Zielen — im eigenen Projekt oder zu beliebigen Adressen im Internet innerhalb und auch außerhalb des WWW. Die gängigsten Formatierungen von Text wie beispielsweise Fettdruck, kursive Schrift oder Schriftfarben sind mit den Mitteln von HTML ebenfalls möglich, können aber durch Einbindung von Cascading Style Sheets[15] erweitert werden.

HTML ist ein Software unabhängiges Klartextformat, wodurch es möglich ist, diese Dateien mit jedem beliebigen Texteditor zu bearbeiten, ohne an ein bestimmtes kommerzielles Software-Produkt gebunden zu sein. Außerdem wird dadurch die Möglichkeit geboten, HTML-Dokumente automatisch mit Hilfe von Programmen, zum Beispiel durch CGI[16]-Skripte, zu generieren.

Aufgrund ihrer Einfachheit kann man HTML als eine so genannte „lingua franca“ — eine Brot- und Butter-Sprache — bezeichnen, die jeder kennt, spricht und braucht und die leicht zu erlernen ist. Im Vergleich zu RTF wirkt diese Sprache daher verständlicher, denn der Quelltext ist im Gegensatz zum RTF-Klartext kompakter, lesbarer und somit auch einfacher anpassbar. [Mue01]

Da der Aspekt der Kompaktheit gegenüber der Austauschbarkeit zwischen verschiedenen Textverarbeitungsprogrammen für die Auswertung des Layouts der Eingabedokumente im Vordergrund steht und sich fast alle RTF-Schriftformatierungen auf HTML-Strukturen abbilden lassen, wurde HTML als ein Zwischenformat in dieser Arbeit gewählt.

3.2.3 Gewählte Zielformate für die einzelnen Dateien

Aufgrund der oben angesprochenen Nachteile der Eingabeformate für die Analyse der Formatierungen und Layoutinformationen wurden in dieser ersten Phase des Ablaufs die Ausgangsdateien entweder in das Zielformat HTML oder in reine Textdateien umgewandelt:

- Das **Abkürzungsverzeichnis**, das als Teil der Einleitung im RTF-Format vorlag, wurde manuell dort herauskopiert und als reine Textdatei abgespeichert, da dieses Verzeichnis keinerlei Schriftformatierungen enthielt.

- Die einzelnen **Personenverzeichnisse** lagen als DOC-Dateien vor und enthielten als Formatierung lediglich den Fettdruck des Anfangsbuchstabens der behandelten Nachnamen in der jeweiligen Datei. Da diese Überschrift in jeder dieser Dateien jeweils die erste nichtleere Zeile

[13] Sir Timothy J. Berners-Lee (*8. Juni 1955 in London) gilt als Begründer des World Wide Webs, da er 1989 am CERN zusammen mit URL und HTML das Protokoll HTTP erfunden hat. [Wik]

[14] SGML (Abkürzung für *Standard Generalized Markup Language*) ist eine Metasprache, mit deren Hilfe man verschiedene Auszeichnungssprachen für Dokumente definieren kann. SGML ist ein ISO-Standard: „ISO 8879:1986 Information processing — Text and office systems — Standard Generalized Markup Language (SGML)“. [Wik]

[15] Cascading Style Sheets (CSS) ist eine deklarative Stylesheet-Sprache für strukturierte Dokumente (zum Beispiel HTML und XML). Durch die Trennung von Stil und Inhalt wird das Veröffentlichen und Betreuen von Web-Seiten vereinfacht. [Wik]

[16] Die CGI-Schnittstelle (Abkürzung für *Common Gateway Interface*) ist eine Möglichkeit, Programme oder Skripte im Web bereitzustellen, die von HTML-Dateien aus aufgerufen werden können, und die selbst HTML-Code erzeugen und an einen Web-Browser senden können. [Mue01]

darstellt, lässt sich ebenso diese Eigenschaft anstatt der Formatierung für die Analyse ausnutzen. Aus diesem Grund wurden die Personenverzeichnisse als Textdateien abgespeichert und die Formatierungen der Überschriften somit verworfen. Dadurch wird die im nächsten Schritt folgende Analyse der Struktur zusätzlich vereinfacht.

- Im Gegensatz zu den vorangegangenen Dokumenten beinhalteten die RTF-**Grundbuchdateien** mehrere relevante Formatierungen, anhand derer die Bedeutung der Zeileninformationen ermittelt werden kann. Aus diesem Grund wurde hier HTML als Zielformat gewählt, wodurch die Schriftformatierungen erhalten blieben.

3.3 Layout- und Strukturanalyse

Dieser zweite Schritt nach der Umwandlung des Eingabeformats stellt die wesentliche Eigenschaft des Structure Mining dar, denn in dieser Phase werden nun Struktur- und Layoutinformationen des betreffenden Dokuments zur Interpretation der Daten herangezogen. Genauer gesagt bedeutet dies, dass den Daten des Dokuments eine erste abstrakte Beschreibung ihrer Bedeutung zugeordnet wird, welche bei der späteren Inhaltsanalyse die Extraktion der Daten in folgenden Aspekten unterstützt:

- **Wahl des geeigneten Extraktionsverfahrens**
  Ist erst einmal die Bedeutung eines Abschnittes evaluiert und bekannt, ob dessen Inhalt zu den strukturierten oder Volltextanteilen zählt, kann im nächsten Schritt des Verarbeitungsprozesses die entsprechende Extraktionsmethode darauf angewandt werden. So sind zum Beispiel die Angaben zur Rubrik 1 in den Grundbucheinträgen regulär aufgebaut und können durch eine Grammatik beschrieben und ausgewertet werden. Im Gegensatz dazu stellen beispielsweise die Eigentümerangaben Volltext dar, der nur durch den Einsatz von wörterbuchbasierten Verfahren und nicht durch Grammatiken analysiert werden kann.

- **Kontextbezogene Informationsextraktion**
  Grundsätzlich können zwei verschiedene Datenmengen mit unterschiedlichen Bedeutungen den gleichen grammatikalischen Aufbau haben und sich nur durch das Layout voneinander abheben. In diesen Fall ist eine vorangehende Interpretation des Layouts vor der Inhaltsanalyse notwendig, um die extrahierten Informationen in den richtigen Kontext zu bringen.

- **Trennung der Aspekte Layout/Struktur und Inhalt**
  Generell kann sich eine Trennung der Analyse der verschiedenen Aspekte eines Dokuments nur vorteilhaft auf den Gesamtprozess auswirken. Einerseits ist damit die modulare Verarbeitung der Eingabedokumente in mehreren, konzeptionell voneinander getrennten Schritten realisiert, andererseits sind die einzelnen Komponenten des Prozesses dadurch einfacher anpassbar bzw. austauschbar. Natürlich wäre die Interpretation des Layouts und die direkt daran anschließende Inhaltsanalyse jeweils eines Dokumentausschnitts möglich. Die Verbindung dieser Konzepte in einem Schritt würde allerdings die Verständlichkeit des Konzepts und der Implementierung beeinträchtigen.

3.3.1 Merkmale für die Layout- bzw. Strukturanalyse

Zur Bestimmung der Bedeutung der Dokumentdaten und deren Kategorisierung kommen grundsätzlich zwei Ansätze zur Anwendung:

- Die **Layoutanalyse** betrachtet Formatierungen, die durch die jeweiligen HTML-Tags zugänglich sind. Dabei werden vor allem fette und kursive Schriftformatierungen des Inhaltes ausgewertet.
- In der **Strukturanalyse** wird der Inhalt auf bestimmte Kennzeichen und Muster hin untersucht und so die Bedeutung ermittelt. Zu den Kennzeichen gehören unter anderem bestimmte einleitende Zeichen am Zeilenanfang und Zeichen, die Daten zu unterschiedlichen Bedeutungen voneinander abgrenzen, wie beispielsweise der Doppelpunkt. Außerdem kann der Inhalt einer Zeile in Bezug auf die Zugehörigkeit zu einer Bedeutungskategorie mit Hilfe eines regulären Ausdrucks überprüft werden.

Grundlegend für die Kategorisierung der Dokumentdaten ist der absatzweise Aufbau der Ausgangsdateien. Das bedeutet, dass der Inhalt eines Absatzes — beendet jeweils durch einen Zeilenumbruch — Informationen zu einer bestimmten Kategorie beinhaltet. So sind zum Beispiel in den Grundbucheinträgen die Angaben zu verschiedenen Eigentümern mit den dazugehörigen Daten in eigene Absätze gekapselt. Genauso befinden sich die Angaben zu den Personen im Personenverzeichnis in jeweils separaten Absätzen. Ausgenommen davon sind natürlich Ausnahmen durch Eingabefehler in den Ausgangsdokumenten, in denen der Nutzer fälschlicherweise Umbrüche eingeführt oder vergessen hat.

Weiterhin muss in einigen Fällen der Bezug eines Absatzes zum vorangegangenen berücksichtigt werden. So ist im Grundbuch der Inhalt der Rubrik „Eigentümer“ in der Regel in mehrere Absätze aufgeteilt, jedoch nur der erste Abschnitt der Rubrik enthält die kennzeichnende kursive Rubrikzahl am Anfang der Zeile. Die für die Analyse der einzelnen Dokumente herangezogenen Merkmale sind in Abbildung 3.3 bis 3.5 ersichtlich.

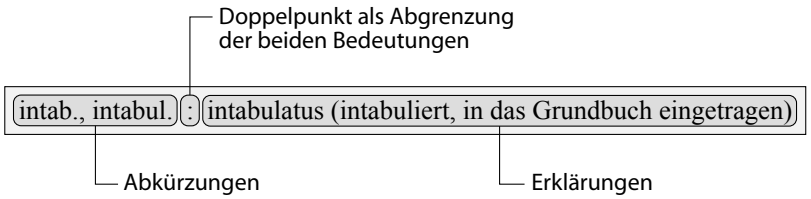


Abbildung 3.3: *Merkmale des Abkürzungsverzeichnisses*

3.3.2 XML als Zielformat

XML[17] ist ein vom W3C[18] entwickeltes Dokumentenformat zur Darstellung von strukturierten und semistrukturierten Daten [KM03]. Allgemeiner ausgedrückt ist XML eine Definintionssprache von Auszeichnungssprachen und eine vereinfachte Untermenge von SGML, das seit 1986 international standardisiert ist [Mue01].

Wie der Name bereits deutlich macht, ist XML eine Sprache zur Definition von anwendungsspezifischen Datenstrukturen und deren semantische Darstellung. Durch so genannte Tags

[17] Abkürzung für *Extensible Markup Language*.

[18] Abkürzung für *World Wide Web Consortium*. Das W3C wurde gegründet, um alle Möglichkeiten des Web zu erschließen. Dazu werden einheitliche Technologien (Spezifikationen, Richtlinien, Software und Tools) entwickelt, die den Fortschritt des Webs fördern und seine Interoperabilität sicherstellen. [W3C]

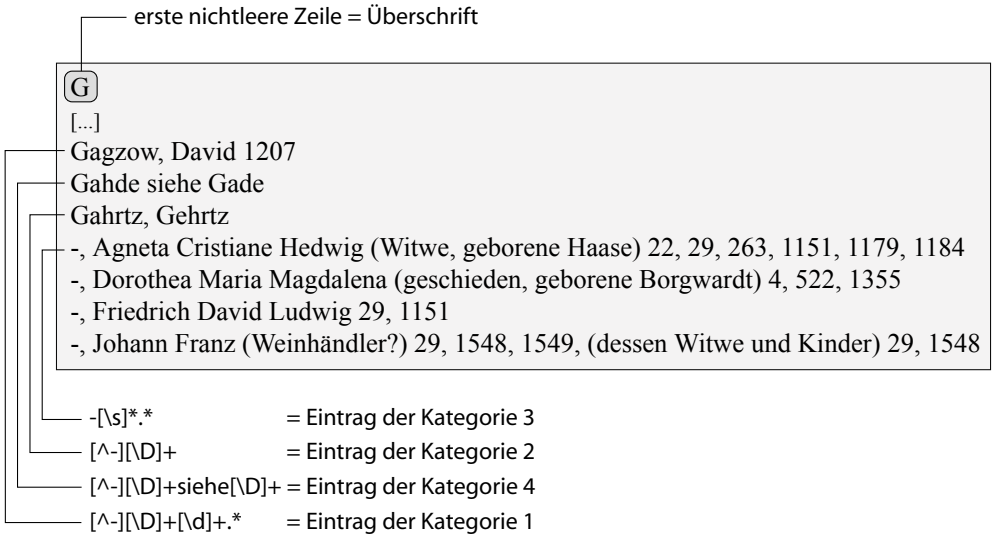


Abbildung 3.4: *Merkmale der Personenindexe*

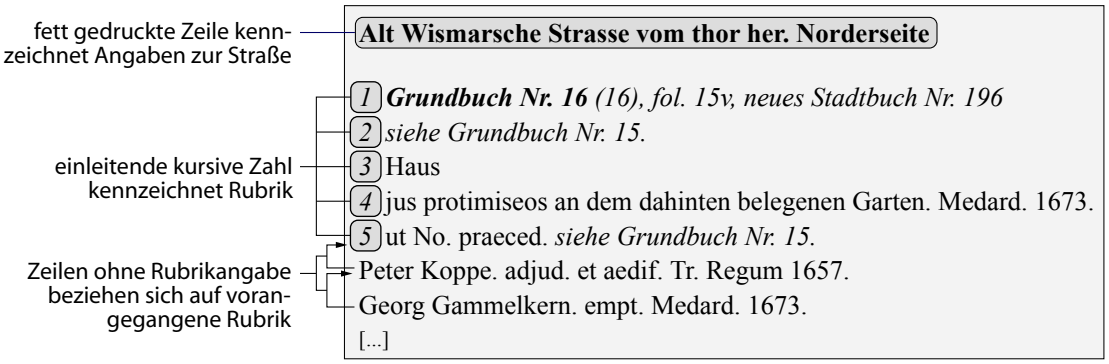


Abbildung 3.5: *Merkmale der Grundbücher*

wird der Inhalt markiert und ihm auf diese Weise logische Auszeichnungen zugeordnet. Mit XML kann jedoch nicht definiert werden, wie eine interpretierende Software den Inhalt bei der Darstellung formatieren soll. Für diesen Zweck ist die Anwendung einer ergänzenden Stylesheet-Sprache notwendig. Im Gegensatz dazu wird in HTML nichts über die Bedeutung des Inhalts ausgedrückt, sondern nur Formatierungsangaben spezifiziert. Der Vorteil von XML gegenüber HTML ist deshalb weiterhin der Aspekt, dass die Daten völlig unabhängig vom Ausgabemedium definiert werden können. [Mue01]

Da XML-Dokumente die Daten ansich und Informationen über diese Daten enthalten, nennt man sie auch selbstbeschreibend, denn sie beschreiben ihre Struktur selbst. Die Dokumente sind daher gut lesbar, sowohl für die Verarbeitung durch Anwendungen als auch für den Menschen. Somit ist XML hervorragend als Austauschformat für verschiedenste Arten von Informationen und

auch als Zwischenformat zur Präsentation von Daten durch Stylesheets[19] geeignet [KM03], was auch in dieser Arbeit zu der Entscheidung für XML führte.

3.3.3 Ablauf der Analyse und XML-Generierung

Der generelle Ablauf

Einen Überblick über den Prozess der Verarbeitung der Eingabedokumente und der Generierung des XML-Ausgabeformats gibt Abbildung 3.6. Demnach wird jeweils eine Zeile der Eingabedatei betrachtet und deren Informationstyp anhand der Eigenschaften — beispielsweise enthaltene Schriftformatierungen oder Erfüllung eines regulären Ausdrucks — ermittelt. Konnte der Typ der Informationen bestimmt werden, so wird im Ausgabedokument ein XML-Element mit dem entsprechenden Tag und dem Inhalt der jeweiligen Zeile generiert. Ist dies nicht der Fall, wird kein XML-Element angelegt und der Zeileninhalt verworfen. Für eine manuelle Nachbearbeitung durch den Nutzer wird in diesem Fall ein Eintrag mit der aktuellen Zeilennummer und dem Inhalt der betreffenden Zeile ins Logbuch aufgenommen.

Besonderheiten der Eingabedokumente

Dieser Ablauf stellt nur einen groben Überblick über die generelle Transformation dar. Für die drei verschiedenen Dokumentarten Abkürzungsverzeichnis, Personenverzeichnis und Grundbuch muss jeweils ein eigens angepasster Abbildungsprozess erstellt werden, da die unterschiedlichen logischen Aufteilungen der Dateien verschiedene Verarbeitungen bedingen:

- **Abkürzungsverzeichnis**
  Hier wird nicht jeweils eine ganze Zeile eingelesen, sondern nur die Zeichen bis zum Doppelpunkt. Dieser Inhalt stellt dann die Abkürzungen dar, während der Rest der Zeile, der im nächsten Schritt betrachtet wird, als Erläuterung zur aktuellen Abkürzung gedeutet wird.

- **Personenverzeichnis**
  Hierbei muss beachtet werden, dass die Zeilen mit den Angaben zu einer Person logisch zu der letzten vorangegangenen Zeile mit Angaben zum aktuellen Familiennamen gehören. Ging der Personenangabe ein Eintrag voraus, der nicht erkannt wurde, so muss auch diese Zeile verworfen werden. Diese Maßnahme verhindert, dass eine Personenangabe einem früheren Familiennamen zugeordnet wird, der logisch jedoch in keinem Bezug zu dieser Person steht.

- **Grundbuch**
  Da es sich bei den Eingabedokumenten der Grundbücher in diesem Schritt um HTML-Dateien handelt, müssen vor der Generierung des XML-Elementes zu einer analysierten Zeile alle HTML-Tags entfernt werden.

[19] Ein Stylesheet verwendet man, um die Form der Darstellung von den Inhalten eines strukturierten Dokuments (zum Beispiel HTML oder XML) zu trennen. Stylesheets umfassen dabei alle Bereiche der Interpretation (bildliche, hörbare, oder fühlbare Darstellung). Stylesheets erlauben es somit, Inhalte abhängig von dem Ausgabegerät (zum Beispiel auch Braille-Lesegeräte für Blinde) zu interpretieren. [Wik]

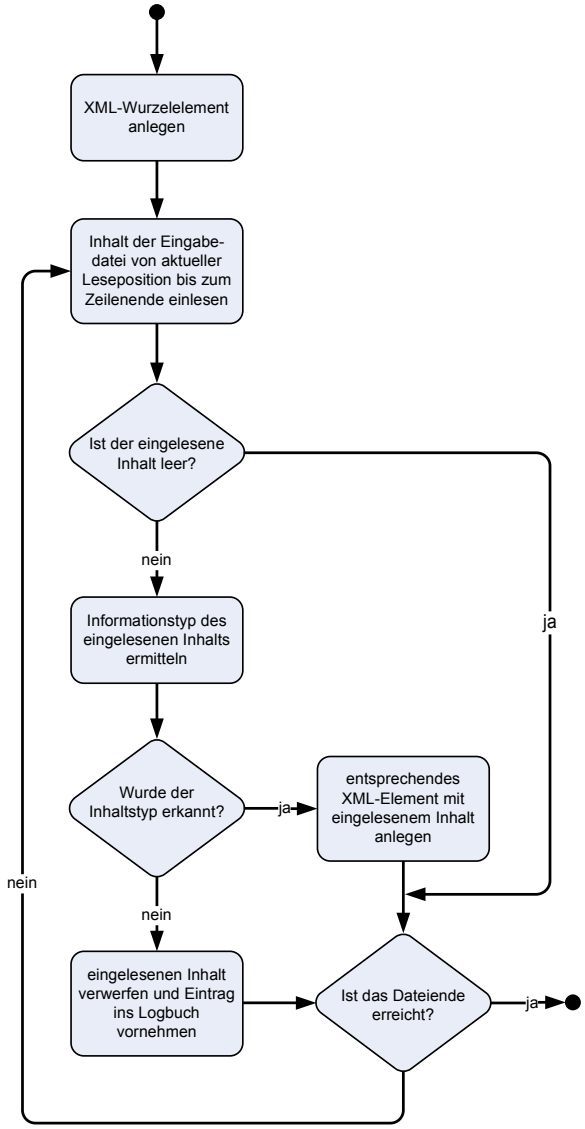


Abbildung 3.6: *Genereller Ablauf der Layoutanalyse*

Bezug zum Originaltext

Für spätere Überprüfungen, ob der Originaltext richtig interpretiert wurde, und zur Unterstützung der manuellen Nachbearbeitung bietet es sich an, einen Bezug der im Verlauf des Verarbeitungsprozesses modifizierten Eingabedaten zum Originalinhalt herzustellen. Für diesen Zweck wurden zwei Ansätze realisiert, deren konkrete Ausbildung im laufenden Beispiel in Anhang E.2 auf Seite 119 ersichtlich ist:

- Zum einen wird zu jedem generierten XML-Dokument eine **Referenz zum Originaldokument** hergestellt, indem der Name und der Speicherort der Ausgangsdatei[20] in Form von Attributen der Dokumentwurzel hinzugefügt werden. Dadurch ist es zu jedem späteren Zeitpunkt möglich, die Ausgangsdateien zu lokalisieren und zusätzlich zu den extrahierten Daten in der Datenbank abzulegen. Weiterhin wird jedem generierten XML-Element mit dem eingelesenen Inhalt die entsprechende **Zeilennummer im Eingabedokument** hinzugefügt, sodass ein genauer Bezug zum Originaldokument hergestellt werden kann. Voraussetzung für diesen Ansatz ist jedoch, dass die Ausgangsdateien vom Nutzer nicht manipuliert werden.

- Zum anderen wird jedem generierten XML-Element ein Element zugeordnet, das noch einmal den eingelesenen Inhalt beinhaltet, welcher im Verlauf des weiteren Prozesses jedoch unberührt bleibt. Dieser **Originalinhalt** wird dann zum Schluss ebenfalls mit in die Datenbank übernommen. Im Fall der Grundbuchdateien, die nicht aus dem TXT-, sondern HTML-Format in XML-Dokumente übertragen werden, wird zusätzlich ein Element mit dem eingelesenen Inhalt inklusive der HTML-Tags übernommen, sodass zum einen der formatierte ursprüngliche Inhalt und der unformatierte Inhalt ohne HTML-Tags zur Verfügung steht. Der Vorteil bei der Erhaltung der Formatierungen besteht darin, dass in den Originalgrundbucheinträgen einige Ergänzungen vom Herausgeber in kursiver Schrift gemacht wurden, welche im Verlauf dieser Arbeit jedoch nicht weiter differenziert wurden. Später ist man durch den Bezug auf die formatierte Originalzeile jedoch in der Lage, diese Ergänzungen wieder als solche zu erkennnen.

Die Vorteile des ersten Ansatzes zielen auf spätere Anwendungen ab, die sich auf die extrahierten Daten beziehen. Vorstellbar wäre zum Beispiel die Anzeige der Originaldatei in einem separaten Fenster bei der Auswahl eines Datensatzes in einer Anwendungsmaske. In diesem Fenster könnte dann diejenige Zeile, aus welcher dieser Datensatz stammt, automatisch selektiert werden. In der Ansicht der Originaldatei kann der Anwender zudem beliebig navigieren und die „Umgebung“ des Datensatzes betrachten.

Der Vorteil der zweiten Variante ist der direkte Zugriff auf den Quellinhalt. Wenn ein extrahierter Datensatz in einer späteren Anwendung mit dem Original verglichen werden soll, ohne dass die vorherigen und nachfolgenden Informationen von Interesse sind, muss nicht die gesamte Originaldatei aus der Datenbank geladen und zu der entsprechenden Zeile navigiert werden. Außerdem lassen sich die in den nächsten Phasen entstehenden XML-Dokumente besser nachvollziehen, da man in ihnen den direkten Vergleich zur Quelle hat.

[20] Damit sind diejenigen Dateien gemeint, die in der ersten Phase des Prozesses generiert wurden.

3.4 Logische Strukturierung

Die Ergebnisdokumente aus dem vorangegangenen Prozess in eine XML-Datei sind zunächst flach strukturiert. Da diese Struktur jedoch nicht dem logischen Aufbau der Ausgangsdokumente genügt, folgt in diesem Schritt die Zusammenfassung von logisch zusammengehörigen Inhalten in eine Einheit, indem die XML-Elemente entsprechend geschachtelt werden. Diese logische Gruppierung von Informationen unterstützt später ebenfalls die korrekte Abbildung der Daten auf die Datenbankstruktur.

3.5 Normalisierung

Auf den XML-Dokumenten, die in der vorangegangenen Phase generiert wurden, wird nun eine Normalisierung durchgeführt. Dieser Prozess, bei dem bestimmte Anteile des Inhaltes auf eine einheitliche Form gebracht werden, lässt sich in mehrere Abschnitte unterteilen. Diese sind in Abbildung 3.7 dargestellt und werden im Folgenden näher erläutert.

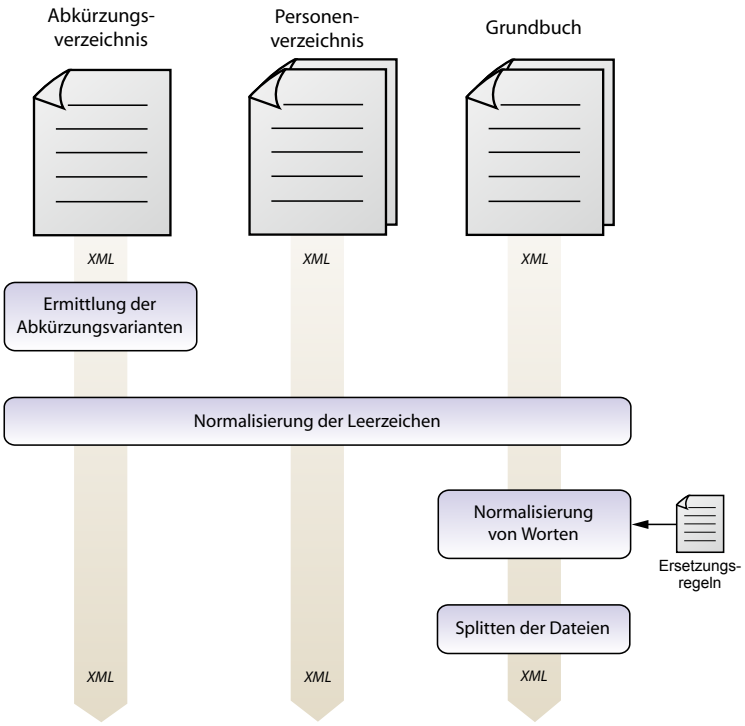


Abbildung 3.7: *Ablauf der Normalisierung*

3.5.1 Ermittlung der Abkürzungsvarianten bei Abkürzungen von Wortgruppen

Wie in Kapitel 2.1 auf Seite 14 bereits erwähnt, wurden in einigen Fällen im Abkürzungsverzeichnis ganze Wortgruppen abgekürzt. Die möglichen Varianten dazu wurden nicht direkt im Verzeichnis

angegeben, sondern in einer speziellen Notation aufgeführt, bei der die Abkürzungsvarianten der einzelnen Worte hintereinander aufgezählt werden. Die Kurzformen zu einem Wort der Gruppe sind dabei durch Kommas voneinander getrennt. Die Abkürzungen zu einem angrenzenden Wort sind durch ein Leerzeichen, das kein Komma als Vorgänger hat, erkennbar. Das Vorgehen bei der Ermittlung aller möglichen Varianten zu einer Wortgruppe ist in Abbildung 3.8 dargestellt.

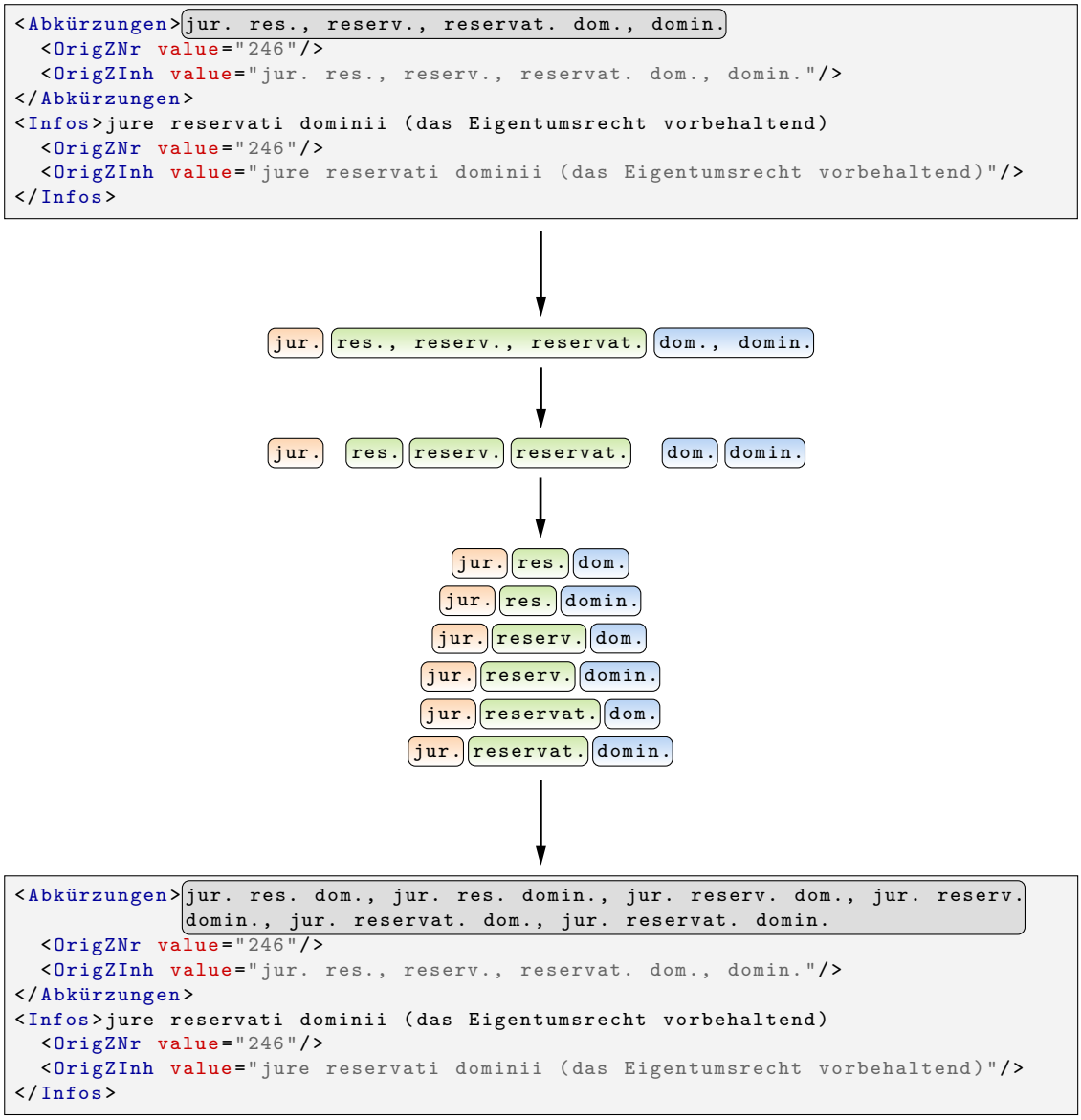


Abbildung 3.8: *Ermittlung aller Abkürzungsvarianten*

Bei diesem Algorithmus gibt es jedoch einige Fälle, in denen die angegebenen Abkürzungen falsch interpretiert und zu ungültigen Kombinationen zusammengesetzt werden. Die Abkürzungen „Justitzr., Just. Rat“ werden durch den Algorithmus zum Beispiel durch die Kombinationen „Justitzr. Rat, Just. Rat“ ersetzt. Die Ursache liegt in der für zwei verschiedene Interpretationsmöglichkeiten

der Abkürzungen gewählten Notation begründet. Diese Fälle können nicht automatisch erkannt, sondern müssen hinterher manuell überprüft und eventuell berichtigt werden. Um dies zu unterstützen, wird für jede Kombinationsermittlung ein Eintrag in das Logbuch vorgenommen, der zum Vergleich die Ausgangsabkürzung und die generierten Kombinationen enthält.

3.5.2 Normalisierung der Leerzeichen

Um eventuelle Eingabefehler in Bezug auf die Angabe von Leerzeichen auszugleichen und unnötige Leerzeichen zu entfernen, werden die Elementinhalte der generierten XML-Dokumente einem Normalisierungsprozess unterzogen, der folgende Korrekturoperationen beinhaltet:

- Leerzeichen an Anfang und Ende des Elementinhaltes werden abgeschnitten.
- Leerzeichen vor Satzzeichen werden entfernt.
- Leerzeichen nach öffnenden Klammern werden entfernt.
- Leerzeichen vor schließenden Klammern werden entfernt.
- Mehrere aufeinander folgende Leerzeichen werden durch eines ersetzt.
- Leerzeichen zwischen Satzzeichen werden entfernt.

Als Satzzeichen gelten dabei `,:;!.?`. Als Klammern werden `(){}` betrachtet. Leerzeichen vor öffnenden oder nach schließenden Klammern einzufügen ist nicht sinnvoll, denn es existieren Fälle wie zum Beispiel „Brand(a)nus“, bei denen diese Korrektur zu einer Verfremdung des Inhaltes führen würde.

Dieser Schritt der Normalisierung wird durchgeführt, um die Regeln für die regulären Anteile, die später durch eine Grammatik ausgewertet werden sollen, zu vereinfachen. Werden nämlich in diesem Schritt schon die oben aufgeführten Konventionen für den gesamten textuellen Inhalt durchgesetzt, entfällt die Berücksichtigung etwaiger Abweichungen in den Grammatikregeln.

3.5.3 Normalisierung von einzelnen Worten

Ein Problem bei der späteren Analyse der Texte mit Hilfe von Wörterbüchern ist die Tatsache, dass zu einem Wort mehrere Varianten der Schreibung bzw. unterschiedliche Flexionsformen existieren. Im Falle der Abkürzungen gibt es zu jeder ausgeschriebenen Form ebenfalls mehrere Kurzworte, die im Text auftauchen können. Bei der späteren Analyse der einzelnen Token müssten aus diesem Grund entweder alle möglichen Varianten zu einem Wort im entsprechenden Wörterbuch aufgeführt oder der Schwellenwert für den Wortabstand genügend groß gewählt werden[21], um die Token identifizieren zu können. Letzteres birgt jedoch die Gefahr, dass Token und Wörterbuchbegriff als übereinstimmend erkannt werden, diese jedoch von vollkommen unterschiedlicher Bedeutung sind.

Aus diesem Grund wird dem Nutzer in Form einer externen Datei die Möglichkeit gegeben, Worte und deren Ersetzung zu definieren. Die Struktur eines Eintrags in dieser Datei lautet wie folgt:

[21] Detaillierte Erläuterungen zum wörterbuchbasierten Verfahren und der Wortabstandsberechnung folgen ab Kapitel 3.7.4 auf Seite 46.

`Wort # Ersetzung`,

wobei das zu ersetzende Wort und der Ersetzungsterm aus mehreren Worten zusammengesetzt sein können. Für die Anwendung dieser Regeln auf den Textinhalt der XML-Dateien kann vom Nutzer definiert werden, ob die Unterscheidung zwischen Groß- und Kleinschreibung relevant ist und ob eine Normalisierung der Rechtschreibung vorgenommen werden soll[22].

Die Ersetzung einer Zeichenkette erfolgt schrittweise anhand dieser Regeln: Zuerst wird die erste aufgeführte Regel auf die Zeichenkette angewandt, indem alle Vorkommen des angegebenen Begriffes in der Zeichenkette durch das Substitut ersetzt werden. Danach werden die folgenden Regeln nacheinander auf die jeweilige Ergebniszeichenkette der vorangegangenen Regel angewandt. Der genaue Ersetzungsalgorithmus lautet dabei wie folgt (siehe auch Abbildung 3.9):

1. Die aktuelle Regeldefinition wird in Bedingung und Substitut aufgesplittet.
2. Der Bedingungsteil der Regel und die zu überprüfende Zeichenkette (ein Textknoten aus dem XML-Dokument) werden anhand von Leer- und Satzzeichen in Token zerlegt.
3. Von der Tokenabfolge der Bedingung und der Zeichenkette werden jeweils Kopien angelegt.
4. Wenn der Nutzer angegeben hat, dass bei dem Vergleich des Textes mit den Ersetzungsregeln keine Unterscheidung zwischen Groß- und Kleinschreibung vorgenommen und/oder die Rechtschreibung normalisiert werden soll, dann werden die einzelnen Token der Bedingung und der Zeichenkette in Kleinschreibung umgewandelt und/oder deren Schreibung normalisiert. Diese Modifikationen an den Token werden auf der Kopie der Tokenabfolge durchgeführt.
5. Die Tokenabfolge (Kopie) der Zeichenkette wird daraufhin überprüft, ob die Tokenabfolge (Kopie) des Bedingungsteils der Regel darin enthalten ist. Alle gefundenen aufeinander folgenden Token der Bedingung werden in der Zeichenkette wie folgt ersetzt: Das erste Token der Bedingung wird durch das Substitut der Regel ersetzt. Alle darauf folgenden Token der Bedingung werden aus der Zeichenkette gelöscht. Diese Modifikationen werden auf dem Original der Tokenabfolge der Zeichenkette vorgenommen.
6. Sind alle Vorkommen der Bedingungstoken in der Zeichenkette gemäß Schritt 5 ersetzt, wird das Ergebnis der Regelanwendung konstruiert, indem die Originaltokenabfolge der Zeichenkette wieder zu einem String zusammengesetzt wird. Dieses Resultat ist dann der Ausgangspunkt für die Anwendung der nächsten Regeldefinition.

Bei der Festlegung der Regeln muss aufgrund dieses Algorithmus Folgendes bedacht werden:

- Es sollten nur Ersetzungen für Worte definiert werden, die eindeutig bestimmt werden können. So macht es zum Beispiel keinen Sinn, die Abkürzung „d.“ durch die im Abkürzungsverzeichnis nachzulesende ausgeschriebene Form „deletum“ ersetzen zu lassen, da „d.“ ebenfalls für „dies“ und „denarius“ eine Abkürzungsmöglichkeit darstellt. Durch so eine Ersetzung würde die Bedeutung des Inhaltes an vielen Stellen verändert werden.
  Diese Mehrdeutigkeiten insbesondere bei den Abkürzungsvarianten führt dazu, dass eine automatische Ersetzung aller Varianten einer Abkürzung — die ja durch die Auswertung des

[22] Der Normalisierungsprozess der Rechtschreibung wird in Kapitel 3.7.9 auf Seite 54 erläutert.

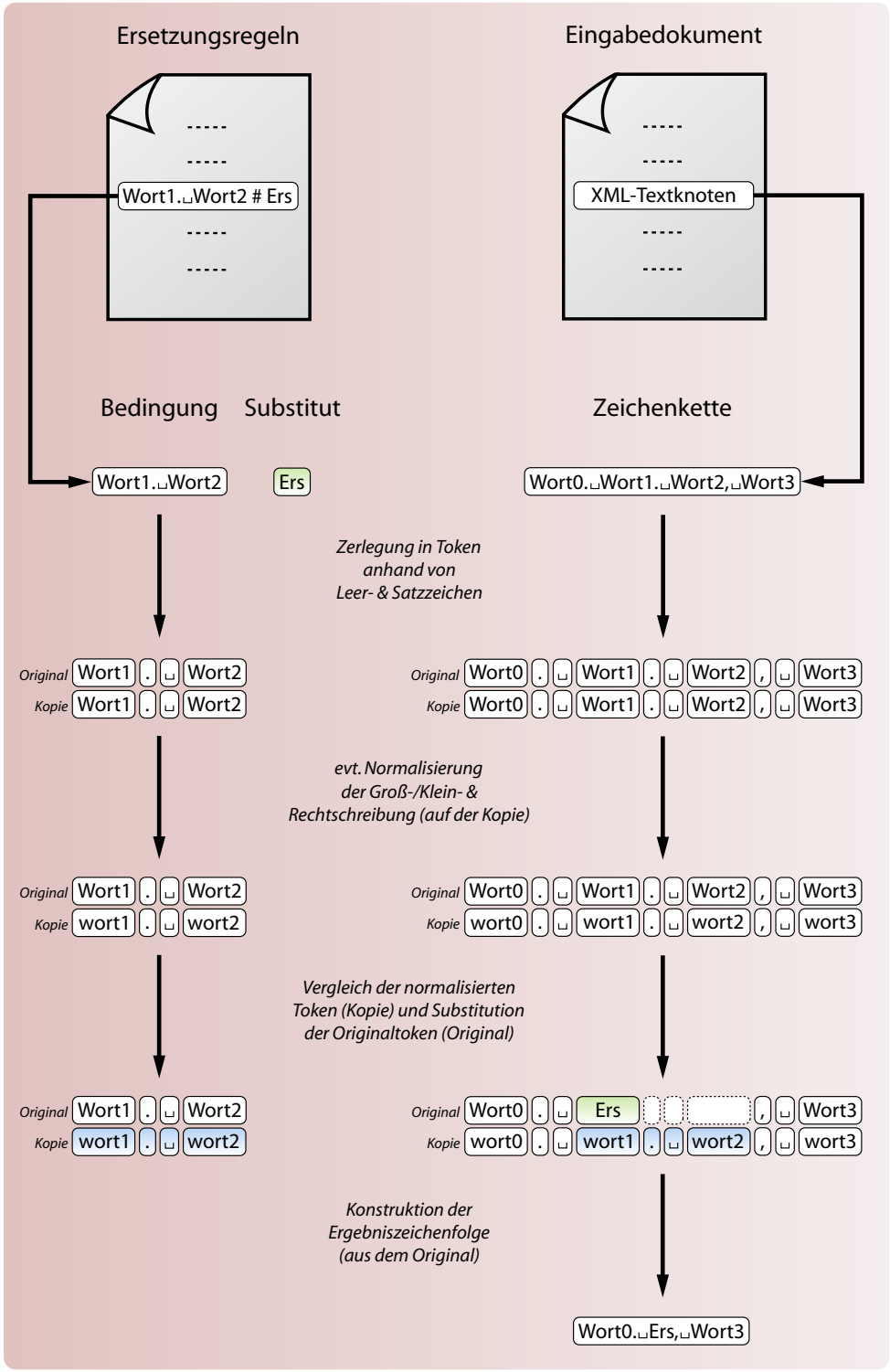


Abbildung 3.9: *Algorithmus der Wortnormalisierung*

Abkürzungsverzeichnisses bekannt sind — ohne komplexe semantische Betrachtung des umgebenden Inhaltes unmöglich umzusetzen ist.

- Der Reihenfolge, in der die Ersetzungsregeln in der entsprechenden Datei aufgeführt sind, kommt eine wichtige Bedeutung zu. Die Anordnung der Regeln bestimmt das generierte Ergebnis und stellt damit sozusagen die Prioritätenverteilung der Ersetzungen dar. Ein Beispiel dieser Problematik ist in Abbildung 3.10 schematisch erklärt.

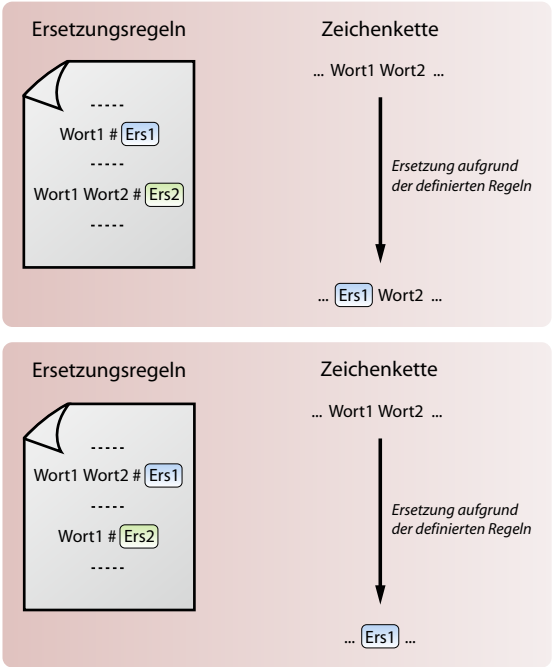


Abbildung 3.10: *Priorisierung der Ersetzungsregeln*

- Die Definition der Regeln kann zu kaskadierenden Ersetzungen führen. Das bedeutet, dass ein bereits in der Zeichenkette durch eine vorangegangene Regel ersetzter Term durch eine nachfolgende Regel weiter verändert wird (siehe Abbildung 3.11). Dies kann unter Umständen erwünscht sein, muss aber vom Nutzer bei der Erstellung der Regeln berücksichtigt werden.

Obwohl der Nutzer bei der Erstellung und Anordnung der Ersetzungsregeln die oben genannten Aspekte berücksichtigen muss, sind die Vorteile der Durchführung dieses Normalisierungsprozesses nicht von der Hand zu weisen. Wie bereits erwähnt, reduzieren sich zum einen damit die Einträge in den später anzuwendenden Wörterbüchern, zum anderen sind auch die letztlich in der Datenbank abgespeicherten Informationen am Ende des Verarbeitungsprozesses der Eingabedokumente auf die wesentlichen Grundwörter reduziert, was wiederum in vereinfachten Anfragen resultiert.

Dieser Ersetzungsprozess wird im Laufe der Normalisierung nur auf diejenigen Inhalte der Grundbuchdateien beschränkt, die nicht durch grammatikalische Regeln, sondern mit Hilfe der Wörterbücher ausgewertet werden. Auch die Anwendung der Ersetzungsregeln auf das Abkürzungs- und Personenverzeichnis ist nicht angebracht, da diese von vornherein vom Herausgeber entsprechend angelegt wurden und sie kaum Volltext mit vielen Flexionsformen der auftretenden Worte enthalten.

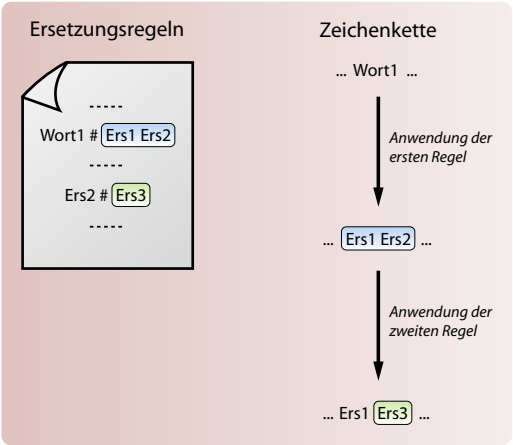


Abbildung 3.11: *Kaskadierende Ersetzungen*

3.5.4 Splitten der Grundbuchdateien

Die einzelnen Grundbuchdateien sind inhaltlich sehr umfangreich, was sich besonders in den folgenden Phasen der Informationsanalyse durch das Hinzufügen von zusätzlichen Daten auswirkt. Dadurch würden relativ große XML-Dokumente entstehen, die sich nicht mehr effizient betrachten oder auswerten lassen. Aus diesem Grund wird der Inhalt der einzelnen Grundbuchdokumente in dieser abschließenden Phase der Normalisierung straßenweise auf mehrere Dateien zur weiteren Analyse aufgeteilt.

3.6 Auswertung der regulären Anteile

Die einzelnen Bestandteile der Eingabedokumente für das Abkürzungs- und das Personenverzeichnis und die erste Rubrik für die Angaben zur Grundbuchnummer in den Grundbucheinträgen sind regulär aufgebaut und können somit durch eine Grammatik beschrieben werden. Für jeden Typ von XML-Element existiert dafür eine grammatikalische Regel, welche die Informationen identifiziert und in eine darauf angepasste Struktur in Form eines abstrakten Syntaxbaums überführt. Das bedeutet, bei der Analyse der Eingabedateien wird für jedes dieser Elemente die entsprechende Regel auf dessen Textknoten angewandt und dafür ein Baum angelegt, der die extrahierten Informationen strukturiert beinhaltet. Die flexible Umsetzung dieses Baums auf eine XML-Struktur und das Einpassen des Fragmentes in die Ausgangsstruktur soll in den nächsten Abschnitten erläutert werden.

3.6.1 Umsetzung des abstrakten Syntaxbaums auf eine XML-Struktur

Die Grammatik erzeugt beim Parsen des übergebenen Strings einen Syntaxbaum, dessen Struktur in der Regeldefinition beeinflusst werden kann. Ein Knoten dieses Syntaxbaums enthält dabei jeweils die Angabe des Knotentyps und den Knotentext, der im Folgenden als Knotenname bezeichnet wird. Da auch XML-Dokumente Baumstrukturen darstellen, lässt sich also ein Syntaxbaum als XML-Struktur interpretieren, wenn die Knoten entsprechend auf Elemente, Attribute und Text-

knoten umgesetzt werden. Dabei werden für diese Arbeit folgende Konventionen eingeführt[23]:

- Hat ein Knoten des Syntaxbaums Kinder, wird er auf ein XML-Element abgebildet.
- Falls ein Knotenname Angaben in eckigen (`[]`) Klammern enthält, wird der Knoten als XML-Element interpretiert, auch wenn er keine Kinder besitzt.
- Angaben in eckigen Klammern werden auf XML-Attribute zu dem jeweiligen generierten XML-Element des Knotens abgebildet. Dafür müssen die Angaben folgendermaßen aussehen:

  `[Attributname1=Attributwert1#Attributname2=Attributwert2#...]`

- Enthält ein Knotenname keine Angaben in eckigen Klammern und hat dieser Knoten keine Kinder, wird der Knoten als XML-Textknoten interpretiert. Wenn zu dem Vaterknoten schon ein XML-Textknoten existiert, wird dieser neue XML-Textknoten an den schon vorhandenen mit einem Leerzeichen davor angefügt.

Diese Umsetzungsregeln bieten flexible Möglichkeiten bei der Generierung eines nutzerdefinierten XML-Formats auf der Grundlage von Syntaxbäumen. Die Vorgehensweise wird durch Abbildung 3.12 noch einmal verdeutlicht.

3.6.2 Einpassung des XML-Baums in die Quellstruktur

Das durch die beschriebene Vorgehensweise konstruierte XML-Fragment muss nun in die Struktur des Eingabe-XML-Dokuments korrekt eingepasst werden, sodass eine adäquate Zielstruktur entsteht. Dabei ist es in einigen Fällen nötig, das Vaterelement des geparsten Textknotens durch mehr als nur ein neues XML-Element zu ersetzen. Aus diesem Grund ist die Substitution des Vaterelements durch das komplette XML-Fragment nicht angemessen. Für eine ausreichend flexible Möglichkeit, die Zielstruktur zu beeinflussen, werden stattdessen alle Kind-XML-Elemente der Fragmentwurzel anstelle des betreffenden Elements in der Quellstruktur eingesetzt. Dabei werden alle Attribute und Kindelemente vom ursprünglichen Element in die neuen Knoten mit übernommen, solange dieser keinen Textknoten darstellt. Ein Beispiel dafür ist in Abbildung 3.12 zu sehen. Eine konkrete Anwendung für das Personenverzeichnis zeigt Abbildung 3.13.

3.7 Auswertung der Volltextanteile

Die Auswertung der Volltextanteile der Grundbuchdateien wird mit Hilfe von Wörterbüchern vorgenommen. Aufgrund dieser Wortverzeichnisse werden die einzelnen Token innerhalb der betreffenden XML-Elemente identifiziert, indem der Wortabstand der Token zu den Begriffen in den einzelnen Wörterbüchern berechnet und interpretiert wird. Der generelle Ablauf der Analyse, dessen einzelne Schritte im Folgenden näher erläutert werden, ist in Abbildung 3.14 dargestellt.

[23] Der Algorithmus nutzt diejenigen Möglichkeiten, welche durch den auf Seite 83 beschriebenen Lexer/Parsergenerator ANTLR gegeben sind.

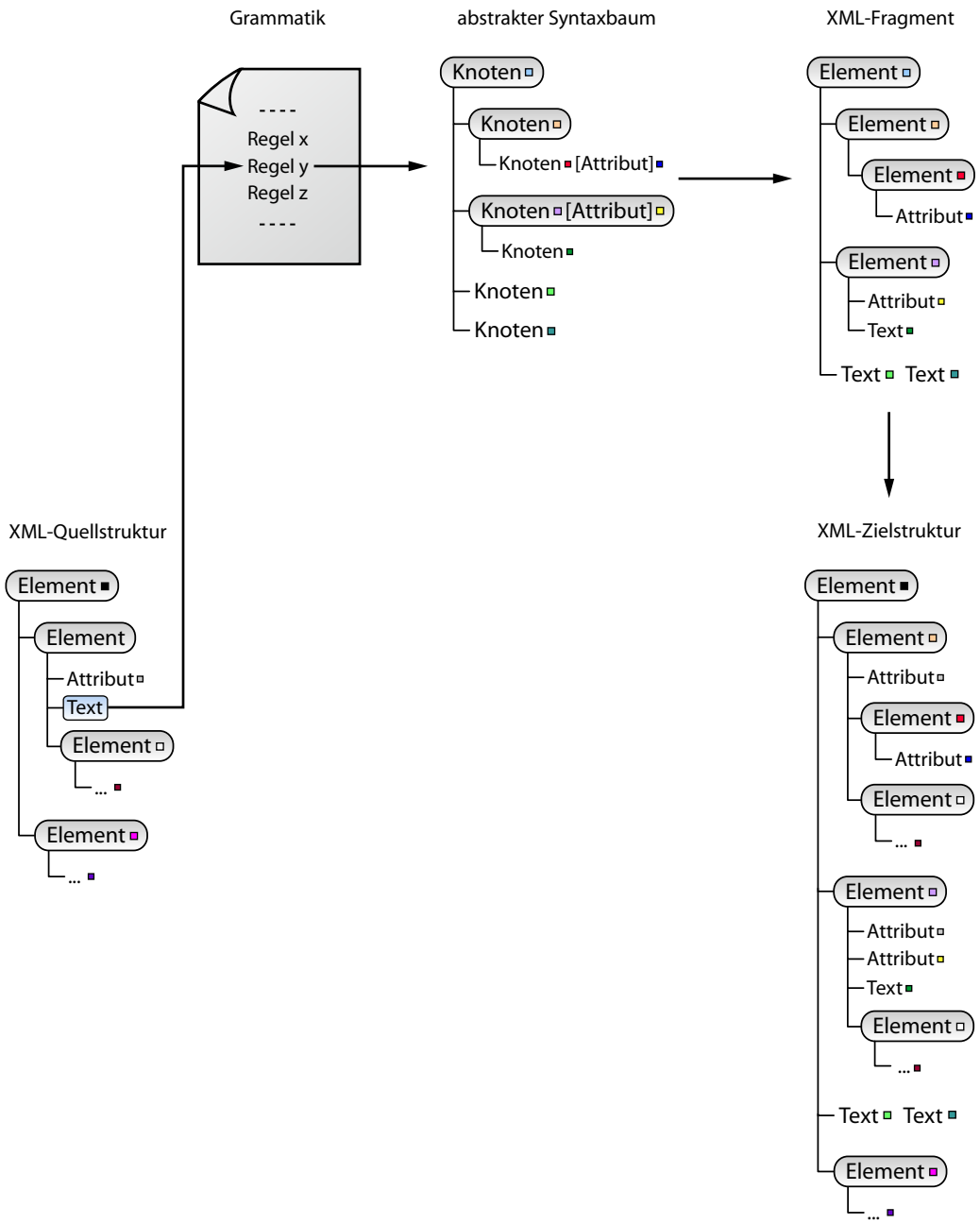
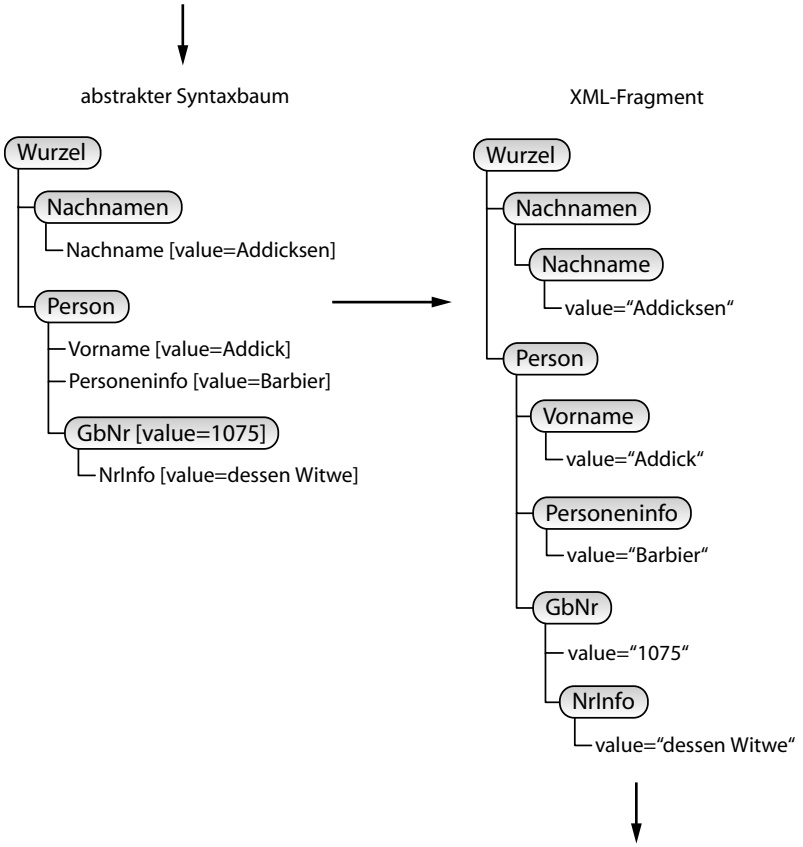


Abbildung 3.12: *Transformation und Einpassung des abstrakten Syntaxbaums*

```
<Eintrag>
  <NachnamenPerson>Addicksen, Addick (Barbier) (dessen Witwe) 1075
  <OrigZNr value="27"/>
  <OrigZInh value="Addicksen, Addick (Barbier) (dessen Witwe) 1075"/>
  </NachnamenPerson>
</Eintrag>
```




```
<Eintrag>
  <Nachnamen>
    <Nachname value="Addicksen"/>
    <OrigZNr value="27"/>
    <OrigZInh value="Addicksen, Addick (Barbier) (dessen Witwe) 1075"/>
  </Nachnamen>
  <Person>
    <Vorname value="Addick"/>
    <Personeninfo value="Barbier"/>
    <GbNr value="1075">
      <NrInfo value="dessen Witwe"/>
    </GbNr>
    <OrigZNr value="27"/>
    <OrigZInh value="Addicksen, Addick (Barbier) (dessen Witwe) 1075"/>
  </Person>
</Eintrag>
```

Abbildung 3.13: *Beispiel für die Auswertung der regulären Anteile*

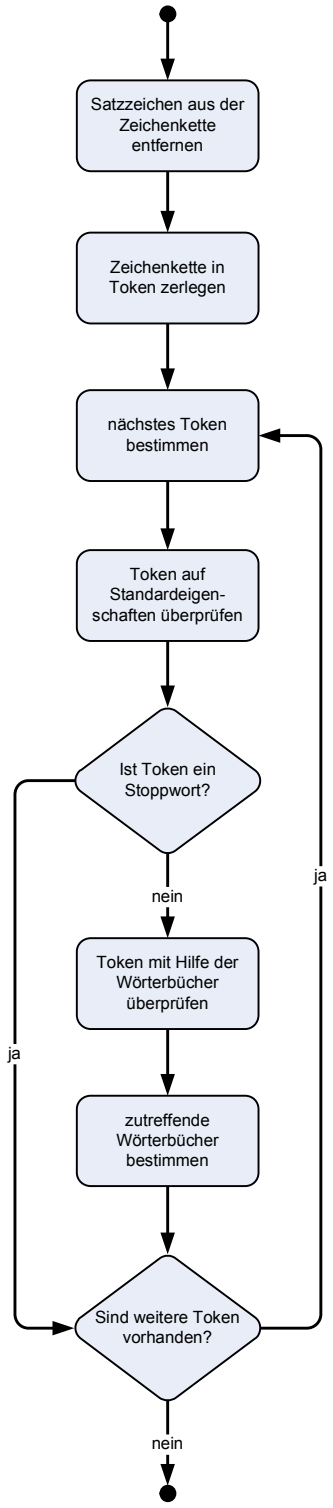


Abbildung 3.14: *Genereller Ablauf der Volltextauswertung*

3.7.1 Entfernung der Satzzeichen

Vor der Analyse der Zeichenkette werden die Satzzeichen aus ihr entfernt, da diese nicht eindeutig als Abgrenzung von unterschiedlichen Informationstypen angesehen werden können. So hat zum Beispiel der Punkt unterschiedliche, nicht eindeutig identifizierbare Funktionen innerhalb eines Eintrags (siehe Abbildung 3.15). Zum einen fungiert er als Abschluss einer Informationseinheit, zum anderen — in den meisten Fällen — macht er eine Abkürzung deutlich. An vielen Stellen ist die Funktion des Punktes zudem nicht eindeutig bestimmbar. Somit kann der Punkt nicht dazu dienen, nicht zusammengehörige Informationen voneinander abzugrenzen. Genauso verhält es sich mit dem Komma: Das Komma im Ausdruck „Nr. 13, 14, 15“ bedeutet hier nicht automatisch, dass die Zahlen in einem unterschiedlichen Zusammenhang gesehen werden müssen.

```
<Grundschuld>2100 m. Hans Elias Ramm in Rostock. Lun. a. Misericord. Domini 1831.
     getilgt d. 9. November 1839.</Grundschuld>
```

Abkürzung
Kennzeichnung einer Ordinalzahl
Ende eines Informationsabschnittes
Abkürzung und Ende eines Informationsabschnittes

Abbildung 3.15: *Funktionen des Punktes innerhalb des Textes*

Dem Fragezeichen hingegen kommt bei den Grundbucheinträgen eine besondere Bedeutung zu, weswegen es nicht aus der zu analysierenden Zeichenkette entfernt wird. Hier wird es nicht als Satzendzeichen eingesetzt, sondern kennzeichnet Unsicherheiten des Herausgebers bei der Quellensicherung. So bedeutet „Wilhelm Friedrich Richter?“, dass der Name zum Beispiel aufgrund der schlechten Lesbarkeit der Quelle nicht eindeutig identifiziert werden konnte. In diesem Zusammenhang kann aber nicht exakt bestimmt werden, ob sich die Unsicherheit nur auf den Nachnamen oder den ganzen Personennamen bezieht. In dieser Arbeit wurde das Fragezeichen immer auf das direkt vorangehende Token bezogen, da diese Betrachtungsweise in den meisten Fällen der Richtigkeit entspricht.

Die meisten Satzzeichen können also keine eindeutigen, zusätzlichen Informationen über den Zusammenhang von Worten der zu analysierenden Zeichenkette liefern, wie am Beispiel des Punktes deutlich wurde. Somit werden Zeichen wie `.,;:!()` im ersten Schritt der Inhaltsanalyse aus der entsprechenden Zeichenkette entfernt.

3.7.2 Zerlegung in Token

Nachdem die Zeichenkette entsprechend vorbereitet wurde, folgt nun ihre Aufteilung in Token. Zur Bestimmung der Abgrenzung von einzelnen Worten der Eingabe wurde dazu das Leerzeichen bestimmt.

3.7.3 Untersuchung der Token auf Standardeigenschaften

Vor der Analyse eines Tokens mit Hilfe der Wörterbücher wird dieses auf zwei Eigenschaften hin überprüft[24]:

[24] Natürlich können später weitere Eigenschaften definiert werden, die sich für die Auswertung der Token als hilfreich herausstellen.

- Zuerst wird geprüft, ob das Token ein **Stoppwort** ist, das heißt, ob es in der Stoppwortliste aufgeführt ist. In diesem Fall wird das Token keiner weiteren Untersuchung unterzogen, weder in Bezug auf die folgende Eigenschaft, noch in Bezug auf die Wörterbücher.
- Falls das Token nicht als Stoppwort identifiziert wurde, wird überprüft, ob das Token eine **Zahl** darstellt. Diese Eigenschaft kommt bei der Analyse mit den Wörterbüchern zum Einsatz und kann darüber hinaus bei der Regeldefinition in der späteren semantischen Analyse hilfreich sein.

Da, wie im vorangegangenen Abschnitt beschrieben, das Fragezeichen in einigen Token enthalten sein kann, muss dieses bei der Untersuchung des Tokens auf die Standardeigenschaften und beim Vergleich mit den Wörterbuchbegriffen außer Acht gelassen werden.

3.7.4 Überprüfung der Token mit Hilfe der Wörterbücher

Nachdem die beiden vorangegangenen Eigenschaften für das Token überprüft wurden, folgt nun die Identifikation des Tokens durch die Wörterbücher[25]. Für diese Überprüfung wird das Token nacheinander mit den Begriffen in den einzelnen Wortverzeichnissen verglichen, wobei die eventuell enthaltenen Fragezeichen außer Acht gelassen werden, und der jeweilige Wortabstand berechnet. Die genaue Vorgehensweise für jedes der zu untersuchenden Wörterbücher ist dabei folgende (siehe auch Abbildung 3.16):

1. Das Wörterbuch wird geladen. Der initiale minimale Wortabstand beträgt ∞.
2. Der erste Begriff wird aus dem Wörterbuch geholt.
3. Wenn eines der beiden Wörter — Token oder Wörterbuchbegriff — eine Zahl ist und das andere nicht, wird der Wortabstand nicht berechnet[26].
4. Wenn für das Wörterbuch vom Nutzer angegeben wurde, dass die Unterscheidung zwischen Groß- und Kleinschreibung irrelevant ist, werden Token und Begriff in Kleinschreibung umgewandelt.
5. Wenn für das Wörterbuch vom Nutzer angegeben wurde, dass beim Vergleich mit diesem die Rechtschreibung normalisiert werden soll, dann wird dies für das Token und den Wörterbuchbegriff durchgeführt.
6. Der Wortabstand zwischen — den eventuell umgeformten — Token und Begriff wird berechnet.
7. Falls vom Nutzer die Wichtung der berechneten Wortabstände definiert wurde, wird dies durchgeführt.
8. Wenn der berechnete Wortabstand des Tokens zum aktuellen Begriff kleiner ist als die bisher angenommene minimale Distanz, dann wird der Minimalwert aktualisiert.

[25] Wie bereits erwähnt, erfolgt der Einsatz der Wörterbücher nur in dem Fall, in dem das Token nicht als Stoppwort erkannt wurde.

[26] Bei dieser Maßnahme wird angenommen, dass, wenn eine Zeichenfolge neben Buchstaben auch Zahlen enthält, dies beabsichtigt und nicht auf Tippfehler zurückzuführen ist.

9. Wenn weitere Begriffe im Wörterbuch existieren, wird der nächste bestimmt und bei 3. fortgesetzt.

3.7.5 Aufbau der Wörterbücher

Für die Beschaffenheit der in den Wörterbüchern enthaltenen Begriffe gibt es zwei Ansätze, die in Abbildung 3.17 gegenübergestellt sind:

- **Wörterbuchvariante 1**
  Jedes einzelne Wort in dem Wörterbuch stellt einen eigenständigen Eintrag dar. Das heißt, logisch zusammengehörige Begriffe wie beispielsweise „Assumptio Mariae“[27] werden als zwei separate Begriffe interpretiert. Bei diesem Ansatz werden also Leerzeichen als Abgrenzung der einzelnen Einträge angesehen.

- **Wörterbuchvariante 2**
  Ein Eintrag in einem Wörterbuch kann aus mehreren, durch Leerzeichen getrennten Worten bestehen. Hier muss dann ein anderes Merkmal, beispielsweise ein Zeilenumbruch, die Abgrenzung der Einträge in so einem Wörterbuch kenntlich machen.

Der Nachteil der ersten Variante besteht darin, dass beispielsweise die beiden erkannten Token „Assumptio“ und „Mariae“, denen die Bedeutung `Feiertag` durch die Identifikation mit dem entsprechenden Wörterbuch zugeordnet wird, später wieder durch semantische Regeln zu einer Informationseinheit zusammengefasst werden müssen[28]. Weiterhin kann diese Betrachtung der Wörterbucheinträge zu Mehrdeutigkeiten bei der Bestimmung der Bedeutung eines Tokens führen, was durch Wörterbücher der zweiten Variante ausgeglichen werden würde. Dieser Aspekt im Vergleich der beiden Varianten ist in Abbildung 3.18 beispielhaft dargestellt. Andererseits ist bei diesem ersten Ansatz die Wahrscheinlichkeit größer, viele Token zu erkennen, da auch ein allein stehendes „Assumptio“ in der Zeichenkette als Feiertagsbegriff identifiziert werden würde.

Die Methode der zweiten Wörterbuchvariante hat den Nachteil, dass der Tokenisierung der zu analysierenden Zeichenkette vorbereitende Schritte vorangehen müssen:

1. Diejenigen Wörterbuchbegriffe, welche aus mehreren Worten zusammengesetzt sind, müssen bestimmt werden.

2. Die Zeichenkette muss nach diesen Begriffen bzw. Wortgruppen durchsucht werden.

3. Falls ein Begriff in der Zeichenkette gefunden wurde, muss dieser Wortgruppe die entsprechende Bedeutung zugeordnet werden.

4. Daraufhin werden die restlichen Token mit den Wörterbüchern identifiziert, wie in Abbildung 3.16 beschrieben. Dabei bleiben diejenigen Token unberücksichtigt, die bereits in Schritt 3 identifiziert wurden.

[27] Bezeichnung für den 15. August.

[28] Durch geschickte Definition der Ersetzungsregeln im vorangegangenen Normalisierungsprozess (Kapitel 3.5.3 auf Seite 36) könnte zum Beispiel jedes Auftreten von „Assumptio Mariae“ durch „Assumptio_Mariae“ ersetzt und dieses Substitut in das Feiertagsverzeichnis aufgenommen werden. Dadurch würde das Zusammenführen der beiden Token entfallen.

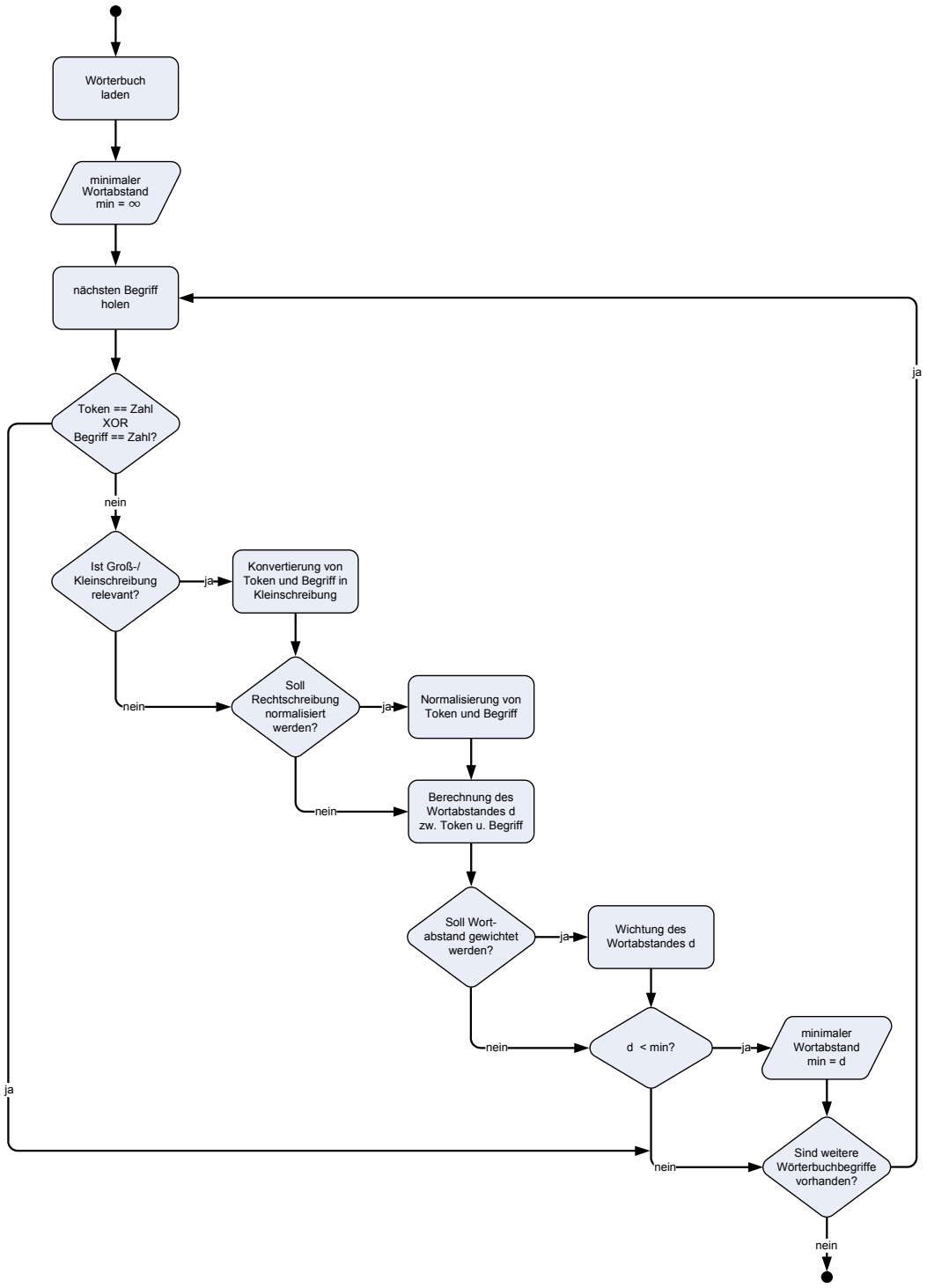


Abbildung 3.16: *Identifikation eines Tokens mit Hilfe eines Wörterbuches*

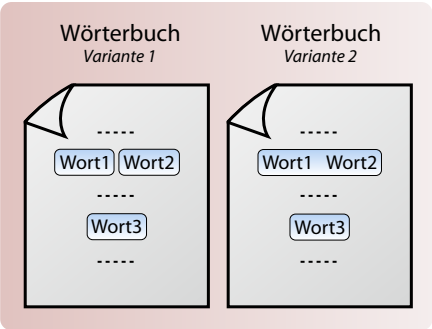


Abbildung 3.17: *Möglichkeiten für die Interpretation der Wörterbucheinträge*

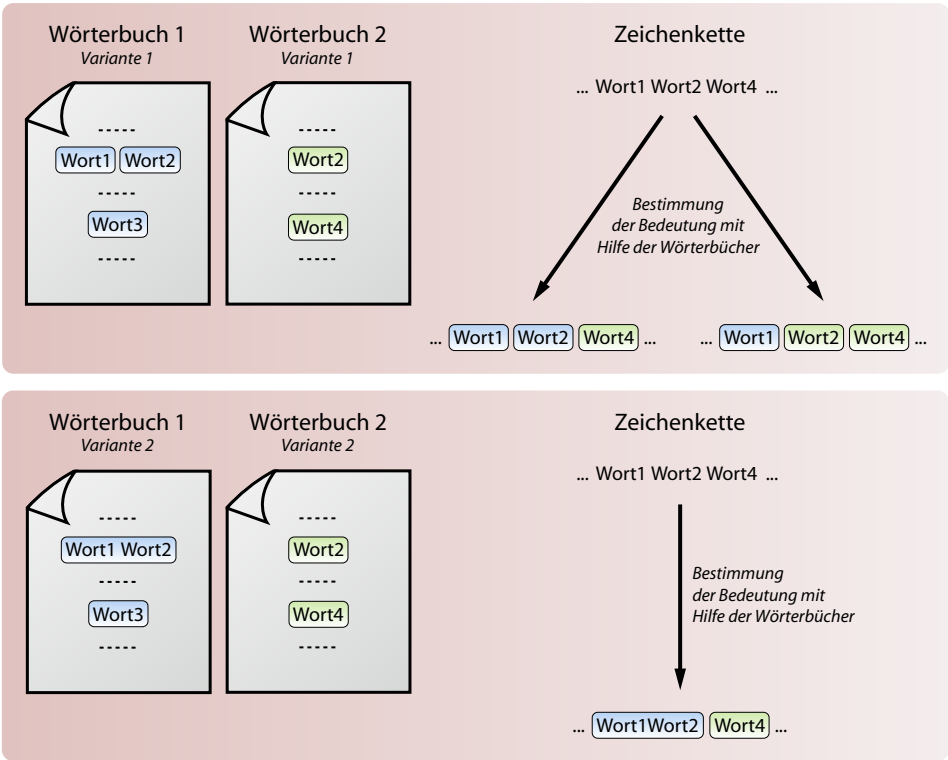


Abbildung 3.18: *Mehrdeutigkeiten bei der Tokenidentifikation*

Ein weiteres Problem stellt bei dieser Art von Wörterbuch der zweite Schritt dar. Die Identifizierung der Wortgruppen in der zu analysierenden Zeichenkette kann nämlich keinesfalls immer eindeutig erfolgen. Um dies zu vermeiden, müsste eine Priorisierung der Wörterbuchbegriffe vorgenommen werden, die aber ohne die semantische Betrachtung der gesamten Zeichenkette kaum möglich ist. Im Gegensatz dazu tritt dieses Problem bei Wörterbüchern der ersten Variante nicht auf, wie in Abbildung 3.19 zu sehen ist.

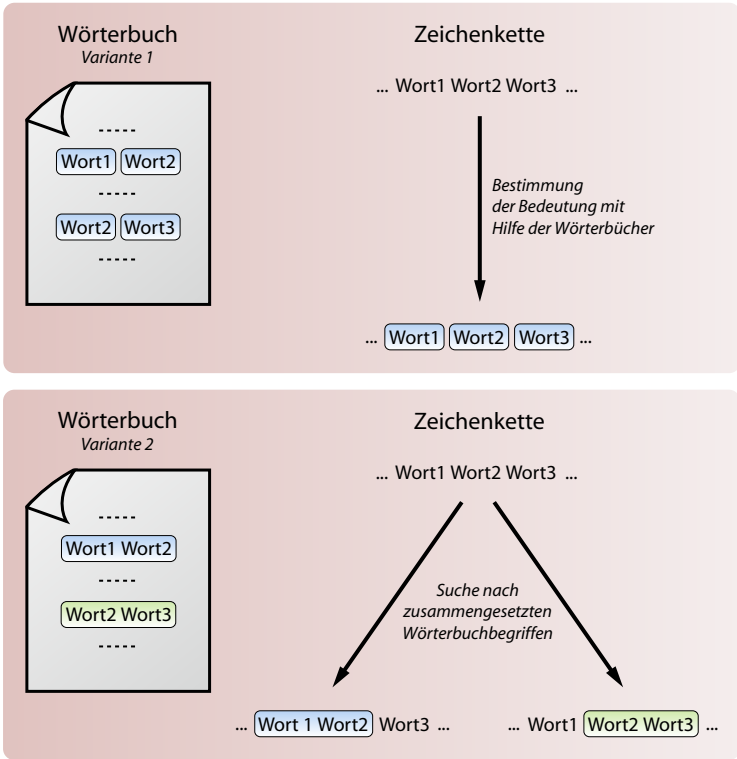


Abbildung 3.19: *Mehrdeutige Identifizierung von zusammengesetzten Begriffen*

Trotz dieser Probleme bietet diese Methode den Vorteil, dass die Zusammenfassung von zusammengehörigen Worten zu einem Begriff nach der Identifikation der Token entfallen würde, da man diese Begriffe in der zu analysierenden Zeichenkette gleich als solche erkennt.

Zusammenfassend lässt sich also sagen, dass beide Varianten der Betrachtungsweise der Wörterbucheinträge Vor- und Nachteile mit sich bringen. Bei der ersten Variante wird die semantische Behandlung zusammengehöriger Begriffe auf eine spätere Verarbeitungsphase verschoben, während diese bei der zweiten Methode schon bei der Identifizierung der zusammengesetzten Begriffe geschehen muss. Da die semantische Analyse also in keinem der beiden Fälle vollkommen entfallen kann, wird in dieser Arbeit aufgrund der einfacheren Vorgehensweise und Verständlichkeit die erste Methode angewendet.

Auf der Grundlage der Evaluationsergebnisse der Volltextanteile wurden in dieser Arbeit 23 Wörterbücher angelegt. Die beiden Wortverzeichnisse für die Vornamen und Nachnamen wurden dabei aus denjenigen Daten in der Datenbank generiert, die vorher aus dem Personenverzeichnis extrahiert wurden, wodurch die eingangs angesprochene iterative Vorgehensweise bei der Auswertung

der Eingabedateien umgesetzt wird.

3.7.6 Berechnung des Wortabstandes

Als Kriterium für die Bestimmung der Ähnlichkeit zwischen dem Token, das identifiziert werden soll, und einem Begriff aus dem Wörterbuch dient der Wortabstand zwischen diesen beiden Wörtern. Die Wortabstandsberechnung ist ein oft eingesetztes Feature in Textverarbeitungsprogrammen für die Rechtschreibkorrektur. Dabei wird der Abstand zwischen einem falsch geschriebenen Wort im Text und den einzelnen Wörtern in den internen Dateien des Programms evaluiert und diejenigen mit den geringsten Abständen als Ersetzungskandidaten vorgeschlagen [Bog].

Für die Berechnung des Wortabstandes existieren unter anderem folgende grundlegende Ansätze:

- Die **Hamming-Distanz** H der beiden zu vergleichenden Strings s und t ist folgendermaßen definiert:

  $$H(s,t) = \text{Anzahl der Stellen, an denen } s \text{ und } t \text{ sich unterscheiden.}$$

  Der Hamming-Abstand ist jedoch nur für den Vergleich von Zeichenketten gleicher Länge definiert und ist somit für die meisten Anwendungen ungeeignet. [Bog]

- Die **Levenshtein-Distanz**[29] ist für den Vergleich von Zeichenketten beliebiger Länge definiert und wird häufig für automatische Rechtschreibprüfungen, Spracherkennung, DNS-Analysen und Plagiat-Entdeckung eingesetzt. Der Algorithmus wurde 1965 vom russischen Wissenschaftler Vladimir Levenshtein entwickelt und ermittelt, wieviele Einfüge-, Lösch- und Austauschoperationen nötig sind, um eine Zeichenkette s in eine andere Zeichenkette t zu überführen. Jede dieser Operationen wird dabei mit Strafpunkten belegt. Für zwei Zeichen a und b ist dabei Folgendes definiert:

  $$\begin{aligned} \text{falls } a = b \quad &\rightarrow \quad r(a,b) = 0, \\ \text{andernfalls} \quad &\rightarrow \quad r(a,b) = 1. \end{aligned}$$

  Seien s und t zwei Zeichenketten der Länge n bzw. m. $s(i)$ bzw. $t(j)$ bezeichne das i-te bzw. j-te Zeichen in s respektive t. Eine Matrix d der Größe $(n+1) \times (m+1)$ wird nun rekursiv mit Werten belegt:

  $$\begin{aligned} d(i,0) &= i \qquad (i = 0, ..., n) \\ d(0,j) &= j \qquad (j = 0, .., m) \\ d(i,j) &= min(d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1)+r(s(i),t(j))). \end{aligned}$$

  Sind alle Matrixeinträge berechnet, stellt der Wert $d(n,m)$ den resultierenden Levenshtein-Abstand $L(s,t)$ der beiden Zeichenketten s und t dar. Ein Beispiel für die schrittweise Berechnung einer solchen Matrix ist in Abbildung 3.20 zu sehen. [Bog, Gil]

[29] Wird auch Edit-Distanz genannt.

s = Gumbo; t = Gambol

d(i,0); d(0,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1					
A	2					
M	3					
B	4					
O	5					
L	6					

d(1,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1	0				
A	2	1				
M	3	2				
B	4	3				
O	5	4				
L	6	5				

d(2,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1			
A	2	1	1			
M	3	2	2			
B	4	3	3			
O	5	4	4			
L	6	5	5			

d(3,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2		
A	2	1	1	2		
M	3	2	2	1		
B	4	3	3	2		
O	5	4	4	3		
L	6	5	5	4		

d(4,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	
A	2	1	1	2	3	
M	3	2	2	1	2	
B	4	3	3	2	1	
O	5	4	4	3	2	
L	6	5	5	4	3	

d(5,j)

i \ j		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

L(s,t)

Abbildung 3.20: *Beispiel für die Berechnung der Levenshtein-Distanz*

Der Hamming-Abstand kann nur sehr eingeschränkt hilfreich sein bei der Wortähnlichkeitsbestimmung. Im Gegensatz dazu zählt die Levenshtein-Distanz zu den etabliertesten Methoden bei dieser Problemstellung, vor allen Dingen im Bereich der Textverarbeitungsprogramme. Aus diesen Gründen wurde in dieser Arbeit der zweite Ansatz zur Bestimmung der Wortabstände zwischen den einzelnen zu identifizierenden Token und den Begriffen der Wörterbücher eingesetzt.

3.7.7 Wichtung des Wortabstandes

Nachdem ein Wortabstand zwischen einem Token s und einem Wörterbuchbegriff t nach der oben dargestellten Methode berechnet wurde, kann zusätzlich eine Wichtung dieser Distanz erfolgen. Dabei wird der berechnete Wert ins Verhältnis zur Tokenlänge n gesetzt:

$$L_w(s,t) = \frac{L(s,t)}{n}.$$

Dies hat zur Folge, dass derselbe Wortabstand bei kurzen Token gewichtet größer als bei längeren Token ist. Damit wird der Umstand ausgeglichen, dass zwei kurze Worte verschiedener Bedeutung sich häufig nur in wenigen Zeichen unterscheiden, ohne dass die Begründung dafür in Rechtschreibfehlern liegt. Im Gegensatz dazu ist die Wahrscheinlichkeit bei zwei längeren Zeichenketten größer, dass ein geringer Wortabstand durch falsche Schreibung entsteht, die Worte aber dennoch die gleiche Bedeutung haben.

3.7.8 Zuordnung des Tokens zu einem Wörterbuch

Nachdem zu jedem Token die Wortabstände zu allen Wörterbüchern ermittelt wurden, muss nun entschieden werden, zu welchem dieser Wörterbücher das Token zugeordnet werden soll, das heißt, welche Bedeutung für das Token ermittelt wird. Dafür muss ein nutzerdefinierbarer Schwellenwert für den Wortabstand angegeben werden. Alle Wortverzeichnisse, dessen berechnete Distanz über diesem Wert liegt, werden dem Token nicht zugeordnet. Für die unter dem Wert liegenden Abstände bzw. Wörterbücher gibt es folgende zwei Ansätze:

- Das Token wird inhaltlich nur dem Wörterbuch bzw. den Wörterbüchern zugeordnet, die von allen Wörterbüchern den **minimalsten Abstand** zu dem Token aufweisen. Alle anderen Wörterbücher, deren Distanz zwar auch unter dem angegebenen Schwellenwert liegt, aber die nicht minimal in Bezug auf die Distanzen der restlichen Wortverzeichnisse ist, werden bei der weiteren Betrachtung der Token außer Acht gelassen.
- Dem Token werden **alle Wörterbücher** zugeordnet, deren berechnete Distanz **unter dem Schwellenwert** liegt.

Als Ergebnis wird jedem Token entweder keine, eine oder mehrere Bedeutungen zugeordnet. Beim zweiten Ansatz ist die Wahrscheinlichkeit, dass einem Token mehrere in Frage kommende Wörterbücher zugeordnet werden, sehr viel größer als bei der ersten Variante, bei der man jedoch in manchen Fällen Bedeutungen ausschließt, obwohl diese dennoch in Betracht gezogen werden müssen. In jedem Fall müssen in nachfolgenden Schritten durch semantische Regeln die verschiedenen Bedeutungen eines Tokens auf möglichst eine reduziert werden, damit den einzelnen Informationen ein eindeutiger Typ zugeordnet werden kann. Dieser Vorgang wird die nächste Phase im Verarbeitungsprozess darstellen.

3.7.9 Normalisierung der Rechtschreibung

Ein essentielles Problem bei den Grundbucheinträgen aus dem 17. Jahrhundert ist die stark regional und temporal variierende Rechtschreibung, für die es zur damaligen Zeit keine allgemeingültigen Regeln gab. Somit existieren zu ein und demselben Wort mehrere Varianten der Schreibung, die bei der Analyse der Token berücksichtigt werden müssen.

Im Personenverzeichnis wurden die verschiedenen auftretenden Schreibweisen für die Nachnamen aufgelistet, wohingegen für Vornamen nur die moderne Form der Schreibung aufgeführt wurde. Das bedeutet, dass in den Fällen, in denen mögliche Spielarten der Orthographie in den Wörterbüchern nicht berücksichtigt wurden, die Wahrscheinlichkeit der Erkennung von Token enorm sinkt. Da es aber in einigen Fällen nicht möglich oder praktikabel ist, alle Schreibvarianten in die Wörterbücher zu integrieren, muss eine Möglichkeit gefunden werden, vor der Berechnung des Wortabstandes das Vergleichspaar in eine normalisierte Form zu überführen. Zum Einsatz kommen dazu sprachorientierte Verfahren, die zwei zu vergleichende Worte so konvertieren, dass gleich klingende Laute auf dasselbe Zeichen oder Token abgebildet werden. Die so entstehenden Worte werden sich in dieser reduzierten Grundform ähnlicher oder sogar gleich. Zwei dieser Verfahren sollen nachfolgend kurz vorgestellt werden[30]:

- Das **Soundex-Verfahren** wurde von Margaret K. Odell und Robert C. Russell entwickelt und patentiert und dient hauptsächlich der Auffindung von Namen. Der Konvertierungsalgorithmus lautet wie folgt:

  1. Entferne alle Buchstaben einer Sequenz, die auf den gleichen Code abgebildet werden, bis auf den ersten der Sequenz.
  2. Behalte den ersten Buchstaben bei.
  3. Entferne die Buchstaben `a`, `e`, `h`, `i`, `o`, `u`, `w`, `y`.
  4. Ersetze folgende Buchstaben durch das jeweilige Token:

Buchstaben	Token
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Tabelle 3.1: *Ersetzungstabelle des Soundex-Verfahrens*

  5. Bringe die entstandene Zeichenfolge in die Form `BuchstabeZifferZifferZiffer`, indem entweder mit Nullen aufgefüllt wird oder Zeichen am Ende abgeschnitten werden.

     Beispiel: `Friedrich` → `F636`

[30] Die Informationen hierzu stammen aus [Fri96].

Der Vorteil dieser Methode besteht in der Reduzierung des Wortes auf eine kurze Ergebniszeichenkette. Jedoch ist dieser Algorithmus auf die englische Sprache ausgelegt und betrachtet nur eine endliche Stellenanzahl im Ergebnis. Dies hat zur Folge, dass bei langen Worten das Wortende nicht berücksichtigt wird. Weiterhin wird die zu transformierende Zeichenkette zu stark generalisiert, indem viele Zeichen auf nur sechs verschiedene Token reduziert werden. Außerdem werden Fehler in den Anfangsbuchstaben durch die zweite Regel des Algorithmus nicht ausgeglichen.

- Die **phonetische Kodierung** ist dem Soundex sehr ähnlich, beinhaltet jedoch umfangreichere Ersetzungen und ist auf die deutsche Sprache ausgelegt. Buchstaben bzw. Buchstabenkombinationen werden hier nicht durch Token, sondern durch ähnlich klingende Buchstaben respektive Buchstabenkombinationen substituiert:

  1. Wandle alle Buchstaben in Großbuchstaben um.
  2. Wandle alle Buchstabenkombinationen nach folgender Tabelle um:

Zeichenfolge	Ersetzung	Zeichenfolge	Ersetzung
SC	C	QU	KV
SZ	C	UE	Y
CZ	C	EU	OY
TZ	C	AE	E
TS	C	OE	Ö
DS	C	KS	X
PH	V	EI	AY
PF	V	EY	AY

Tabelle 3.2: *Phonetische Kodierung für Buchstabenkombinationen*

  3. Wandle einzelne Buchstaben nach folgender Tabelle um:

Zeichen	Ersetzung	Zeichen	Ersetzung
K	C	W	V
G	C	F	V
Q	C	T	D
Ü	Y	ß	S
I	Y	P	B
J	Y		

Tabelle 3.3: *Phonetische Kodierung für einzelne Buchstaben*

  4. Filtere unerlaubte (alle Zeichen, die keine Buchstaben sind) und doppelte Zeichen heraus.

     Beispiel: `Friedrich` → `VRYEDRYCH`

Der Vorteil dieses Ansatzes gegenüber dem Soundex ist die Genauigkeit der transformierten Ergebniszeichenketten, denn hier wird keine endliche Stellenzahl betrachtet und somit keine Zeichen am Wortende abgeschnitten. Andererseits ist die Ersetzungstabelle nicht unbedingt vollständig, so könnte zum Beispiel die Zeichenfolge AH durch das Token A ersetzt werden.

Aufgrund der angesprochenen Vorteile der phonetischen Kodierung wurde dieser Ansatz für die Normalisierung der Worte eingesetzt. Zu dieser Wahl trug außerdem die Tatsache bei, dass viele Schreibvarianten in den Grundbucheinträgen auf die Ersetzung von Zeichensequenzen durch gleichlautende Äquivalente zurückzuführen sind, welche durch die oben angeführten Tabellen abgedeckt werden. Weiterhin sind die Tabellen jederzeit erweiterbar und damit flexibel auf die jeweilige Anwendung anpassbar.

Im Rahmen dieser Arbeit wurde die phonetische Kodierung in der Hinsicht angepasst, dass die Unterscheidung zwischen Groß- und Kleinschreibung eingeführt wurde. Damit entfällt der erste Schritt des Originalalgorithmus und die Ersetzungstabellen müssen entsprechend erweitert werden:

Zeichenfolge	Ersetzung	Zeichenfolge	Ersetzung
SC	C	QU	KV
Sc	C	Qu	Kv
sc	c	qu	kv
SZ	C	UE	Y
Sz	C	Ue	Y
sz	c	ue	y
CZ	C	EU	OY
Cz	C	Eu	Oy
cz	c	eu	oy
TZ	C	AE	E
Tz	C	Ae	E
tz	c	ae	e
TS	C	OE	Ö
Ts	C	Oe	Ö
ts	c	oe	ö
DS	C	KS	X
Ds	C	Ks	X
ds	c	ks	x
PH	V	EI	AY
Ph	V	Ei	Ay
ph	v	ei	ay
PF	V	EY	AY
Pf	V	Ey	Ay
pf	v	ey	ay

Tabelle 3.4: *Angepasste Kodierung von Buchstabenkombinationen*

Zeichen	Ersetzung	Zeichen	Ersetzung
K	C	j	y
k	c	W	V
G	C	w	v
g	c	F	V
Q	C	f	v
Q	c	T	D
Ü	Y	t	d
ü	y	ß	s
I	Y	P	B
i	y	p	b
J	Y		

Tabelle 3.5: *Angepasste Kodierung von einzelnen Buchstaben*

Durch diese Erweiterung bleibt der Analyseprozess der Token flexibler in Bezug auf die Ansprüche des Nutzers, inwieweit beim Vergleich der Token mit den Wörterbuchbegriffen die Unterscheidung zwischen Groß- und Kleinschreibung von Relevanz ist.

Wurde also vom Nutzer für ein Wörterbuch angegeben, dass für die Überprüfung eines Tokens mit diesem Wörterbuch eine Normalisierung der Rechtschreibung durchgeführt werden soll, werden das Token und der jeweilige Wörterbuchbegriff auf der Grundlage der phonetischen Kodierung auf ihre Grundform reduziert und dann erst der Wortabstand zwischen diesen Normalformen berechnet.

Das sprachorientierte Verfahren — die phonetische Kodierung — berücksichtigt also Varianten in der Schreibung, während das buchstabenorientierte Verfahren — die Wortabstandsberechnung — ein Maß für den Unterschied zwischen zwei Zeichenketten definiert. Eine Kombination dieser beiden Techniken ist immer dann angebracht, wenn in einem Wörterbuch auf die Integration aller möglichen Schreibvarianten verzichtet wurde — wie in dem Fall der Vornamen. Sind die verschiedenen Schreibweisen eines Begriffes bekannt und überschaubar, können diese als Einträge in das Wörterbuch mit aufgenommen und auf die Normalisierung verzichtet werden[31].

3.7.10 Nutzerdefinierbare Einstellungen

Wie in den vorangegangenen Abschnitten schon kurz angesprochen, kann der Nutzer einige Einstellungen definieren, welche die Analyse der einzelnen Token in Bezug auf die Wörterbücher entscheidend beeinflussen können. Zu diesen Parametern zählen folgende:

- Der Nutzer kann für jedes Wörterbuch angeben, ob beim Vergleich der Token mit den Begriffen die **Unterscheidung zwischen Groß- und Kleinschreibung** relevant ist. Da im 17. Jahrhundert die Rechtschreibung in Bezug auf die Groß- und Kleinschreibung stark variierte und diese Schwankungen auch in der Quellensicherung der Wismarer Grundbücher erhalten wurden, wird in den meisten Fällen eine Unterscheidung nicht sinnvoll sein. Im

[31] Letzteres führt zum Vergleich des unveränderten Tokens mit den Originalwörterbuchbegriffen, was eine gewisse Absicherung der Genauigkeit der letztendlich ermittelten Bedeutung bietet.

Fall der Feiertagsbezeichnungen und der Vor- und Nachnamen wurde jedoch die Großschreibung vereinheitlicht. In diesen Fällen könnte also eine Unterscheidung von Vorteil sein, um Mehrdeutigkeiten von Token einzuschränken.

- Zu jedem Wörterbuch kann festgelegt werden, ob vor dem Vergleich der Token mit den Begriffen in diesem Wörterbuch eine **Normalisierung** des Vergleichspaares **auf die reduzierte Grundform** durchgeführt werden soll.
- Für die Zuordnung von Bedeutungen zu einem Token muss ein **Schwellenwert für die Tokendistanz zu einem Wörterbuch** angegeben werden. Je niedriger dieser Schwellenwert gewählt wird, desto genauer muss ein Token mit einem Begriff aus dem Wörterbuch übereinstimmen, damit dieses dem Token als „gültig“ zugeordnet wird. Ein Schwellenwert von `0` bedeutet also, dass das Token in einem Wörterbuch identisch enthalten sein muss.
- Weiterhin muss festgelegt sein, welche **Auswahlmethode für die Wörterbücher**, deren Distanz zu dem Token unter dem angegebenen Schwellenwert liegt, gewählt werden soll. Entweder werden alle Bedeutungen oder nur diejenigen mit dem minimalsten Wortabstand für das Token akzeptiert.
- Der Nutzer kann festlegen, ob eine **Wichtung der berechneten Wortabstände** in Bezug auf die Tokenlänge vorgenommen werden soll.

Alle diese Parameter werden vom Nutzer in einer Settings-Datei festgelegt, die zu einem späteren Zeitpunkt näher vorgestellt wird.

3.8 Semantische Analyse

In der Phase der semantischen Analyse wird nun versucht, durch nutzerdefinierbare Heuristiken diejenigen Token eindeutig zu identifizieren, denen in der vorangegangenen Phase kein oder mehrere Wörterbücher zugeordnet wurden[32]. Nähere Details werden in den folgenden Abschnitten dargestellt.

3.8.1 Ablauf der Regelanwendung

Für die Anwendung der Regeln werden alle Token eines Informationsabschnitts mit ihren Eigenschaften der Reihenfolge entsprechend als Liste betrachtet, wodurch bei der Bearbeitung eines Tokens auch der Zugriff auf die Vorgänger- bzw. Nachfolgertoken möglich ist. Für die Eigenschaften eines Tokens werden im Folgenden die Namen der zugeordneten Wörterbücher als „Bedeutungen“ interpretiert. Diese Struktur ist beispielhaft in Abbildung 3.21 dargestellt.

Die Liste wird nun Token für Token abgearbeitet, indem für jedes Token, das keine oder mehr als eine Bedeutung besitzt — außer Stoppworte — jede definierte semantische Regel angewandt wird. Den Regeln ist dabei eine bestimmte Priorität zugeordnet, was bedeutet, dass bei zwei Regeln, welche die gleiche Bedingung beschreiben, diejenige mit der höheren Priorität zuerst ausgewertet wird. Die Vorgehensweise ist in Abbildung 3.22 noch einmal verdeutlicht.

[32] Als Stoppwort erkannte Token werden bei der weiteren Betrachtung außer Acht gelassen.

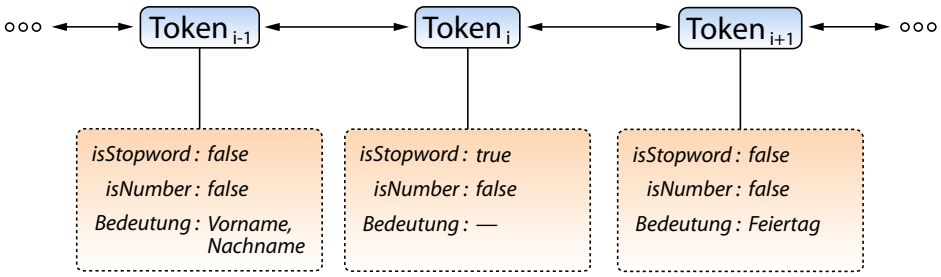


Abbildung 3.21: *Tokenliste für die Regelanwendung*

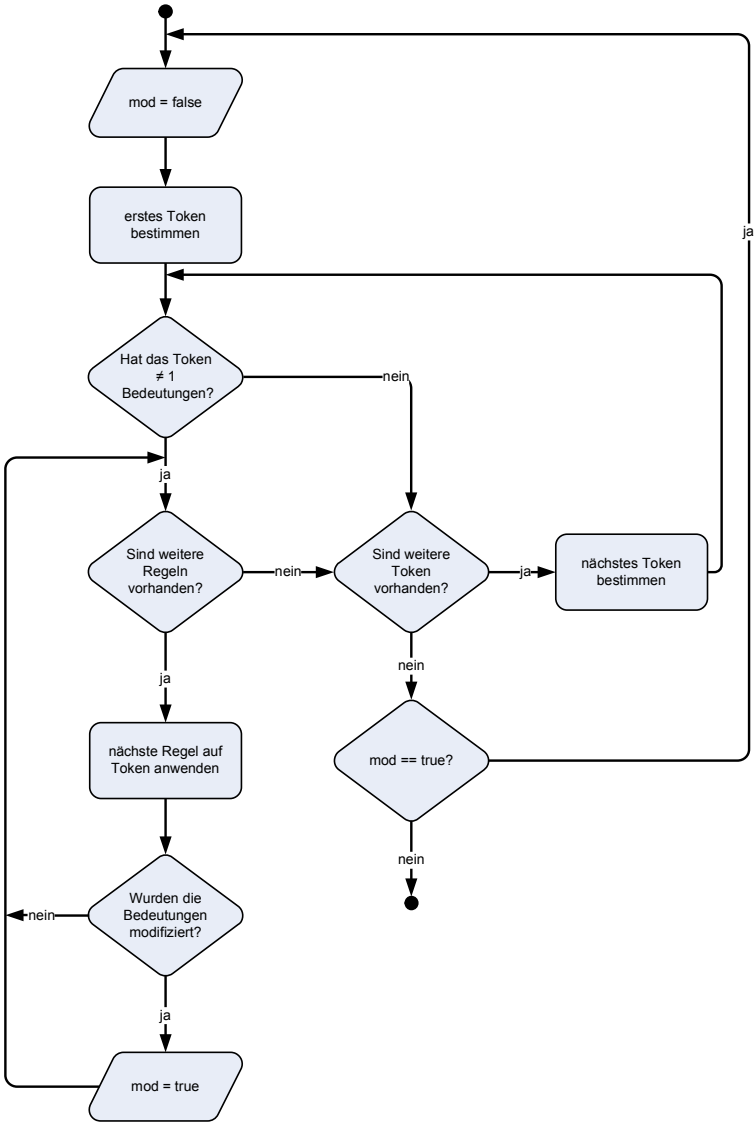


Abbildung 3.22: *Ablauf der semantischen Analyse*

Da die Anwendung einer Regel auf ein Token und dessen damit einhergehende Modifikation seiner Bedeutungen wiederum Auswirkungen auf die umgebenden Token haben kann (siehe Beispiel in Abbildung 3.23), findet die Anwendung der Regeln in mehreren Durchläufen statt: Wurde bei der Analyse der Tokenfolge etwas an den Bedeutungen geändert (in Abbildung 3.22 und 3.23 gekennzeichnet durch das Flag `mod=true`), wird nach dem letzten Token die Auswertung der Liste wiederholt. Dieser Vorgang wird solange fortgesetzt, bis keine Modifikationen an den Bedeutungen mehr vorgenommen, das heißt, keine Regel mehr angewandt werden konnte.

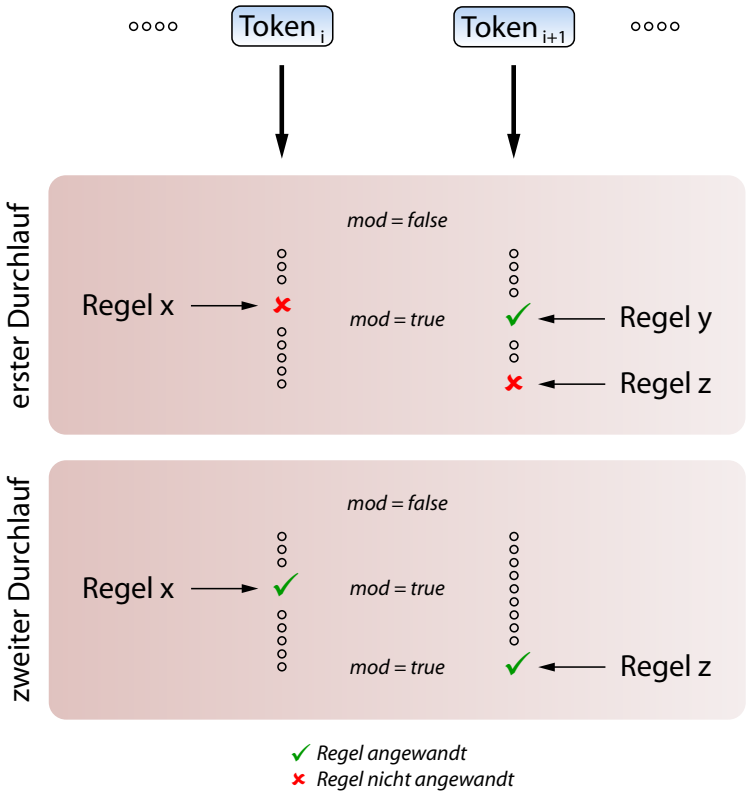


Abbildung 3.23: *Wiederholung der Regelanwendung*

3.8.2 Definition der semantischen Regeln

Eine Regel zur Bedeutungsanalyse eines Tokens besteht aus folgenden zwei Bestandteilen:

- Der **Bedingungsteil** beschreibt, welche Voraussetzungen erfüllt sein müssen, damit die angegebene Aktion durchgeführt wird. Mehrere Konditionen können dabei durch logische Operatoren beliebig geschachtelt werden.
- Der **Aktionsteil** einer Regel wird ausgeführt, wenn die angegebenen Bedingungen erfüllt sind. Der Aktionsteil kann beliebig viele Teilaktionen beinhalten.

Zur Beschreibung der Bedingungen können beispielsweise folgende Aspekte hilfreich sein:

- Wurde dem Token **keine Bedeutung** zugeordnet?

- Wurde dem Token **genau eine Bedeutung** zugeordnet?
- Wurde dem Token **mehr als eine Bedeutung** zugeordnet?
- Wurde dem Token **eine bestimmte Bedeutung** zugeordnet?
- Wurde dem Token **eine bestimmte Bedeutung nicht** zugeordnet?
- Wurde dem Token **ausschließlich eine bestimmte Bedeutung** zugeordnet?
- Wurden dem Token **mehrere bestimmte Bedeutungen** zugeordnet?
- Wurden dem Token **mehrere bestimmte Bedeutung nicht** zugeordnet?
- Wurden dem Token **ausschließlich mehrere bestimmte Bedeutungen** zugeordnet?
- Ist das Token ein **Stoppwort**?
- Ist das Token eine **Zahl**?

Die Aktionen, durch welche die Bedeutungen der Token modifiziert werden können, sind folgende:

- **Ordne** dem Token **eine oder mehrere bestimmte Bedeutungen zu**. Alle weiteren Bedeutungen, die diesem Token ursprünglich ebenfalls zugeordnet waren, werden ausgeschlossen.
- **Schließe eine bestimmte Bedeutung** für das Token **aus**.
- **Füge** dem Token **eine bestimmte Bedeutung hinzu**.

Da die Token eines Informationsabschnittes eine ihrer Reihenfolge entsprechend aufgebaute Liste darstellen, können für ein Token auch Bedingungen an seine Vorgänger und Nachfolger gestellt werden, was in den meisten Fällen zur exakten Identifikation eines mehrdeutigen Wortes unabdinglich sein wird.

Durch geschickte Definitionen von Heuristiken können so mit Hilfe dieser Tokeneigenschaften und bereitgestellten Aktionen Mehrdeutigkeiten von Token eliminiert werden. Ebenfalls ist es möglich, für Token ohne vorher ermittelte Wörterbücher Bedeutungen festzulegen. So kann beispielsweise vierstelligen Zahlen, die auf Feiertagsangaben folgen, die Rolle `Jahr` beigeordnet werden.

3.8.3 Beispiele für semantische Regeln

Eine vollständige Aufstellung von Heuristiken zur eindeutigen Tokenidentifikation kann im Rahmen dieser Arbeit nicht gegeben werden, da der Umfang der nötigen Regeln im engen Zusammenhang mit folgenden Aspekten steht:

- Je öfter ein Begriff in mehreren Wörterbüchern aufgeführt ist, desto wahrscheinlicher werden Ambiguitäten.
- Je größer der Schwellenwert des Wortabstandes für die Identifikation der Token gewählt wird, desto wahrscheinlicher werden Ambiguitäten auftreten.

- Wird bei dem Vergleich der Token mit den Wörterbuchbegriffen nicht zwischen Groß- und Kleinschreibung unterschieden und/oder die Rechtschreibung normalisiert, können mehr Vieldeutigkeiten auftreten.
- Je weniger Begriffe in den Wörterbüchern enthalten sind bzw. je weniger Wörterbücher zur Anwendung kommen, desto mehr Token kann überhaupt keine Bedeutung zugeordnet werden.
- Je geringer der Schwellenwert für den Wortabstand gewählt wird, desto mehr Token wird wahrscheinlich keine Bedeutung beigeordnet.

⇒ Je häufiger Ambiguitäten auftreten und je mehr Token nicht erkannt werden konnten, desto umfangreichere semantische Regeln müssen definiert werden.

Um dennoch den Ansatz der semantischen Analyse zu verdeutlichen, wurden in dieser Arbeit mehrere Regeln beispielhaft definiert, die in Anhang A aufgeschlüsselt sind.

Die Tokenliste, in der die Bedeutungen der einzelnen Token mit Hilfe der Regeln eingeschränkt bzw. modifiziert wurden, wird nach Abschluss der semantischen Analyse wiederum auf eine XML-Struktur abgebildet, die in Anhang E.4 auf Seite 122 beispielhaft dargestellt ist.

3.9 Informationsstrukturierung

Nachdem in der semantischen Analyse versucht wurde, die mehrdeutigen Informationen durch Heuristiken zu identifizieren, werden nun logisch zusammengehörige Informationen zu Einheiten gruppiert. Diese Phase lässt sich wiederum in Teilschritte untergliedern, die in den nächsten Abschnitten näher erläutert werden.

3.9.1 Aufsplitten von reduzierten Begriffen

Wie in Fußnote 28 auf Seite 47 angemerkt, kann man während der Normalisierungsphase zusammengesetzte Begriffe zu einem reduzieren, indem man die enthaltenen Leerzeichen durch Unterstriche ersetzen lässt. Den so normalisierten Begriff kann man dann als Einheit in ein Wörterbuch aufnehmen, wodurch dieser bei der Analyse des Volltextes mit Hilfe der Wörterbücher zusammenhängend erkannt wird.

Diese Unterstriche müssen nun in der abschließenden Phase wieder durch ein Leerzeichen ersetzt werden, um den ursprünglichen Inhalt zu rekonstruieren. Dies wird am Anfang dieser Phase auf allen mit Wörterbüchern analysierten Informationen durchgeführt.

3.9.2 Zusammenfügen von mehrteiligen Begriffen

Bestimmte aufeinander folgende Informationen eines Typs, die logisch zusammengehören, müssen zu einer Information vereinigt werden. Das ist zum einen notwendig bei zusammengesetzten Wörterbucheinträgen, deren einzelne Bestandteile unabhängig voneinander im Text erkannt werden. Beispielsweise würde der Eintrag „Assumptio Mariae“ im Text als zwei aufeinander folgende Feiertagsbegriffe erkannt werden. Da diese Informationen jedoch logisch zusammengehören, werden diese nun zu einer Information reduziert.

Die Auflistung aller davon betroffenen Informationstypen ist stark abhängig davon, welche Wörterbücher tatsächlich zusammengesetzte Begriffe enthalten, die unabhängig voneinander

erkannt werden würden, und inwieweit die Wortersetzungsregeln der Normalisierungsphase unterstützend definiert wurden. Einen Überblick über die in dieser Arbeit diesbezüglich umgesetzten Strukturierungsregeln gibt Anhang B.

3.9.3 Informationsgruppierung

Zusätzlich zum Zusammenfügen von logisch zusammengehörigen Begriffen werden zum einen einzelne Informationen zu Einheiten gruppiert. So kann beispielsweise aus einem Vornamen gefolgt von einem Nachnamen die Informationsgruppe `Person` gebildet werden. Dieser Gruppe werden dann weitere Informationen wie Berufe hinzugefügt. Zum anderen werden bestimmte Informationen anderen logisch zugeordnet, beispielsweise ein Namenszusatz wie „geborene“ einem Nachnamen. Dabei werden allgemein nur direkt aufeinander folgende Informationen zu einer Gruppe zusammengefasst bzw. einander zugeordnet.

Diese Gruppierungen werden — genauso wie das Zusammenfügen von Begriffen — anhand von priorisierten Strukturierungsregeln vorgenommen, die nutzerdefinierbar und beliebig erweiterbar sind. Eine Auflistung der in dieser Arbeit realisierten Regeln ist in Anhang B zu sehen. Wie bei der vorherigen Phase der semantischen Analyse spielt die Priorisierung der Regeln eine große Rolle. Je nachdem, in welcher Reihenfolge die Regeln angewendet werden, haben diese Auswirkung auf die entstehende Struktur.

Die verschiedenen sinnvollen Informationsgruppen und -zuordnungen, die auch anhand der in Kapitel 2.3 vorgenommenen manuellen Analyse des Volltextes identifiziert wurden, sind in Abbildung 3.24 ersichtlich. Jede Gruppe von Information beinhaltet dabei mindestens eine der angegebenen Teilinformationen. Das bedeutet, dass ein allein stehender Vorname ebenfalls eine eigenständige Person-Gruppe bildet. Die Zuordnung von Informationen zu anderen ist nicht zwingend. So muss beispielsweise einem Nachnamen kein Namenszusatz beigeordnet werden.

3.10 Abbildung auf Datenbanktabellen

Den Abschluss der Verarbeitung der Dokumente stellt die Abbildung der analysierten Informationen auf Datenbankstrukturen dar. Der konzeptionelle Datenbankentwurf und die dazu nötigen Vorbereitungen der XML-Dateien wird im Folgenden vorgestellt.

3.10.1 Grundlegende Ansätze zur Speicherung der Informationen

Prinzipiell kann man die extrahierten Daten auf zwei verschiedene Arten in den Tabellen speichern. Wurden beispielsweise in einem Grundbuchdokument zwei Personen mit exakt denselben Eigenschaften gefunden (beispielsweise Vorname „Hans“ und Nachname „Meier“), kann man diese beiden Personen wie folgt ablegen:

- Zum einen können die Daten **nichtredundant** gespeichert werden. Im Fall des Beispiels würde das bedeuten, dass in den entsprechenden Datenbanktabellen nur eine Person mit Vornamen „Hans“ und Nachnamen „Meier“ abgelegt wird, die durch diese Attribute eindeutig bestimmt ist. Dieser Datensatz wird dann zweimal referenziert.

- Zum anderen kann man die Daten **redundant** ablegen. Das heißt, dass in den Tabellen zwei identische Personendatensätze mit Vornamen „Hans“ und Nachnamen „Meier“ eingefügt

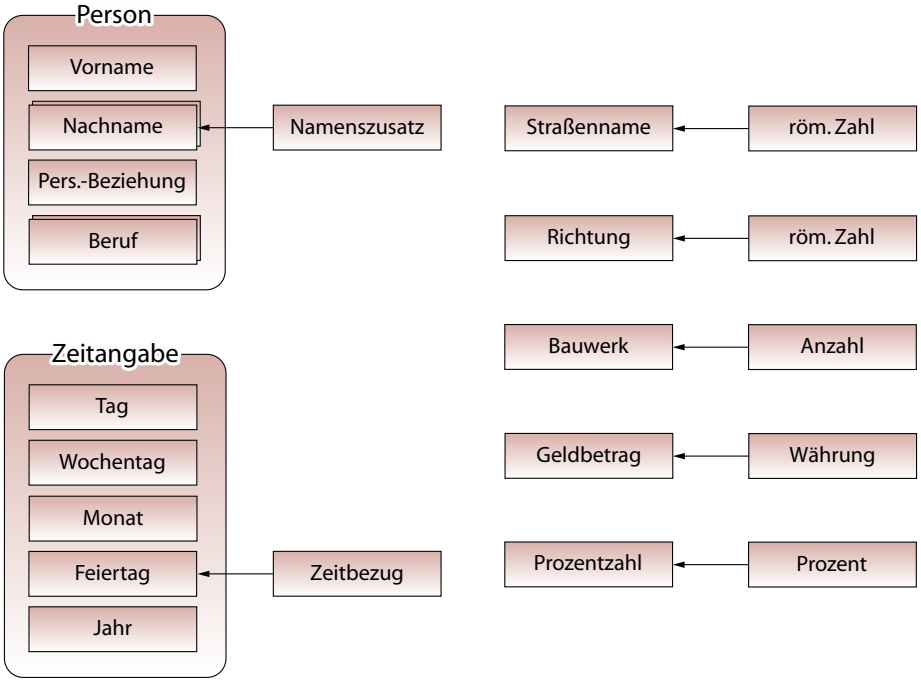


Abbildung 3.24: *Gruppierung von Informationen*

werden, die sich nur durch einen generierten Schlüssel unterscheiden. Jeder Datensatz wird dann jeweils nur einmal referenziert.

Beide Varianten bringen Vor- und Nachteile mit sich. Im ersten Fall wird natürlich die Anzahl der in den Relationen enthaltenen Tupel gering gehalten. Der Nachteil besteht jedoch hier in der komplexen Anpassung von Daten. Stellt sich zum Beispiel heraus, dass der Nachname der zweiten Person nicht „Meier“, sondern „Meyer“ geschrieben wird, muss ein neuer Personendatensatz angelegt und alle Referenzen entsprechend angepasst werden.

Im Gegensatz dazu muss bei der redundanten Speicherung der Informationen nur das Nachnamenattribut des zweiten Personendatensatzes entsprechend geändert werden. Die umständliche Anpassung von Referenzen ist in diesem Fall nicht nötig.

In dieser Arbeit wurde der zweite Ansatz zur Abbildung der extrahierten Informationen gewählt, da die leichte Änderbarkeit der gewonnenen Daten im Vordergrund steht. Der Nutzer soll auch nach der Speicherung der Informationen die Datensätze bequem anpassen können. Außerdem kommt in der heutigen Zeit dem Speicherplatzbedarf keine so wichtige Bedeutung mehr zu, sodass der Vorteil der nichtredundanten Speicherung weniger ins Gewicht fällt.

3.10.2 Globale ID-Vergabe

Wie im vorangegangenen Abschnitt erwähnt, werden bei der redundanten Speicherung die Informationen nicht durch sich selber eindeutig identifiziert, da mehrmaliges Vorkommen der gleichen Information sich auch in mehrmals auftretenden Datensätzen widerspiegelt. Aus diesem Grund müssen den einzelnen Informationen eindeutige Schlüssel zugeordnet werden. Da es sich bei den

extrahierten Informationen in diesem Stadium des Verarbeitungsprozesses um XML-Elemente bzw. –Attribute in mehreren XML-Dokumenten handelt, werden die Schlüssel dokumentübergreifend eindeutig vergeben. Ein Auszug eines um Schlüsselwerte angereichertes XML-Dokument ist in Abbildung 3.29 zu sehen.

3.10.3 ER-Modelle

Die Entity-Relationship-Modelle für die Daten des Abkürzungsverzeichnisses, der Personenindexe und der Grundbuchdateien sind in Abbildung 3.25 bis Abbildung 3.28 zu sehen. Im Fall der Grundbuchdaten stellen die Kandidaten der IST-Beziehung hauptsächlich diejenigen Informationen dar, welche durch die Wörterbücher und die semantischen bzw. Strukturierungsregeln identifiziert respektive strukturiert wurden. Die Erweiterung um neue Wörterbücher oder die Änderung der Informationsstrukturierung würde ebenfalls eine notwendige Erweiterung bzw. Anpassung der Entitäten bedingen. Ein konkretes Beispiel für die Umsetzung der XML-Elemente bzw. –Attribute auf die Entitäten stellt Abbildung 3.29 dar. Die aus den ER-Modellen abgeleiteten Relationenschemata sind in Anhang D ab Seite 114 zusammengefasst.

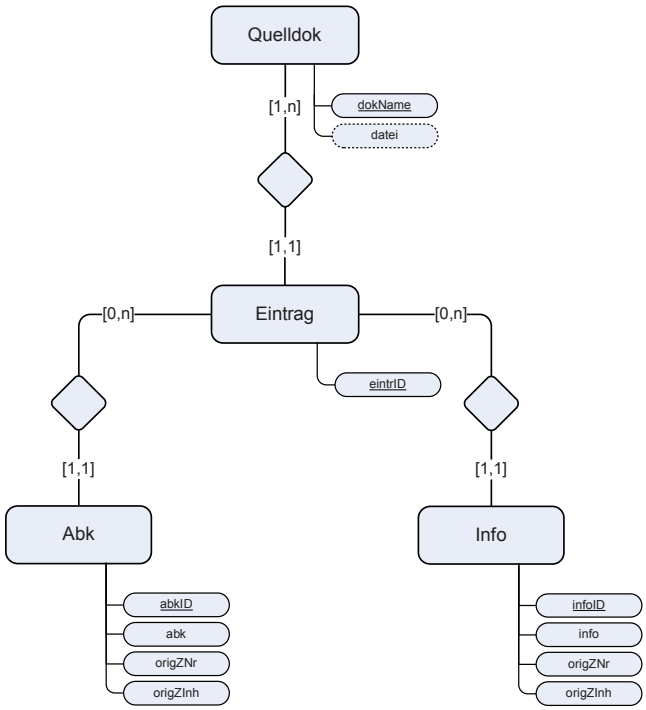


Abbildung 3.25: *ER-Modell für das Abkürzungsverzeichnis*

3.11 Prüfmechanismus

In den folgenden Abschnitten wird die Umsetzung eines Prüfmechanismus erläutert, der es dem Nutzer ermöglichen soll, aufgetretene Fehler oder Schwierigkeiten bei der Analyse der Eingabedokumente zu erkennen und darauf aufbauend eine manuelle Nachbehandlung der Dokumente oder

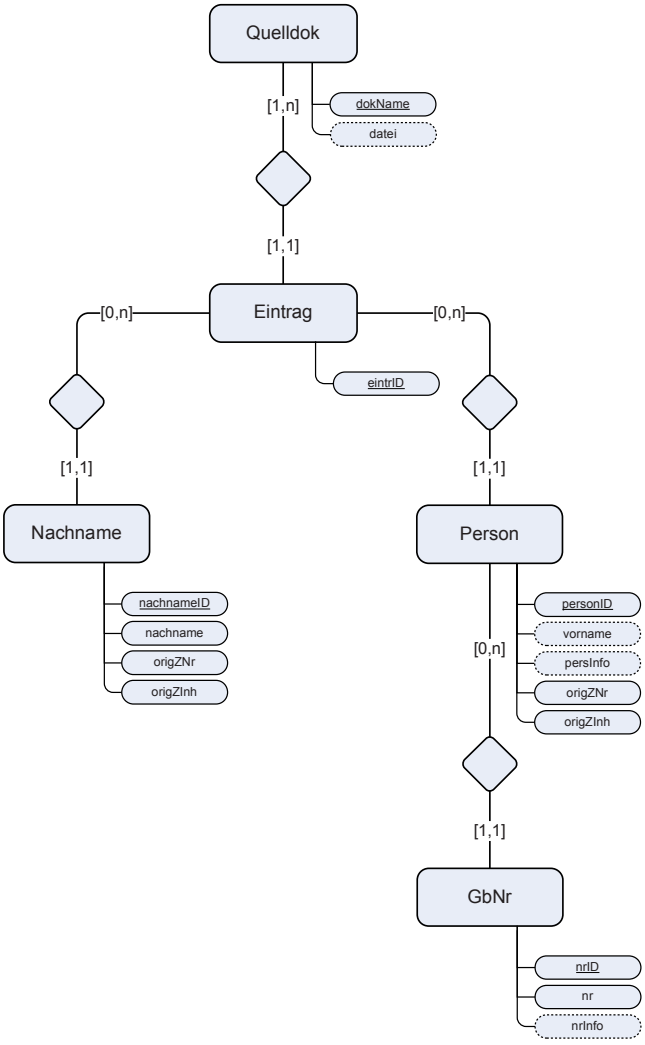


Abbildung 3.26: *ER-Modell für das Personenverzeichnis*

eine Anpassung der Wörterbücher, nutzerdefinierten Regeln und Einstellungen vorzunehmen.

3.11.1 Integration eines Logbuches

Für eine angemessene Dokumentation des Verarbeitungsprozesses der Eingabedateien ist ein Logbuch, das alle wichtigen Informationen aufzeichnet, unerlässlich. Das Logbuch in dieser Arbeit verfolgt dabei folgende Zwecke:

- Zum einen enthalten die Logbuchnachrichten **grundlegende Informationen über den Status des Prozesses**, in dem sich die Verarbeitung der Dokumente befindet. Dadurch bleibt der Nutzer jederzeit über den Fortgang des Ablaufs informiert und er kann einzelne Schritte besser nachvollziehen.
- Zum anderen werden alle **Warnungen und Fehlermeldungen**, die während der Abarbei-

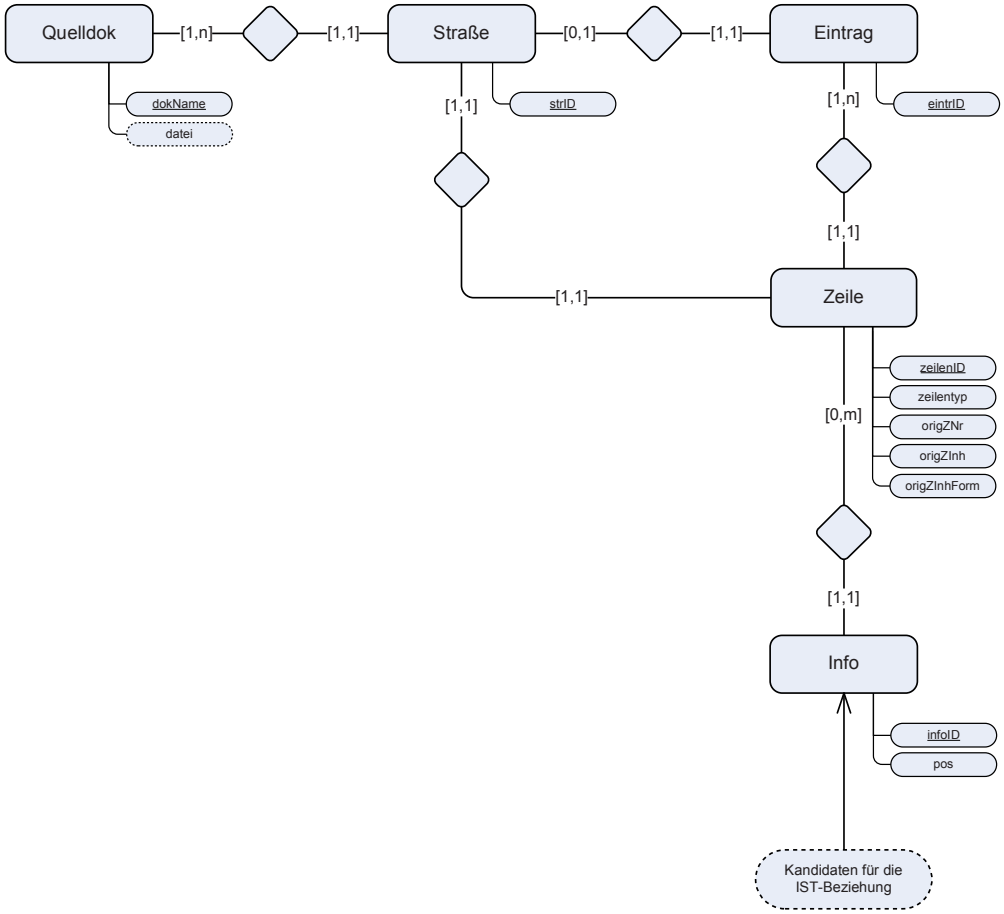


Abbildung 3.27: *ER-Modell für das Grundbuch*

tung auftreten, im Logbuch aufgezeichnet. So werden beispielsweise bei der Transformation der TXT- bzw. HTML-Dateien in eine XML-Struktur durch den Wrapper Warnungen ausgegeben, wenn einer Zeile der Eingabedatei kein Informationstyp zugeordnet werden konnte. Dieser Warnung wird die entsprechende Zeilennummer beigefügt, wodurch der Nutzer genau nachvollziehen kann, an welcher Stelle des Eingabedokuments der Fehler auftrat. Genauso werden bei der Analyse der regulären Dokumentanteile mit Hilfe von Grammatiken Fehlermeldungen erzeugt, sobald ein Eintrag nicht erkannt werden konnte. Durch diese detaillierten Logbuchnachrichten ist es dem Nutzer im Nachhinein möglich, die aufgetretenen Fehler nachzuvollziehen und eine entsprechende manuelle Nachbehandlung der Dokumente vorzunehmen.

- Weiterhin werden über Logbuchnachrichten wichtige **Statistiken** ausgegeben. So wird zum Beispiel nach der Wörterbuchanalyse eines Grundbuchdokuments aufgelistet, wie viele Token eindeutig einem Wörterbuch respektive keinem Wörterbuch zugeordnet und wie viele Token mehrdeutig erkannt wurden. Zu jedem Wörterbuch wird ebenfalls aufgeschlüsselt, wie viele der Token diesem zugeordnet wurden. Aufgrund dieser Statistiken ist es dem Nutzer

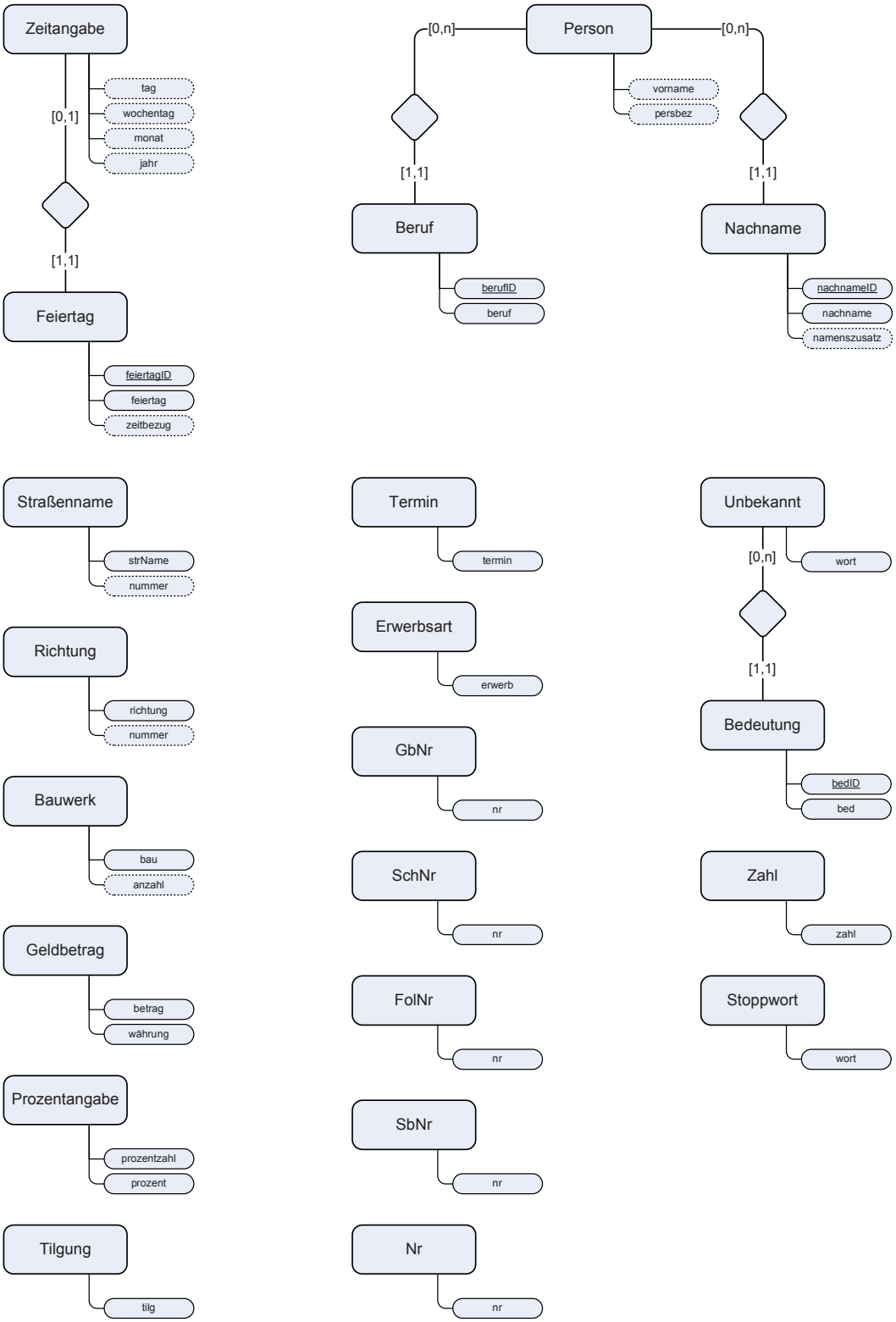


Abbildung 3.28: *Kandidaten für die IST-Beziehung*

```
<Eigentümer zeilenID="6" >
  <Person pos="1" infoID="24" >
    <Vorname value="Hinrich" />
    <Nachname value="Wiggers" nachnameID="1" />
  </Person>
  :
  :
  <OrigZNr value="4491" />
  <OrigZInh value="Hinrich Wiggers. empt. ibid." />
  <OrigZInhForm value="&lt;p&gt;Hinrich Wiggers. empt. ibid.&lt;/p&gt;" />
</Eigentümer>
```

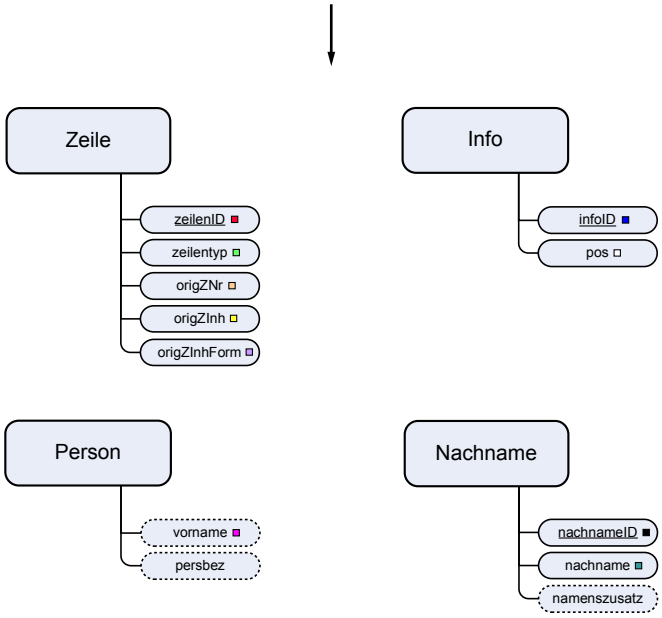


Abbildung 3.29: *Konkretes Beispiel für die Abbildung von XML-Daten auf Entitäten*

möglich zu überprüfen, ob die Wörterbücher eventuell erweitert oder anderweitig angepasst werden müssen. Außerdem lässt sich daran die Auswirkung der nutzerdefinierbaren Einstellungen aus Kapitel 3.7.10 auf Seite 57 erkennen. Darüber hinaus stellen diese Statistiken eine wichtige Quelle zur Identifikation von interessanten Daten für die spätere Visualisierung der extrahierten Informationen dar.

Um die Wirksamkeit der semantischen Analyse in Bezug auf die Eliminierung von Mehrdeutigkeiten und der Zuordnung von Bedeutungen zu nicht erkannten Inhalten zu kontrollieren, wird eine ähnliche Statistik am Ende der semantischen Auswertung eines Dokuments wiederholt. Zusätzlich dazu wird für jede semantische Regel vermerkt, wie oft diese erfolgreich angewandt wurde. Ebenso geschieht dies bei der Informationsstrukturierung mit den Strukturierungsregeln. Ein kurzer Auszug aus einer Statistik nach der semantischen Analyse ist in Abbildung 3.30 zu sehen.

```
INFO  ---------- Statistiken ----------

INFO  3361 von 4580 (73%) Token wurde eindeutig eine Bedeutung zugeordnet.

INFO  998 von 4580 (22%) Token konnte keine Bedeutung zugeordnet werden.

INFO  221 von 4580 (5%) Token wurden mehrere Bedeutungen zugeordnet.

INFO  232 von 4580 (5%) Token wurde die Bedeutung 'Erwerbsart' zugeordnet.

INFO  50 von 4580 (1%) Token wurde die Bedeutung 'Nummer' zugeordnet.

INFO  177 von 4580 (4%) Token wurde die Bedeutung 'Wochentag' zugeordnet.

INFO  721 von 4580 (16%) Token wurde die Bedeutung 'Vorname' zugeordnet.

INFO  472 von 4580 (10%) Token wurde die Bedeutung 'Nachname' zugeordnet.

INFO  82 von 4580 (2%) Token wurde die Bedeutung 'Beruf' zugeordnet.

INFO  6 von 4580 (0%) Token wurde die Bedeutung 'FolNr' zugeordnet.

[...]

INFO  354 von 4580 (8%) Token sind Stoppwörter.

INFO  313 von 4580 (7%) Token sind Zahlen.
```

Abbildung 3.30: *Beispiel einer Statistik aus dem Logbuch*

Anhand dieser Logbuchnachrichten lassen sich also der Ablauf des Verarbeitungsprozesses und die dabei eventuell aufgetretenen Fehler adäquat nachvollziehen, wodurch der Nutzer imstande ist, Fehler zu lokalisieren und entsprechende Korrekturen an nutzerdefinierbaren Regeln und anderen Einstellungen vorzunehmen.

3.11.2 Validierung der Ergebnisdokumente mit DTDs

Die in den einzelnen Schritten des Verarbeitungsprozesses entstehenden XML-Dokumente müssen einer bestimmten logischen Struktur genügen, um gültige Ergebnisse darzustellen. Während des Verarbeitungsprozesses können durch Parserfehler oder sonstige Schwierigkeiten bei der Analyse der Informationen inkorrekte Strukturen entstehen. Für jeden Dokumenttyp werden dazu DTDs bereitgestellt, anhand derer sie nach jedem Verarbeitungsschritt einer Validitätsüberprüfung unterzogen werden. Die dabei auftretenden Inkonsistenzen werden dem Nutzer über Logbuchmeldungen mitgeteilt, aufgrund derer er zusätzlich zu den übrigen Fehlerhinweisen aus der Dokumentverarbeitung in der Lage ist, die Fehlerquelle zu lokalisieren und zu bereinigen.

3.11.3 Überprüfung der Informationsstrukturen anhand von Regeln

Während des Verarbeitungsprozesses können dem Nutzer unerwünschte Informationsstrukturen entstehen. Diese Zustände können wiederum durch Regeln beschrieben werden, welche auf ein Dokument angewandt werden und als Ergebnis eine Auflistung aller XML-Elemente mit den jeweils identifizierenden XPath-Ausdrücken liefern, die dieser Regel entsprechen. So wäre es beispielsweise denkbar, dass eine durch die Informationsstrukturierung gebildete Person, die lediglich aus einem

Vornamen besteht, als unerwünschte Informationsausprägung angesehen wird und daher weiterer manueller Behandlung bedarf.

Da diese Beschreibungen der unerwünschten Informationsstrukturen stark abhängig vom Anwendungsgebiet sind und auch durch die Regeldefinition der Informationsstrukturierung beeinflusst wird, wurden in dieser Arbeit beispielhaft nur einige Regeln umgesetzt, die in Anhang C aufgelistet sind.

Die definierten Regeln bedürfen im Gegensatz zu den semantischen und Strukturierungsregeln keiner Priorität, sind aber genauso uneingeschränkt erweiterbar. Dadurch ist eine weitere flexible Möglichkeit gegeben, die Eingabedokumente zu überprüfen und aufbauend auf den Ergebnissen eine manuelle Nachbehandlung vorzunehmen.

Kapitel 4

Umsetzung & Implementierung

Nachdem im vorangegangenen Kapitel der konzeptionelle Gesamtablauf beschrieben wurde, soll in den nächsten Abschnitten auf die konkrete Realisierung eingegangen werden. Dabei werden die eingesetzten externen Werkzeuge und verschiedenen Techniken der Implementierung vorgestellt.

4.1 Konvertierung von RTF bzw. DOC in TXT

Wie im vorangegangen Kapitel bereits erwähnt, wurden die Personenindexe und das Abkürzungsverzeichnis in reine Textdateien überführt. Dieses wurde manuell mit MS Word vorgenommen, da das Aussehen der Ergebnisdatei klar definiert und somit keine separate Konvertierungs-Software notwendig ist. Zudem ist diese Aktion durch den Endanwender auch leicht selber über den entsprechenden „Speichern“Dialog im jeweiligen Textverarbeitungsprogramm durchzuführen. Aus diesen Gründen wurde auf eine automatische Konvertierung dieser Dateien verzichtet.

4.2 Konvertierung von RTF in HTML

Um RTF-Dokumente in HTML-Dateien unter Erhaltung aller Schriftformatierungen zu transformieren, gibt es nun verschiedenen Möglichkeiten, von denen einige im Folgenden kurz vorgestellt werden sollen.

Konvertierung mit Microsoft Office Word 2003

Word bietet in seinem Programmumfang auch das Speichern von Texten im HTML-Format an. Dabei werden alle Formatierungen des Textes korrekt übernommen, jedoch fügt Word sehr viele Microsoft spezifische Tags und Style-Angaben ein. Diese Office eigenen Angaben dienen dazu, beim späteren Bearbeiten des Textes mit Word auf alle Word-Funktionen zugreifen zu können.

Um einen Text in ein kompakteres HTML-Format umzuwandeln, gibt es in Word darüber hinaus die Möglichkeit, den Inhalt als gefiltertes HTML-Dokument zu speichern. Dabei werden die von Microsoft-Office-Programmen verwendeten Tags entfernt. Einige Word-Funktionen können dadurch beim späteren Bearbeiten der Datei jedoch möglicherweise nicht wie gewohnt verwendet werden.

In Abbildung 4.1 ist der Quelltext zu sehen, der entsteht, wenn man das Beispiel für ein RTF-Dokument aus Abbildung 3.2 auf Seite 26 mittels Word als gefilterte HTML-Datei speichert. Sogar bei Anwendung des gefilterten Formats auf einen sehr kurzen Ausgangstext entsteht ein relativ umfangreicher HTML-Klartext mit vielen Style-Angaben, die auch in kompakterer Form umgesetzt werden könnten. Diese Tatsache und der Aspekt, dass die HTML-Dateien später nicht mit Word bearbeitet werden müssen und somit keine Office spezifischen Tags im Quelltext nötig sind, führten zu der Entscheidung gegen MS Word für die Transformation der RTF-Ausgangsdokumente.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
    <meta name="Generator" content="Microsoft Word 11 (filtered)">
    <title>This is plain text</title>
    <style>
      <!--
      /* Font Definitions */
      @font-face
      {font-family:"Tms Rmn";
      panose-1:2 2 6 3 4 5 5 2 3 4;}
      /* Style Definitions */
      p.MsoNormal, li.MsoNormal, div.MsoNormal
      {margin:0cm;
      margin-bottom:.0001pt;
      text-autospace:none;
      font-size:10.0pt;
      font-family:"Tms Rmn";}
      /* Page Definitions */
      @page Section1
      {size:612.0pt 792.0pt;
      margin:70.85pt 70.85pt 2.0cm 70.85pt;}
      div.Section1
      {page:Section1;}
      -->
    </style>
  </head>
  <body lang="DE" style="text-justify-trim:punctuation">
    <div class="Section1">
      <p class="MsoNormal"> </p>
      <p class="MsoNormal"> </p>
      <p class="MsoNormal"><b>This is plain text.</b></p>
    </div>
  </body>
</html>
```

Abbildung 4.1: *RTF-HTML-Konvertierung mit Word*

Optimierung von Word-HTML mit Macromedia Dreamweaver MX Version 6.0

Mit dem Dreamweaver kann der Nutzer Dokumente öffnen oder importieren, die in Microsoft Word als HTML-Dateien gespeichert wurden. Anschließend kann man den Word-HTML-Code optimieren, um die von Word erstellten überflüssigen Angaben zu entfernen. Der Code, der von Dreamweaver gelöscht wird, dient in Word, wie im vorherigen Abschnitt schon erwähnt, hauptsächlich zur Formatierung und Anzeige des Dokuments und ist für die Betrachtung der HTML-Datei nicht erforderlich. Bei der Optimierung sind dabei im Dreamweaver vielfältige Optionen verfügbar, mit denen die Anpassung des Quelltextes beeinflusst werden kann.

Das Ergebnis der Dreamweaver-Optimierung des Beispiels aus der vorangegangen Darstellung ist in Abbildung 4.2 zu sehen. Der Unterschied in Bezug auf die Kompaktheit des Quelltextes ist deutlich zu erkennen. Da aber beide Programme, Word und Dreamweaver, kommerzielle Software darstellen und nicht davon ausgegangen werden kann, dass diese Werkzeuge jedem zur Verfügung stehen, wurde in dieser Arbeit auf deren Einsatz verzichtet.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>This is plain text</title>
<style><!--
.MsoNormal
  {text-autospace:none;
  font-size:10.0pt;
  font-family:"Tms Rmn";}
.Section1
  {page:Section1;}
-->
</style>
</head>
<body bgcolor="#FFFFFF" class="Normal" style="text-justify-trim:punctuation" lang
    ="DE">
<div class="Section1">
  <p class="MsoNormal"><b>This is plain text.</b></p>
</div>
</body>
</html>
```

Abbildung 4.2: *Word-HTML mit Dreamweaver optimiert*

Kommerzielle RTF-Konverter

Auf dem Markt gibt es verschiedenste kommerzielle Konverter, die RTF-Dokumente in HTML und teilweise auch in andere Formate transformieren. Nur ein Beispiel ist **R2Net** von Logictran [Log03], der das RTF-Format in HTML, XHTML, DocBook oder OeB unwandelt. Darüber hinaus kann man dem Produkt Unterstützung für eigene Formate hinzufügen. R2Net ist laut Herstellerangaben sehr anpassungsfähig und beinhaltet mehrere Bibliotheken zum Einbetten von R2Net in eigene Systeme oder Web-Seiten.

Convert Doc von Softinterface, Inc. [Sof04] ist ein weiteres kommerzielles Tool, das unter der Einschränkung, ständig einen neuen Freischaltungscode beziehen zu müssen, auch kostenlos genutzt werden kann. Convert Doc unterstützt die Umwandlung mehrerer Formate, darunter PDF, MS Word, RTF und HTML. MS Word und Adobe Acrobat müssen dafür nicht auf dem System vorhanden sein. Bei der Auswahl des Transformationsalgorithmus kann man sich zwischen dem programminternen und der MS-Word-Konvertierung entscheiden. Außerdem bietet Convert Doc weitere nützliche Funktionen, wie beispielsweise die Konkatenation von Dateien, Entfernen leerer Zeilen, Kopieren, Löschen und andere Dateioperationen, String-Suche und –Ersetzung. Darüber hinaus können komplexe Konvertierungskonfigurationen als Job gespeichert und über die GUI[33] oder die Kommandozeile ausgeführt werden.

Obwohl es auf dem Markt zahlreiche Konvertierungstools mit vielen nützlichen Funktionen gibt, wird in dieser Arbeit auf den Einsatz von kommerziellen Werkzeugen verzichtet und

[33] Abkürzung für *Graphical User Interface*.

die Realisierung mit Unterstützung von externen Programmen, die für jeden frei verfügbar sind, angestrebt.

Nichtkommerzielle RTF-Konverter

Neben den kommerziellen Konvertern gibt es auch einige wenige nichtkommerzielle Lösungen, die für jeden frei verfügbar sind. Ein Beispiel dafür ist das **PIPER Tool Kit** [PIP], dessen Hauptaufgabe darin besteht, die Konformität von RTF-Dokumenten bezüglich bestimmter Guidelines zu überprüfen. Neben dieser Funktion ist außerdem die Transformation von RTF-Dateien der Version 1.5 oder früher in HTML-Dokumente möglich. Dabei werden alle Schriftformatierungen korrekt auf HTML-Strukturen abgebildet. Darüber hinaus kann der Nutzer eine Textdatei mit einer Liste von zu konvertierenden Dokumenten und mehreren Optionen anlegen, was die Transformation vieler Dateien komfortabler gestaltet. Ein Nachteil bei der Umwandlung in das HTML-Format ist die Tatsache, dass das PIPER Tool Kit aus einem Eingabedokument jeweils zehn Dateien generiert, wobei nur eine davon den eigentlichen Inhalt enthält und der Rest Bilder, Frame-Dateien etc. darstellt. Außerdem ist die Einbindung des Programms in andere Anwendungen über die Kommandozeile generell möglich, aber dennoch nicht vollkommen transparent gegenüber dem Benutzer, da die GUI trotz der zuvor angelegten und als Parameter übergebenen Konvertierungsliste Nutzereingaben erwartet.

Ein weiterer nichtkommerzieller RTF-Konverter kann im Verzeichnis des Free Software Directory unter [FU] gefunden werden. Das FSD ist ein Projekt der Free Software Foundation (FSF) und der United Nations Education, Scientific and Cultural Organization (UNESCO) und katalogisiert nützliche freie Software insbesondere für das GNU-Betriebssystem und seinen GNU/Linux-Varianten. Das dort bereitgestellte Tool **RTF to HTML** konvertiert RTF-Dokumente in HTML-Dateien und unterstützt dabei unter anderem fette, kursive und unterstrichene Schrift, Tabellen, links- und rechtsbündige Ausrichtungen sowie zentrierten Text. Das Tool verfügt über keine graphische Benutzerschnittstelle, sondern wird ausschließlich über die Kommandozeile betrieben. Außerdem ist zur Ausführung des Programms eine PHP[34]-Installation auf dem System nötig. Da jedoch die Konvertierung der Ausgangsdokumente dieser Arbeit mit RTF to HTML zum Abbruch führte, was anscheinend an der Größe der RTF-Dateien lag, wurde dieses Tool nicht eingesetzt.

Realisierte Umsetzung

Aufgrund der Nachteile der oben beschrieben Konvertierungsmöglichkeiten kam in dieser Arbeit das frei verfügbare Tool **rtf-converter**, das ebenfalls im Verzeichnis des Free Software Directory unter [FU] bereitgestellt ist, zum Einsatz. Die Software wurde für das Umweltministerium von Neuseeland als Teil des Kaitiaki-Web-Site-Projektes entwickelt. Der in dem Tool integrierte RTF-Token-Parser basiert auf rtf2html von Dimitry Porapov, während die RTF-Token-Listen und der Hashing-Algorithmus an die Übersetzer-Sammlung RTF Tools, Release 1.10 von Paul DuBois angelehnt sind.

Der Konverter, der ausschließlich über die Kommandozeile aufgerufen wird, transformiert RTF in HTML, wobei jedoch lediglich der Body-Code generiert wird und die Body-Tags und der

[34]PHP (rekursives Akronym für *PHP: Hypertext Preprocessor*, ursprünglich *Personal Home Page Tools*) ist eine Skriptsprache mit einer an C bzw. Perl angelehnten Syntax, die hauptsächlich zur Erstellung dynamischer Web-Seiten verwendet wird. [Wik]

HTML-Header selbst hinzugefügt werden müssen. Der rtf-converter unterstützt folgende Konvertierungen:

- Überschriften und normale Formatierungen (Umsetzung mit `h`- bzw. `p`-Tags),
- Zeichenformatierungen fett, kursiv und unterstrichen (Umsetzung mit `b`-, `i`- bzw. `u`-Tags),
- Tabellen (Abbildung auf HTML-Tabellen) und
- Fußnoten (werden am Ende des Dokuments platziert).

Schriftinformationen, Farben, Einrückungen und Graphiken werden nicht umgesetzt, sodass diese Formatierungen nachträglich eingefügt werden müssen.

Die Einbettung dieses Tools in eigene Anwendungen gestaltet sich durch den Aufruf über die Kommandozeile als sehr transparent gegenüber dem Nutzer. Da das Programm keine GUI startet, wird somit der eigentliche Programmablauf nicht gestört. Weiterhin überzeugen die äußerst geringe Verarbeitungsdauer und das Konvertierungsergebnis, denn alle für die Layoutanalyse wichtigen Formatierungen werden korrekt umgesetzt. Die Tatsache, dass der generierte HTML-Code nur den reinen Body-Inhalt und nicht das gesamte HTML-Gerüst enthält, erleichtert das Parsen in der nächsten Stufe und ist in keiner Weise hinderlich, da alle gängigen Browser diese HTML-Seiten trotzdem fehlerlos anzeigen. Der generierte HTML-Code zum Beispiel in Abbildung 2.3 auf Seite 17 ist in Abbildung 4.3 zu sehen.

```
<p><b>Alt Wismarsche Strasse vom thor her. Norderseite</b></p>
<p><i>1 <b>Grundbuch Nr. 16 </b>(</i>16<i>), fol. 15v, neues Stadtbuch Nr. 196</i
    ></p>
<p><i>2 siehe Grundbuch Nr. 15.</i></p>
<p><i>3 </i>Haus</p>
<p><i>4 </i>jus protimiseos an dem dahinten belegenen Garten. Medard. <i>1</i>673
    .</p>
<p><i>5 </i>ut No. praeced. <i>siehe Grundbuch Nr. 15.</i></p>
<p>Peter Koppe. adjud. et aedif. Tr. Regum 1657.</p>
<p>Georg Gammelkern. empt. Medard. 1673.</p>
<p>Johan Faber. empt. Vener. ante Jacobi 1693.</p>
<p>Hans Martensson. empt. Vener. post Johannis Baptist.</p>
<p>Gabriel Froh. Emti. Ven. a. Philip. Jac. 1731.</p>
<p>de&szlig;elben Erben. hered. <i>Eod.</i></p>
<p>Benedix Ohrbahn. Emti. <i>Eod.</i></p>
<p>Jacob Kobau. Eod.</p>
<p>de&szlig;en Kinder. hered. ven. post Margarethae 1758.</p>
<p>Johann Peter Wehberg. emt. <i>Eod.</i></p>
<p>de&szlig;en Witwe und Erben in spec. de&szlig;en altesten Sohn Joh. Peter
    Wehberg. Transact. et emt. v. a. Purif. Mar. 1794.</p>
<p>Ins neue Stadtbuch &uuml;bertragen d. 28. Mai 1853.</p>
<p><i>6 </i>400 Rthlr. Joh. Christoff M&uuml;ller. Clem. 1677. delirt Mattaei
     1688.</p>
<p>200 Rthlr. M. Thomas Baltzers. <i>Clem. 1677</i>. delirt Concept. Mar. 1686.</
    p>
<p>600 m. l. die Stadt C&auml;mmerey t. Joh. mit 5 procent zu vertzinsen.</p>
<p>300 m. werden hievon getilget. V. a. Phil. Jacob. 1731.</p>
<p>die andern 300 m. geh&ouml;ren der Patrimonial Cassa. ven. a. Viti 1749.</p>
<p>delet. v. a. Rogate 1797.</p>
<p>2 rthl. j&auml;rlich C&auml;merey. Medard. 1673. delet. Margareth. 1731.</p>
<p>---</p>
<p>Contig. Oldeb&ouml;terstr. Ost <i>Grundbuch Nr. 310</i>.</p>
<p>---</p><hr>
```

Abbildung 4.3: *Konvertierungsergebnis des eingesetzten RTF-Konverters*

4.3 Generierung von XML aus den Eingabedokumenten mit Wrappern

In der Phase der Layout- und Strukturanalyse werden, wie im vorangegangenen Kapitel beschrieben, die TXT- bzw. HTML-Dateien in XML-Strukturen transformiert, die den Informationen bereits eine erste Beschreibung der Daten zuordnen. Wie diese XML-Dateien konkret generiert werden, soll Thema der folgenden Abschnitte sein.

Allgemeines zu Wrappern

Unter einem Wrapper[35] versteht man ein Programm zur Konvertierung von Daten und Anfragen zwischen zwei verschiedenen Datenmodellen [Goh00]. Die Aufgaben eines Wrappers lassen sich dabei in drei grundlegende Phasen unterteilen (siehe auch [Lan02]):

- **Zugriff**
  In diesem Schritt wird auf die spezifizierte Datenquelle zugegriffen. Im Anwendungsgebiet des WWW erfolgt hier das Abfragen der Web-Seite einer WWW-Ressource. Das Ergebnis ist in diesem Fall ein HTML-Dokument.

- **Extraktion**
  In der zweiten Phase erfolgt die Suche, Erkennung und Extraktion der gewünschten Informationen.

- **Ausgabe**
  Der letzte Schritt bildet die gewonnenen Daten auf ein nutzerdefiniertes, geeignet strukturiertes Ausgabeformat ab, um die Informationen anderen Anwendungen zur weiteren Manipulation zur Verfügung zu stellen.

Diese einzelnen Schichten werden durch eine Menge von Spezifikationsregeln beschrieben. Eine Anforderung an den Wrapper ist dabei, dass die Extraktion der Daten an deren Granularität anpassbar sein muss. Das bedeutet, dass zum Beispiel die Datenstruktur bei HTML-Dokumenten einerseits anhand der HTML-Struktur, andererseits aber auch durch die Textstruktur innerhalb der Tags analysierbar sein muss. Dies leisten zum Beispiel reguläre Ausdrücke [Goh00]. Eine Übersicht über die prinzipielle Vorgehensweise von Wrappern bietet Abbildung 4.4.

Ein großes Anwendungsgebiet von Wrappern ist das WWW, das sich in den letzten Jahren zu einer wichtigen Informationsquelle entwickelt hat. Beispiele für Anwendungen sind Online-Preisvergleiche, die Überwachung aktueller Aktienkurse oder die individuelle Zusammenstellung von Nachrichtenmeldungen. Dabei basieren über 80% der publizierten Informationen auf Daten, die im Hintergrund in Datenbanken abgelegt sind. Wenn aus diesen Daten HTML-Dokumente generiert werden, geht die Struktur der zugrunde liegenden Datenbank verloren [SA99]. Wrapper versuchen, diesen Prozess umzukehren, indem sie die Informationen wieder in einem strukturierten Format speichern und somit das Konzept des Reverse Engineering verwirklichen [SA01].

Für jede Datenquelle wird aufgrund der einzigartigen Struktur des Eingabedokuments ein speziell dafür entwickelter Wrapper benötigt. Besonders im Bereich des WWW müssen Wrapper wegen der Dynamik und der kontinuierlichen Weiterentwicklung des Internets ständig angepasst

[35] Englisch für *einhüllen*; *packen*; *umwickeln*; *bedecken*.

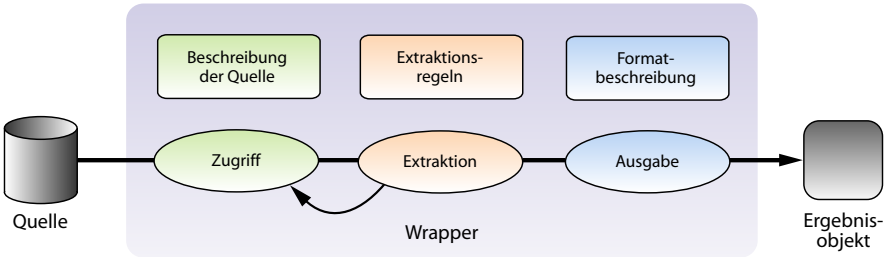


Abbildung 4.4: *Phasen eines Wrappers nach [Lan02]*

oder sogar neu entwickelt werden. Aufgrund dieses hohen Anpassungsaufwandes und der Tatsache, dass die Realisierung eines Wrappers aufgrund der Eingabestrukturen schnell sehr komplex werden kann, ist die manuelle Erstellung — zum Beispiel mit einer Programmiersprache und regulären Ausdrücken — sehr aufwändig und nur für kleinere Anwendungen sinnvoll. Toolkits helfen dabei, basierend auf nutzerdefinierten Regeln und Parametern komplette Wrapper für eine gegebene Datenquelle zu generieren. Diese Toolkits können anhand verschiedener Merkmale klassifiziert werden, zum Beispiel aufgrund der Ausgabemethoden, der Web-Crawling-Fähigkeiten oder dem Einsatz einer GUI [KT02]. Unter [LRS+02] ist eine Kategorisierung mehrerer Toolkits basierend auf den Erzeugungsmethoden von Wrappern zu finden. Ebenso bietet [KT02] eine kurze Übersicht über kommerzielle und nichtkommerzielle Wrapper-Toolkits mit ihren grundlegenden Merkmalen. Im Rahmen dieser Arbeit soll anschließend lediglich eine kurze Vorstellung des Wrapper-Generators W4F folgen, wonach auf das in dieser Arbeit eingesetzte Toolkit X-Fetch Wrapper näher eingegangen wird.

W4F[36]

W4F steht für WysiWyg Web Wrapper Factory und wurde an der Universität Pennsylvania als kommerzielles Toolkit zur Generierung von Wrappern für WWW-Datenquellen basierend auf Java entwickelt. Das Toolkit umfasst folgende Komponenten:

- Der **W4F-Parser** dient der Analyse von HTML-Dokumenten. Der Parser ist fehlertolerant, kann also auch fehlerhafte HTML-Seiten teilweise bearbeiten.
- Der **W4F-Compiler** generiert eine Java-Klasse zu einer Wrapper-Spezifikation.
- Das **W4F-Laufzeitmodul** erlaubt die Ausführung des generierten Java-Wrappers als eigenständiges Programm.
- Verschiedene **Wizards** unterstützden den Nutzer beim Wrapper-Entwurf.

Die prinzipielle Arbeitsweise der generierten Wrapper folgt einer klar nach Aufgaben modularisierten Schichtenarchitektur. Die Unabhängigkeit der Schichten voneinander erleichtert dabei die Änderbarkeit und Wiederverwendbarkeit der Wrapper-Spezifikationen. Die drei Schichten sind in Abbildung 4.5 dargestellt und seien im Folgenden näher erläutert:

[36] Die Informationen hierzu stammen aus [Goh00, SA98].

- **Retrieval-Schicht**
  In der Retrieval-Schicht wird das durch Retrieval-Regeln spezifizierte HTML-Dokument geladen, welches dann die Eingabe für die Extraktionsschicht darstellt. Die Regeln legen dabei fest, welche HTML-Seite mit welchen Parametern von welcher Web-Adresse mit welcher HTTP-Übertragungsmethode geholt werden soll. Es kann jedoch immer nur ein HTML-Dokument geladen werden.

- **Extraktionsschicht**
  Nach dem Laden der HTML-Seite beginnt der Parser automatisch mit der Erstellung des Analysebaums. Die Zuordnung von Dokument und Baum ist dabei eineindeutig. Der HTML-Baum besteht aus folgenden Elementen:

  – der Wurzel (mit dem Namen `html`),
  – internen Knoten (repräsentieren geschlossene HTML-Tags, können Attribute und Kinder haben) und
  – Blättern (repräsentieren offene HTML-Tags und den eigentlichen Textinhalt des Dokuments, können Attribute haben).

  Die Identifikation und der Zugriff auf die Knoten und Blätter erfolgen über Pfadausdrücke. Eigenschaften der Knoten können über Funktionen erfragt werden. Die Extraktionsregeln werden in der kleinen, aber mächtigen deklarativen Spezifikationssprache HEL[37] mit Hilfe dieser Pfadausdrücke und regulären Ausdrücke definiert. Als Ergebnis werden Attributwerte bestimmter Knoten im intern verwendeten NSL-Format[38] geliefert. Das NSL-Format stellt eine geschachtelte Liste von String-Listen dar und ist wie folgt definiert:

  ```
  NSL = null | string | listOf(NSL).
  ```

- **Abbildungsschicht**
  In der letzten Schicht erfolgt die Abbildung des NSL-Formats in eine nutzerdefinierte Zielstruktur. Diese kann von einem Java-Standarddatentyp sein oder ein vom Nutzer implementiertes Java-Objekt darstellen. Die Abbildung kann dabei jedoch nur für jeweils eine Extraktionsregel angegeben werden.

Zur Integration des Wrapper Toolkits in andere Anwendungen bietet das W4F außer den Wizards auch eine Java-API[39]. Durch die klare Trennung der Schichten nach ihren Aufgaben und die kleine, aber mächtige HEL ist die Verständlichkeit, Wiederverwendbarkeit und einfache Änderbarkeit der Wrapper-Spezifikationen gesichert. Ein Nachteil dabei ist eventuell, dass nicht mehrere WWW-Datenquellen gleichzeitig abgefragt oder Daten aus mehreren Quellen extrahiert und zu einer gemeinsamen Struktur zusammengefügt werden können.

Das Wrapper Toolkit W4F wurde schon in vergangenen Diplomarbeiten erfolgreich eingesetzt. Da es aber schon seit einiger Zeit nicht mehr verfügbar ist und auch auf Anfrage nur eine begrenzte Nutzungsmöglichkeit gewährt wurde, kam das W4F in dieser Arbeit nicht zum Einsatz.

[37] Abkürzung für *HTML Extraction Language*.
[38] Abkürzung für *Nested String List*.
[39] Abkürzung für *Application Programming Interface*.

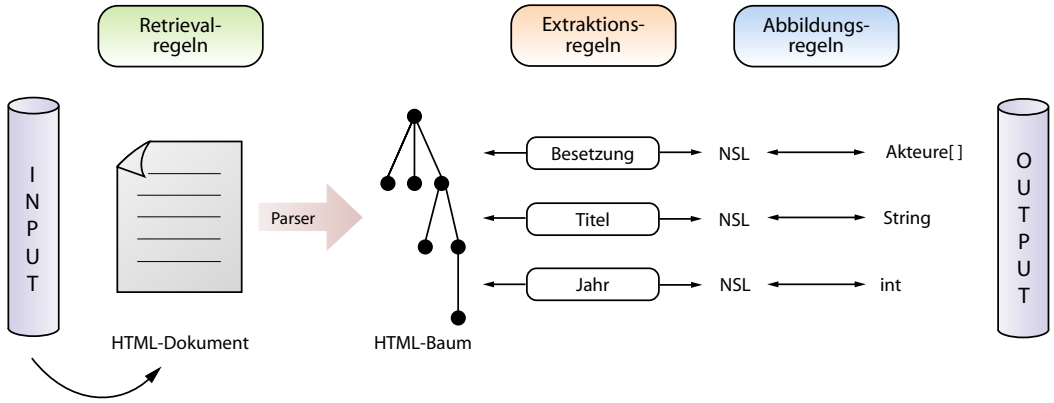


Abbildung 4.5: *Architektur des W4F nach [Goh00]*

X-Fetch Wrapper — Version 2.2

Der X-Fetch Wrapper ist Teil der X-Fetch Suite der Republica Corporation[Rep], welche weiterhin den X-Fetch AgentServer, X-Fetch Performer und X-Fetch Unifier beinhaltet und bei wissenschaftlichem oder nichtkommerziellem Einsatz ohne Einschränkungen kostenlos genutzt werden kann. Die Funktionalität des X-Fetch Wrappers umfasst die Konvertierung jedes ASCII-, Unicode-basierten oder binären Eingabeformats in jedes XML-basierte Format (XML, WML, XHTML, ebXML), welche durch den so genannten DEL-Prozessor durchgeführt wird. Die Architektur des Wrappers ist in Abbildung 4.6 dargestellt.

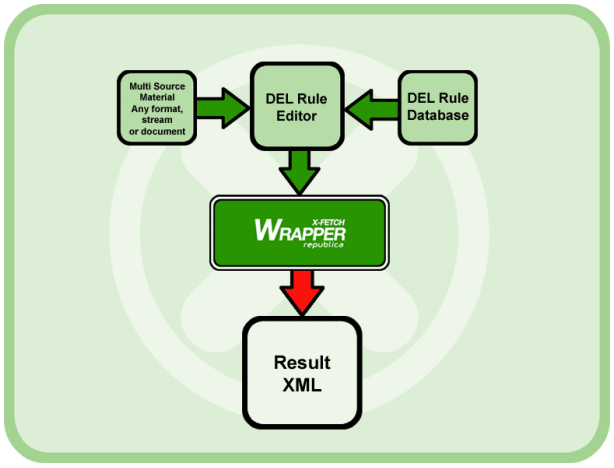


Abbildung 4.6: *Architektur des X-Fetch Wrappers [Rep]*

Der X-Fetch Wrapper benutzt eine eigene Skriptsprache zur Spezifizierung der Extraktions- und XML-Konvertierungsregeln. Diese DEL[40] wurde bereits vom W3C auf seiner Referenzliste als erster Beitrag im Bereich der XML-basierten Datenextraktionssprachen akzeptiert. Mit den Elementen dieser Sprache legt der Nutzer fest, welche Daten aus der Eingabedatei extrahiert werden sollen

[40] Abkürzung für *Data Extraction Language*.

und wie das Ausgabeformat auszusehen hat. Damit bleibt das generierte Ergebnisdokument frei definierbar.

Die Sprache ist intuitiv und leicht zu erlernen, denn es existieren wenige — unter 40 — logisch gewählte Schlüsselwörter und mit der Kenntnis einiger lassen sich bereits einfache Konvertierungen vornehmen. Die Sprache folgt einer XSLT-ähnlichen Syntax und bietet im Bereich der Extraktionsregeln die Möglichkeiten der String-Erkennung, regulären Ausdrücke und die Extraktion fester Zeichenlängen, welche auch untereinander beliebig kombiniert werden können. Für die Erzeugung des Ausgabeformats stehen unter anderem auch Regeln für einfache mathematische Berechnungen, bedingte Anweisungen und Schleifen zur Verfügung. So genannte Register können zum Speichern von Werten eingesetzt werden, womit ein Pendant zu Variablen in Programmiersprachen realisiert ist. Außerdem ist es möglich, mehrere Ausgabedokumente von einer Eingabedatei zu erzeugen.

Wie das im vorangegangenen Abschnitt beschriebene WebL realisiert der X-Fetch Wrapper kein modulares Schichtenkonzept. Jedoch lässt sich hier ebenfalls eine logische Trennung der einzelnen Komponenten anhand der DEL-Sprachelemente erkennen, deren Anweisungen sich in Extraktions-, Abbildungs- und programmiersprachenähnlichen Regeln (Schleifen, bedingte Anweisungen etc.) kategorisieren lassen. Im Gegensatz zu den vorangegangenen beiden Toolkits wird vom X-Fetch Wrapper das Eingabedokument vor der Verarbeitung jedoch nicht in eine interne abstrakte Repräsentation überführt, sondern als eine große Zeichenkette angesehen, die sequentiell eingelesen und ausgewertet werden kann. Dieser Umstand ermöglicht die Flexibilität des Wrappers, nahezu jedes Eingabeformat zu behandeln und auf eine nutzerdefinierte Zielstruktur abzubilden. Das Web-Crawling ist dabei nur im Zusammenhang mit dem in der X-Fetch Suite enthaltenen X-Fetch AgentServer möglich, dessen Verwendung in dieser Arbeit jedoch nicht notwendig war.

Im Kern ist der X-Fetch Wrapper eine Java-Komponente und bietet daher dem erfahrenen Nutzer eine API, um die Konvertierung von Dateien in bestehende Anwendungen zu integrieren. Der Wrapper kann weiterhin über die Kommandozeile oder die mitgelieferte GUI, die ebenfalls einen komfortablen Editor zur Erstellung der DEL-Skripte beinhaltet (siehe Abbildung 4.7), betrieben werden. Im Bereich des Loggens und Debuggens besteht die Möglichkeit, verschiedene Loglevel zu setzen und mit Hilfe von DEL-Regeln eigene Logmeldungen zu generieren. Gerade dieser Aspekt ist bei der Entwicklung einer größeren Anwendung eine hilfreiche Unterstützung.

Dieser letzte Punkt, die leicht zu erlernende DEL und die große Flexibilität in Bezug auf das Format der Eingabedateien führten zusammen mit der Tatsache, dass eine Java-API zur Verfügung steht, zu der Entscheidung, dieses Produkt im Rahmen dieser Arbeit einzusetzen.

4.4 Auswertung der regulären Anteile mit Parsern

Lexer- und Parsergeneratoren

Die regulären Anteile der Eingabedateien wurden — wie in der Konzeption schon beschrieben — mit Hilfe von Parsern analysiert. Ein Parser ist dabei ein Programm, das eine Quellsprache in eine Zielsprache übersetzt. Der allgemeine Übersetzungsprozess besteht aus zwei Teilabschnitten: die Analyse und die Synthese. Während der Analyse wird das Quellprogramm in seine Bestandteile zerlegt und eine Zwischendarstellung erstellt, woraus in der Synthese das gewünschte Zielprogramm konstruiert wird.

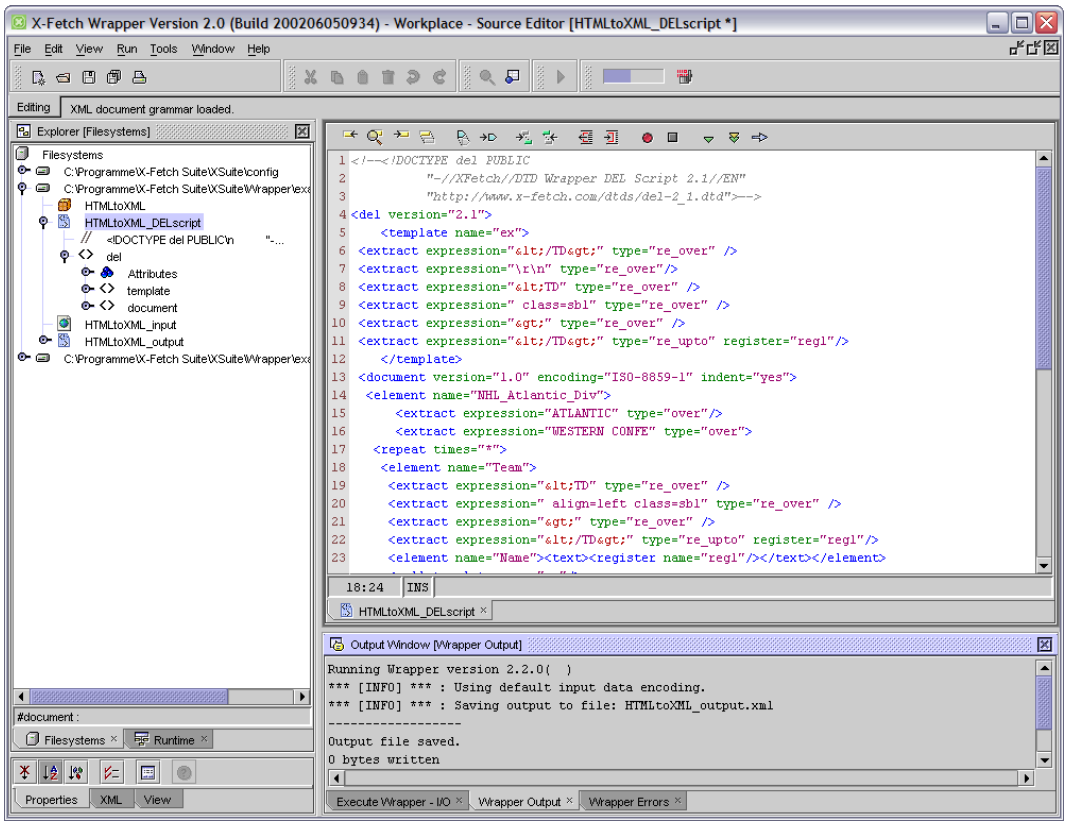


Abbildung 4.7: *GUI des X-Fetch Wrappers*

Die Analyse beinhaltet drei Teilaufgaben: In der lexikalischen Analyse wird der Eingabestrom durch den Lexer in einzelne Symbole zerlegt (siehe Abbildung 4.8). Dieser in so genannte Token zerlegte Strom wird nun in der syntaktischen und semantischen Analyse vom Parser eingelesen, durch eine Grammatik auf syntaktische und semantische Korrektheit geprüft und daraus zuerst Symbolgruppen mit hierarchischer Struktur und dann ein Zwischencode erstellt. [Hof]

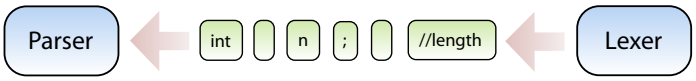


Abbildung 4.8: *Zerlegung des Eingabestroms durch den Lexer*

Das manuelle Erstellen von Lexern und Parser kann bei umfangreichen Grammatiken sehr komplex werden und mit einigen Schwierigkeiten verbunden sein. Aus diesem Grund wurden Werkzeuge entwickelt, die Compiler-Compiler genannt werden und aus einer angegeben Grammatik den Quellcode für einen Lexer und Parser generieren. Die bekanntesten Beispiele dafür sind Lex (ein Lexergenerator) und Yacc (der dazugehörige Parsergenerator). Auch wenn mit „handgeschriebenen“ Übersetzern eine etwa 20% höhere Geschwindigkeit erreicht werden kann, so muss dies durch die schwierige Erstellung des Quelltextes „erkauft“ werden. In dieser Arbeit wurde für die Lexer- und Parsergenerierung das Tool ANTLR eingesetzt, das im Folgenden kurz vorgestellt wird.

ANTLR — Version 2.7.2

Im Gegensatz zu Lex und Yacc vereinigt ANTLR[41], welches 1989 aus PCCTS hervorging und von Terrence Parr entwickelt wurde, die beiden Komponenten Lexer- und Parsergenerator in einem Programm. Für die Erzeugung von Lexer und Parser wird eine Grammatikdatei erstellt, in welcher die zu implementierende Sprache mit Hilfe von EBNF-artigen Regeln spezifiziert wird. Dabei ist auch die Vererbung und Wiederverwendung von Grammatiken erlaubt. Aus dieser Grammatikdatei erzeugt ANTLR den Code — wahlweise in Java, C# oder C++ — für die darin definierten Lexer- und Parserklassen. Abbildung 4.9 zeigt den prinzipiellen Aufbau einer solchen Grammatikdatei.

```
header { /* Code, der an den Anfang aller  generierten Dateien geschrieben werden
     soll */ }

options { /* Optionen für die gesamte Grammatikdatei */ }

{ /* optionaler Klassenvorspann - wird in der generierten Klassendatei der
    Klassendefinition direkt vorangestellt */ }

/* Definition des Lexers */
class MyLexerClass extends Lexer;

options { /* Optionen für den Lexer */ }

tokens { /* explizite Definition von Literalen */ }

/* Regeln für den Lexer */
rulename[args] returns [retval] options { /* Optionen für die Regel */ }
  { /* optionaler Code */ }
  : /* grundlegende Form: */
    Alternative_1
  | Alternative_2
    ...
  | Alternative_n
  ;

/* beliebig viele weitere Regeln */

{ /* optionaler Klassenvorspann - wird in der generierten Klassendatei der
    Klassendefinition direkt vorangestellt */ }

/* Definition des Parsers */
class YourParserClass extends Parser;

options { /* Optionen für den Parser */ }

tokens { /* ... */ }

/* Regeln für den Parser */
rulename[args] returns [retval] options { /* ... */ }
  { /* optionaler Code */ }
  : /* ... */
  ;

/* beliebig viele weitere Regeln */

/* beliebige weitere Lexer, Parser und Treeparser können analog folgen */
```

Abbildung 4.9: *Aufbau einer Grammatikspezifikation*

[41] Abkürzung für *ANother Tool for Language Recognition*. [Par]

Wie in diesem Beispiel zu erkennen ist, ist es dem Nutzer möglich, an jeder beliebigen Stelle der Definitionsdatei eigenen Code hinzuzufügen. Weiterhin gestattet ANTLR neben der Spezifikation von Lexern und Parsern auch die Erstellung von Treeparsern, die im Gegensatz zu den normalen Parsern nicht auf einer Symbolfolge, sondern auf einem Baum als Eingabe arbeiten. Somit kann der Treeparser einen abstrakten Syntaxbaum, den der Parser zuvor optional für erkannte Teile des Eingabestroms konstruiert hat, verarbeiten. Außerdem bietet ANTLR dem Nutzer die Möglichkeit des Einsatzes von syntaktischen und semantischen Prädikaten. Diese stellen Nebenbedingungen dar, die erfüllt sein müssen, damit ein Zweig der Grammatik erkannt wird. Dadurch wird die Grammatikentwicklung zusätzlich vereinfacht.

Im Gegensatz zu Lex/Yacc (LL(1)) ist ANTLR ein LL(k)-Lexer-/Parsergenerator. Dies bedeutet bei der Analyse ein Vorgehen von links nach rechts — also top-down — mit k Eingabesymbolen Vorausschau. Ein größerer Lookahead ist dabei ein wichtiger Vorteil, denn dieser erhöht die Erkennungsfähigkeiten und vereinfacht die Grammatikentwicklung, da intuitivere Regeln möglich sind [PQ96]. Auch wenn ein LL(k)-Parser mit $k > 1$ mehr Speicherplatz und Zeit für die Analyse benötigt, ist dieser Aspekt bei der heutigen rapiden Entwicklung der Computertechnik nahezu irrelevant und kann damit vernachlässigt werden. Aus den genannten Gründen wurde deshalb ANTLR zur Umsetzung der Lexer und Parser in dieser Arbeit eingesetzt.

4.5 Umsetzung des Logbuches mit log4j

Da die Logging-Möglichkeiten über die Standardausgabe `System.out` den aktuellen Ansprüchen bei weitem nicht mehr genügen, wurde für die Integration des Logbuches in dieser Arbeit das Open-Source-Projekt log4j von Apache [Apa] eingesetzt. Mit dieser Schnittstelle sind die definierten Logger zur Laufzeit voll konfigurierbar, da log4j auf externe Konfigurationsdateien aufsetzt.

log4j besteht aus folgenden drei Hauptkomponenten, die es dem Entwickler ermöglichen, Logmeldungen entsprechend ihres Typs und Levels in seinen Quellcode zu integrieren und zu kontrollieren, wie diese Logmeldungen formatiert und wo sie ausgegeben werden:

- dem Logger,
- den Appendern und
- den Layouts.

Die wichtigste Eigenschaft von log4j ist dabei die Realisierung einer Logger-Hierarchie, die es gestattet, Logger zu kategorisieren und ihre Eigenschaften weiterzuvererben. Darüber hinaus wird den einzelnen Lognachrichten ein bestimmter Dringlichkeitslevel — `debug`, `info`, `warn`, `error` oder `fatal` — zugeordnet, anhand dessen die verschiedenen Meldungen klassifiziert und zu Anschauungszwecken gefiltert werden können.

Das Ausgabemedium eines Loggers wird durch einen so genannten Appender bestimmt, wobei ebenfalls die Zuordnung von mehreren Appendern zu ein und demselben Logger erlaubt ist. Mögliche Appender sind dabei beispielsweise der `ConsoleAppender`, der die Standardausgabe `System.out` ersetzt, der `FileAppender`, durch den die Lognachrichten in eine Datei geschrieben werden, oder der `HTMLAppender`, welcher die Meldungen im HTML-Format ausgibt. Zusätzlich können die Logging-Informationen mittels des `SocketAppender` über eine Standard-

Socket-Verbindung versendet werden. Für den Empfang dieser Nachrichten existieren eine Vielzahl graphischer Front-Ends, zum Beispiel Chainsaw V2.

Neben dem Ausgabemedium kann bei einigen Appendern auch das Ausgabeformat beeinflusst werden. Hierzu wird einem Appender ein Layout zugewiesen, welches für die Formatierung der Lognachrichten zuständig ist. So können beispielsweise die Meldungen durch die Angabe eines `PatternLayouts` dahingehend formatiert werden, dass der auslösende Thread, der Level und die Nachricht ansich ausgegeben werden.

In dieser Arbeit wurde die Logger-Hierarchie dazu eingesetzt, Logger für die verschiedenen Phasen des Verarbeitungsprozesses zu definieren. Dadurch ist der Nutzer in der Lage, die Lognachrichten komfortabel zu klassifizieren und den Ablauf der einzelnen Schritte angemessen nachzuvollziehen. Die Ausgabe der Lognachrichten kann dabei über die Konsole, Text- bzw. HTML-Dateien und über eine Socket-Verbindung erfolgen, was in der Konfigurationsdatei `settings.ini` festgelegt wird. Ebenfalls kann über diese Datei definiert werden, welche Dringlichkeitsstufen der Logmeldungen in die Ausgabe mit einbezogen werden sollen.

4.6 DOM-Manipulation mit dom4j

Für die Analyse der in den einzelnen Verarbeitungsschritten entstehenden XML-Dokumente ist zum einen deren Interpretation als Baumstruktur und zum anderen dessen Manipulation notwendig. Daher müssen die XML-Dateien in die entsprechende DOM-Struktur[42] überführt werden, worauf dann XPath-Ausdrücke zur Auswahl der zu bearbeitenden Knoten schnell und effizient ausgewertet werden können.

In dieser Arbeit wurde dafür das Open-Source-Projekt dom4j [Dom] eingesetzt, das momentan die beste Umsetzung für die XPath-Bearbeitung von DOM-Strukturen darstellt. Andere XML-Prozessoren wie zum Beispiel Xerces oder Xerces 2 von Apache beherrschen zwar DOM, können jedoch keine XPath-Anfragen bearbeiten. Dazu müsste eine zusätzliche XPath-API — beispielsweise Apache Xalan — eingebunden werden. Für den Performance-Vergleich der beiden Umsetzungen von dom4j und Apache Xerces/Xalan führte [BD] interessante Untersuchungen durch, nach denen dom4j in nahezu allen Bereichen überlegen war.

Zurzeit beinhaltet dom4j keine Mechanismen zur Validierung von XML-Dokumenten. Man ist also gezwungen, einen externen validierenden XML-Prozessor einzubinden. Für die Implementation der Validierungskomponente dieser Arbeit wurde deshalb Xerces 2 gewählt, welches auch von dom4j vorgeschlagen wird. Durch die modulare Implementierung ist Xerces jedoch jederzeit durch einen anderen XML-Pozessor austauschbar.

4.7 Speicherung der Daten in DB2

Im Rahmen dieser Arbeit wurde für die Speicherung der extrahierten Daten das objektrelationale Datenbanksystem DB2 Universal Database in der Version 8.1 von IBM eingesetzt. Durch den Extender-Mechanismus realisiert DB2 die Integration verschiedener Erweiterungen in das Datenbanksystem, wozu beispielsweise Speicherungstechniken und Anfragemechanismen für Volltexte

[42] Das *Document Object Model* ist eine API und beschreibt, wie man programmiersprachenunabhängig auf HTML oder XML-Dokumente zugreifen kann. [Wik]

gehören. Für die Unterstützung von XML-Dokumenten existiert der XML Extender, der kurz vorgestellt werden soll.

DB2 UDB XML Extender — Version 7.2

Die Unterstützung für verschiedene Speicherungstechniken von XML-Dokumenten wird vom XML Extender, der seit DB2 UDB V7.1 integraler Bestandteil ist, geboten. Zur Speicherung von XML-Daten bietet der XML-Extender zwei Möglichkeiten:

- Der Begriff **XML Collection** umfasst alle Methoden zur Abbildung von XML-Strukturen auf objektrelationale Tabellen und umgekehrt. Dazu gehören die Dekomposition von XML-Dokumenten in Tabellenstrukturen, die Speicherung der Elementinhalte und Attribute als Datenbankattribute sowie die direkte Abbildung von Relationen auf XML-Strukturen. Diese einzelnen Aufgabenbereiche werden durch eine Menge von Stored Procedures realisiert.

- Mit **XML Column** stellt der Extender einen XML-Datentyp zur Verfügung, der zur Speicherung von ganzen XML-Dokumenten oder –Fragmenten eingesetzt wird. Die Struktur und das Verhalten dieses abstrakten Datentyps werden dabei über UDTs[43] (XMLFile, XMLVarchar, XMLClob) bzw. UDFs[44] festgelegt. Dem Datentyp liegt das XPath-Modell zugrunde und bietet im Operationenteil Funktionen und Methoden für den Import, das Retrieval, Update-Operationen und die Extraktion von XML-Bestandteilen.

Einen Überblick über den XML Extender gibt Abbildung 4.10. Die genaue Form der Speicherung wird mit einer Data Access Definition (DAD) beschrieben. Diese Datei, die selbst der XML-Syntax folgt, übernimmt die Steuerung des Abbildungsprozesses von XML-Dokumenten auf Datenbankinhalte und umgekehrt. Dabei kann die Abbildung sowohl auf spezielle Attribute von einem XML-Datentyp (XML Column) als auch auf Attribute, die von einem SQL-Standarddatentyp sind (XML Collection), erfolgen. Es lassen sich dabei zwei Abbildungsmethoden unterscheiden:

- **SQL zur Komposition von XML-Dokumenten**
  Durch SQL-`SELECT`-Anweisungen kann dann die Abbildung von Datenbankattributen auf XML-Daten als Attributwerte mit `attribute_node` und Elementinhalte `element_node` erfolgen.

- **`RDB_node` zur Komposition und Dekomposition**
  Die Spezifikation von Datenbanktabellen und Beziehungen zwischen Tabellen und XML-Elementen wird hierbei durch die Angabe von speziellen `RDB_node`-Knoten realisiert.

Die Integration von XML-Extender-Funktionalitäten in die Anwendungsentwicklung kann wie bei generell allen DB2-Anwendungen über Embedded SQL, ODBC, JDBC und SQLJ erfolgen.

Warum der XML Extender nicht eingesetzt wurde

Trotz dieser komfortablen Möglichkeit, XML-Daten auf DB2-Tabellen abzubilden, wurde der XML Extender in dieser Arbeit nicht eingesetzt. Die anzuwendende Abbildungsmethode wäre dabei die

[43] Abkürzung für *User Defined Types*.
[44] Abkürzung für *User Defined Functiones*.

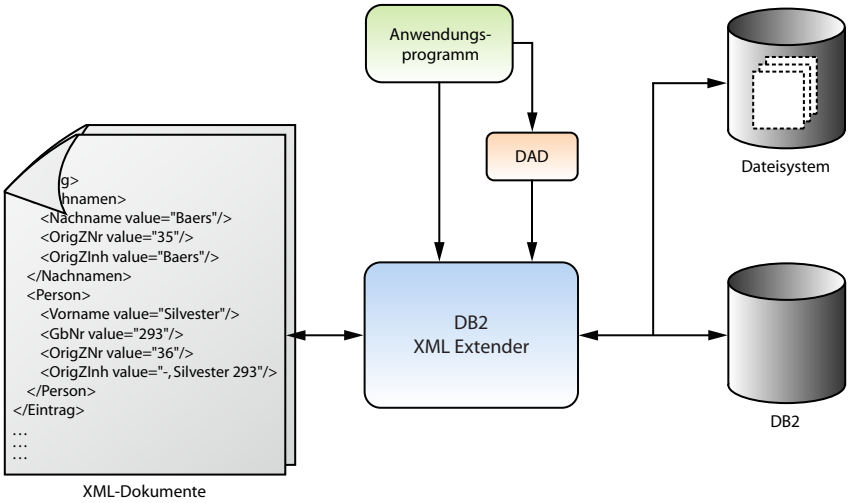


Abbildung 4.10: *Überblick über den XML Extender*

Dekomposition mittels der `RDB_node`-Zuordnung gewesen, bei der die Element- und Attributwerte in Zeilen einer oder mehrerer Tabellen extrahiert werden. Diese Verfahrensweise ist jedoch in der Hinsicht eingeschränkt, dass jede Tabelle maximal 1024 Zeilen aus jedem zerlegten Dokument enthalten kann. Wenn beispielsweise die Daten eines XML-Dokuments in fünf Tabellen zerlegt werden, kann jede dieser fünf Tabellen bis zu 1024 Zeilen für dieses Dokument aufnehmen. Falls eine Tabelle zum Beispiel Zeilen für zwanzig Dokumente beinhaltet, kann sie bis zu 20.480 Tupel umfassen[45].

Bedenkt man, dass bei relativ großen XML-Dokumenten — wie im Fall der Dateien für die Grundbucheinträge — schnell die Grenze der Beschränkung von 1024 Zeilen pro Dokument für eine Tabelle erreicht ist, stellt sich die Dekompositionsmethode über das `RDB_node`-Mapping als nicht adäquat heraus. Aus diesem Grund wurde auf den Einsatz des XML Extenders verzichtet und die Abbildung von XML-Elementen und –Attributen mit Hilfe der Möglichkeiten von dom4j und unter Einsatz von JDBC direkt umgesetzt (siehe Abbildung 4.11). Durch die modulare Implementierung ist dieser Ansatz jedoch jederzeit durch eine Alternative ersetzbar.

4.8 Aufbau der Java-Klassenstruktur

Die Implementierung dieser Arbeit wurde komplett in Java umgesetzt, dessen größter Vorteil die Systemunabhängigkeit darstellt. Dadurch ist der Einsatz einer Java-Applikation unter verschiedenen Betriebssystemen gewährleistet. Ein weiterer wichtiger Aspekt ist die kostenlose Nutzung von Java, sofern es sich bei der entwickelten Anwendung nicht um ein kommerzielles Produkt handelt. Das Java 2 SDK[46] wurde dafür in der Version 1.4.2 genutzt. Im Folgenden werden die verschiedenen Packages mit den wichtigsten Klassen und deren Bedeutung zusammengefasst.

[45] Diese und andere Beschränkungen für den Einsatz des XML Extenders können unter [IBM] nachgelesen werden.
[46] Abkürzung für *Software Development Kit*.

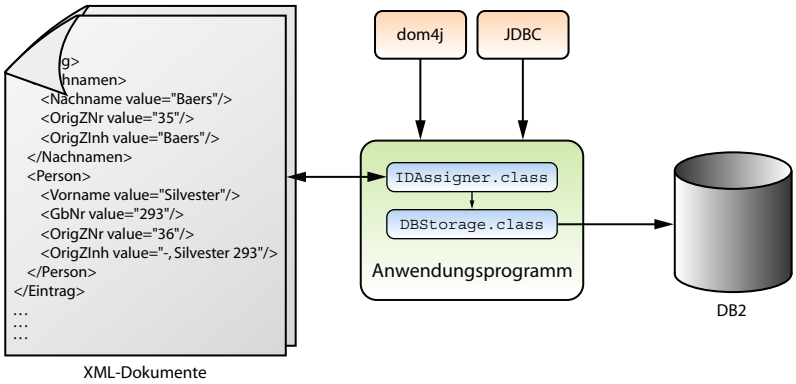


Abbildung 4.11: *Eigener Ansatz zur Speicherung der Daten in DB2*

Package `de.diplom.transformer`

Dieses Package beinhaltet die beiden Klassen, welche die vorbereitende Transformation der Eingabedokumente in das XML-Format umsetzen.

Klasse	Beschreibung
`RTFConverter`	Diese Klasse wandelt eine RTF-Datei in das HTML-Format unter Verwendung des in Kapitel 4.2 beschriebenen Konverters um.
`FileWrapper`	Mittels der Klasse `FileWrapper` wird eine Eingabedatei in das XML-Format umgewandelt. Der Wrapper wird dabei durch ein externes DEL-Skript spezifiziert und durch den X-Fetch-Wrappergenerator erzeugt.

Tabelle 4.1: *Package de.diplom.transformer*

Package `de.diplom.normalizer`

Die hierin enthaltenen Klassen dienen der Umsetzung des Normalisierungsprozesses und der Normalisierung der Rechtschreibung.

Klasse	Beschreibung
`AbbreviationNormalizer`	Innerhalb dieser Klasse wird der Algorithmus zur Ermittlung der Abkürzungvarianten von Seite 34 realisiert.
`SpellingNormalizer`	Diese Klasse implementiert die phonetische Kodierung zur Normalisierung der Rechtschreibung.
`WordNormalizer`	Die Klasse `WordNormalizer` wird dazu eingesetzt, die Ersetzung von einzelnen Worten der Grundbuchdateien anhand der in einer externen Datei definierten Regeln vorzunehmen.

Tabelle 4.2: *Package de.diplom.normalizer*

Package `de.diplom.nester`

Die Klassen dieses Packages realisieren die erste Strukturierung derjenigen flachen XML-Dateien, die als Ergebnis der Layoutanalyse generiert werden.

Package `de.diplom.reganalyzer`

Das Package enthält Klassen zur Analyse der regulären Anteile der Eingabedateien mit Hilfe von Grammatiken. Dazu gehören unter anderem die mittels ANTLR generierten Lexer- und Parserklassen der verschiedenen Dokumenttypen.

Package `de.diplom.irreganalyzer`

Mittels der Java-Klassen in diesem Package werden die Volltextanteile der Grundbuchdateien mit Hilfe der Wörterbücher ausgewertet.

Klasse	Beschreibung
`Dictionary`	Ein `Dictionary`-Objekt stellt die interne Repräsentation eines Wörterbuches auf der Basis einer `Hashtable` dar. Innerhalb dieser Klasse erfolgt ebenfalls die Berechnung des Wortabstandes eines Tokens zu diesem Wörterbuch.
`LevenshteinDistance`	Diese Klasse implementiert die Berechnung des Levenshtein-Abstandes zwischen zwei Strings.
`IrregularContentAnalyzer`	Diese Klasse steuert den Ablauf der Analyse einer Eingabedatei, wie er in Abbildung 3.14 beschrieben wurde.

Tabelle 4.3: *Package de.diplom.irreganalyzer*

Package `de.diplom.semanalyzer`

In diesem Package sind alle Komponenten enthalten, die für die semantische Analyse der zuvor ausgewerteten Volltextinformationen benötigt werden. Dazu gehören neben den definierten semantischen Regeln das zugrunde liegende Interface und die Klasse zur Steuerung des Ablaufs der semantischen Analyse.

Klasse/Interface	Beschreibung
`SemanticRule`	Dieses Interface bildet die Grundlage für die Implementierung von nutzerdefinierten semantischen Regeln, wodurch die uneingeschränkte Erweiterung der semantischen Analyse gewährleistet wird.
`SemanticAnalyzer`	Diese Klasse steuert die Analyse einer Eingabedatei mit den definierten Regeln, wie sie in Abbildung 3.22 schematisch dargestellt wurde.

Tabelle 4.4: *Package de.diplom.semanalyzer*

Package `de.diplom.structurer`

Ähnlich des vorangegangenen Packages beinhaltet dieses das Interface `StructureRule`, auf dessen Basis alle Strukturierungsregeln definiert werden müssen, und die Klasse zur Durchführung der Informationsstrukturierung einer Grundbuchdatei. Alle implementierten Regeln müssen ebenfalls in diesem Package abgelegt werden.

Package `de.diplom.checker`

Dieses Package beinhaltet ebenfalls ein Interface (`CheckRule`), das der Definition der in Kapitel 3.11.3 beschriebenen Regeln zur Überprüfung der Informationsstrukturierung dient. Die Klassen, welche dieses Interface implementieren, müssen sich ebenso in diesem Package befinden.

Package `de.diplom.db`

Innerhalb dieses Packages sind alle Klassen definiert, die für die Speicherung der analysierten Informationen in der Datenbank notwendig sind.

Klasse	Beschreibung
`IDAssigner`	Diese Klasse ordnet den XML-Elementen entsprechende IDs zu, die als Primärschlüsselwerte in den Datenbanktabellen genutzt werden.
`DBAccess`	In dieser Klasse werden grundlegende Funktionen zum Zugriff auf die Datenbank und die Modifikation der Tabelleninhalte bereitgestellt.
`DBStorageGb`	Die Klasse `DBStorageGb` steuert die Abbildung der XML-Elementinhalte und –Attributwerte der analysierten Grundbuchdokumente auf die entsprechenden Relationenschemata. Analog dazu existieren die beiden Klassen `DBStorageAbkVerz` und `DBStoragePersVerz`.

Tabelle 4.5: *Package de.diplom.db*

Package `de.diplom.util`

Dieses Package beinhaltet wichtige nützliche Hilfsklassen, die von mehreren anderen Klassen unterstützend genutzt werden.

Klasse	Beschreibung
`FileRoutines`	Diese Klasse beinhaltet Funktionalitäten für den Umgang mit Dateien. Dazu gehören zum Beispiel die Überprüfung der Verfügbarkeit von Dateien und Verzeichnissen und das dynamische Laden von Java-Klassen, die ein bestimmtes Interface implementieren.
`Logging`	Diese Klasse übernimmt die Initialisierung der Logger und stellt Funktionen für eine adäquate Ausgabe der Lognachrichten bereit.
`ProjectProperties`	In der Klasse `ProjectProperties` werden die nutzerdefinierten Einstellungen in der Konfigurationsdatei `settings.ini` geladen und den anderen Klassen zur Verfügung gestellt.

Klasse	Beschreibung
`XMLBuilder`	Innerhalb dieser Klasse sind Funktionen implementiert, die unter anderem der Generierung von XML-Strukturen aus den internen Repräsentationen der Daten dienen. Dazu gehört beispielsweise die Abbildung des abstrakten Syntaxbaums auf XML-Elemente und –Attribute.
`XMLValidator`	Diese Klasse dient der Validierung von XML-Dokumenten mit Hilfe von DTDs. Auftretende Validierungsfehler werden dabei über Logging-Mechanismen ausgegeben.

Tabelle 4.6: *Package de.diplom.util*

Package `de.diplom.gui`

In diesem Package befinden sich alle Klassen zur Implementierung der graphischen Benutzeroberfläche.

Wie in dieser Auflistung erkennbar ist, wurden die einzelnen Phasen des Verarbeitungsprozesses und wichtige Funktionalitäten streng modular voneinander getrennt. So ist zum Beispiel die Implementierung der Wortabstandsberechnung oder die Realisierung der RTF-Konvertierung einfach auswechselbar, da jeweils nur eine Klasse angepasst werden muss. Weiterhin wurde allen nutzerdefinierbaren Regeln ein Interface zugrunde gelegt, welches die jeweils benötigte Klassenstruktur definiert. Diese Regeln werden dann während der Verarbeitung der Dokumente dynamisch aus dem jeweiligen Package geladen, wodurch eine uneingeschränkte Erweiterung um beliebig viele Regeln ermöglicht wird.

Kapitel 5

Beurteilung der Ergebnisse

Abschließend soll eine Einschätzung der Ergebnisse erfolgen, welche durch die Anwendung des konzipierten Algorithmus auf das Abkürzungsverzeichnis, den Personenindex und den ersten Band der vorliegenden Grundbücher erzielt werden konnten. Dazu werden zum einen Schwierigkeiten und Einschränkungen bei der Auswertung der Dokumente dargelegt und zum anderen Statistiken für die quantitative Beurteilung aufgezeigt.

5.1 Layout- und Strukturanalyse

Wie in Kapitel 3.3.1 beschrieben, dient als grundlegendes Merkmal für die Auswertung der Eingabedokumente deren zeilenweiser Aufbau. Demnach beinhaltet jede Zeile — beendet jeweils mit einem Zeilenumbruch — Daten zu einer abgeschlossenen Informationseinheit.

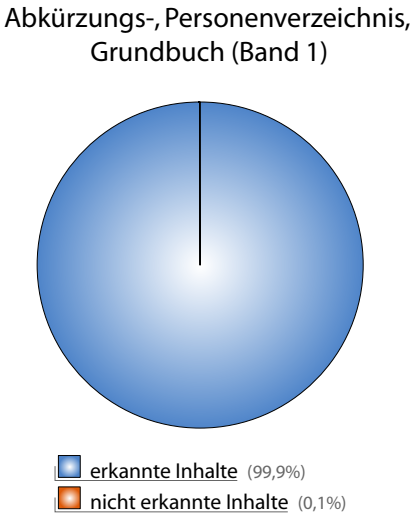
Darauf aufbauend wurde auch der Wrapping-Algorithmus konzipiert (siehe Abbildung 3.6), der Zeile für Zeile separat betrachtet und deren Inhaltstyp identifiziert. Sind nun beispielsweise durch inkorrekte Nutzereingaben Zeilenumbrüche an falschen Stellen in den Quelldokumenten eingefügt, ist der Algorithmus nicht mehr in der Lage, den Inhaltstyp zuverlässig zu bestimmen. Deshalb muss die strukturelle Korrektheit der Quelldokumente sichergestellt sein.

Trifft weiterhin auf eine eingelesene Zeile kein dem Wrapper bekanntes Muster zu, so wird dieser Inhalt nicht erkannt. Eine den Nutzer unterstützende Fehlererkennung wird — wie in Kapitel 3.11.1 angemerkt — dahingehend realisiert, dass entsprechende Fehlermeldungen über Logbuchnachrichten ausgegeben werden. Außerdem werden kaskadierende Fehler vermieden, indem bei Nichterkennen einer Zeile davon inhaltlich abhängige Zeilen ebenfalls als nicht identifiziert markiert werden.

Trotz dieser Einschränkungen konnte der Hauptteil der Inhalte bestimmt werden. Statistiken über den Erfolg des Wrappens der einzelnen Eingabedokumenttypen sind in Abbildung 5.1 dargestellt.

5.2 Auswertung der regulären Anteile

Die in dieser Arbeit erstellten Grammatikregeln zur Interpretation der strukturierten Informationen erkennen einen Großteil der in den Dokumenten enthaltenen Anteile (siehe Statistik in Abbildung 5.2). Dennoch existieren einige Fälle, in denen der Inhalt nicht der definierten Gram-



	erkannt	nicht erkannt
Abkürzungsverzeichnis	100%	0%
Personenverzeichnis	99,83%	0,17%
Grundbuch (Band 1)	99,97%	0,03%

Abbildung 5.1: *Statistik zum Wrapping-Prozess*

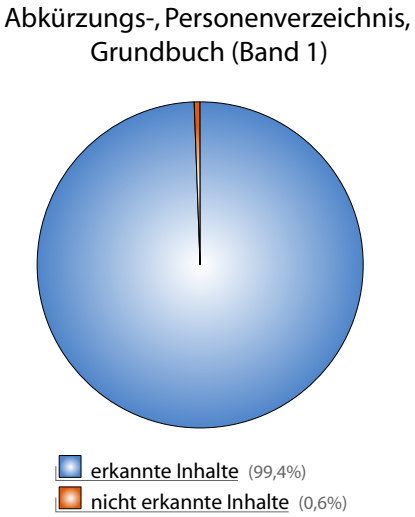
matik genügt und somit nicht durch den Parser analysiert werden kann. Zu diesen Fällen zählen vor allem Tippfehler in den Quelldokumenten, in denen Zeichen, die zur Erkennung des Informationstyps in der Phase des Wrappens genutzt werden, versehentlich vertauscht, eingefügt oder vergessen wurden. So werden beispielsweise in den Personenverzeichnissen Personeneinträge dadurch eindeutig identifiziert, dass sie mit „-“ beginnen. Wurde nun im Quelldokument irrtümlich ein Nachnameneintrag mit diesem Zeichen begonnen, genügt der als Person erkannte Inhalt in der nächsten Phase nicht mehr der entsprechenden Grammatikregel.

Des Weiteren gibt es einige wenige Spezialfälle, die von den definierten Regeln nicht abgedeckt werden, da sie nur einen geringen Bruchteil des Gesamtinhaltes ausmachen. Die Integration dieser Besonderheiten in die Grammatiken ist realisierbar, wäre jedoch mit einem hohen Aufwand verbunden, da die Grammatiken dadurch sehr komplex würden und der Lookahead entsprechend erweitert werden müsste, um Nichtdeterminismen zu vermeiden. Einen solchen Sonderfall stellen beispielsweise Namen in den Personenverzeichnissen dar, bei denen verschiedenen Schreibweisen durch Klammern gekennzeichnet wurden (zum Beispiel „H(e)inrich“).

5.3 Analyse der Volltextanteile

Die Ergebnisse der Volltextanalyse mit Hilfe der Wörterbücher sind stark abhängig von folgenden Einflussfaktoren:

- Vollständigkeit der Wörterbücher,
- Einsatz der Rechtschreibnormalisierung und Relevanz der Unterscheidung zwischen Groß- und Kleinschreibung beim Vergleich der Token mit den Wörterbuchbegriffen,
- Wahl des Schwellenwertes für den Wortabstand und dessen Wichtung,
- Wahl der Selektionsmethode für jene Wörterbücher, deren berechneter Abstand unter diesem Grenzwert liegt.



	erkannt	nicht erkannt
Abkürzungsverzeichnis	100%	0%
Personenverzeichnis	99,33%	0,67%
Grundbuch (Band 1)	100%	0%

Abbildung 5.2: *Statistik zur Auswertung durch Grammatiken*

Für die Anwendung des wörterbuchbasierten Verfahrens auf die Grundbuchdokumente des ersten Bandes wurden folgende Festlegungen getroffen:

- **Schwellenwert:** 0.2,
- **Wichtung des Schwellenwertes:** ja,
- **Selektionsmethode:** Wörterbuch mit dem minimalsten Wortabstand,
- **Normalisierung der Rechtschreibung:** für das Wörterbuch `Vorname`,
- **Unterscheidung zwischen Groß- und Kleinschreibung:** für Stoppworte und für die Wörterbücher `Beruf`, `Feiertag`, `Grundbuch`, `Nachname`, `Nummer`, `Personenbeziehung`, `röm. Zahl` und `Vorname`.

Die Ergebnisse der Volltextauswertung sind in den beiden Abbildungen 5.3 und 5.4 zu sehen. Zusätzlich zeigt Abbildung 5.5, welchem Anteil der Grundbucheinträge Begriffe aus den entsprechenden Wörterbüchern zugeordnet werden konnte. Wie in der ersten Statistik erkennbar ist, konnten nahezu der Hälfte der Token keine Bedeutung zugeordnet werden. Der Grund hierfür liegt darin, dass im zeitlichen Rahmen dieser Arbeit keine vollständige Aufstellung aller notwendigen Wörterbuchbegriffe realisierbar ist. Eine Erweiterung der Wörterbücher, die externe TXT-Dateien darstellen, ist jedoch jederzeit einfach durchführbar.

5.4 Semantische Analyse

Der Zweck der semantischen Analyse besteht darin, denjenigen Token, die in der Volltextanalyse mit Hilfe der Wörterbücher nicht exakt bestimmt werden konnten, eine eindeutige Kategorie zuzuordnen. Die Auswirkung der dazu in dieser Arbeit beispielhaft definierten Regeln (siehe Anhang

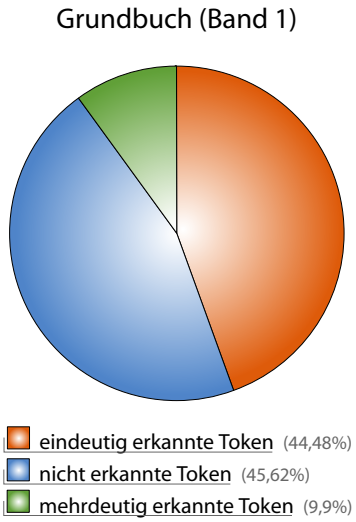


Abbildung 5.3: *Statistik zur Volltextauswertung*

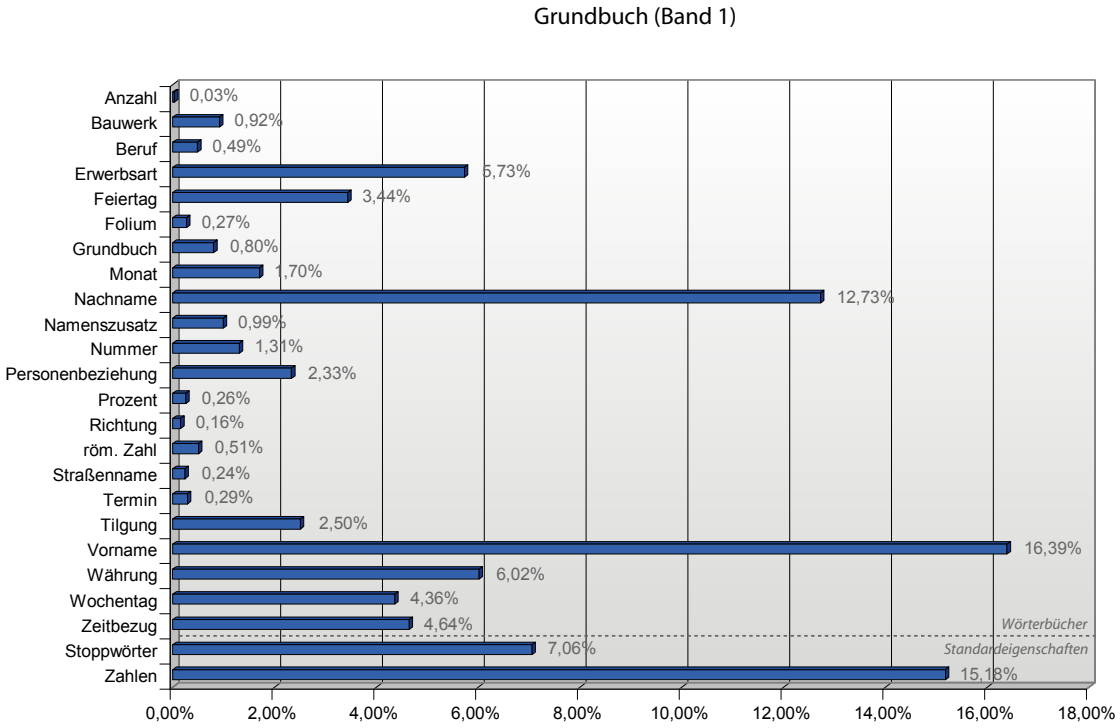


Abbildung 5.4: *Verteilung der Bedeutungen nach der Volltextanalyse*

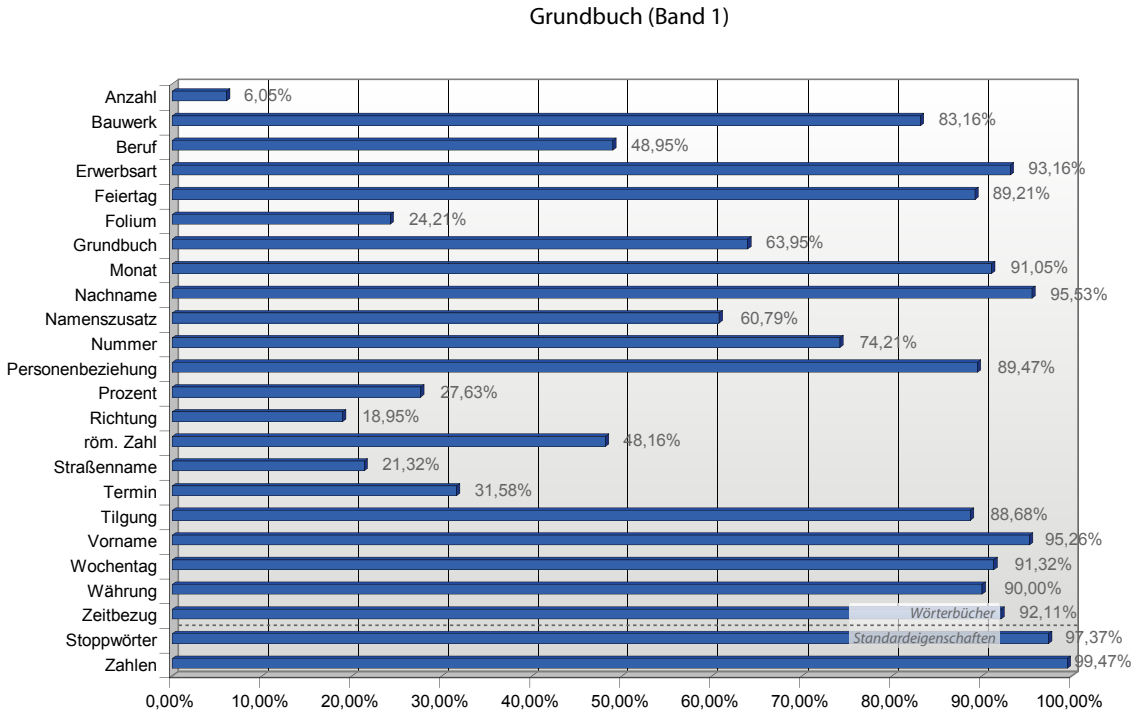


Abbildung 5.5: *Anteil der Grundbucheinträge mit zuordbaren Wörterbüchern*

A) auf die Ergebnisdokumente aus dem vorangegangenen Abschnitt ist in Abbildung 5.6 und 5.7 veranschaulicht.

Obwohl die in dieser Arbeit definierten Regeln keine vollständige Auflistung darstellen[47], lässt sich anhand der Statistiken bereits das Potential der semantischen Analyse erkennen: Wie in Statistik 5.6 im Vergleich zu Abbildung 5.3 zu sehen ist, konnte der Anteil der nicht identifizierten Token enorm minimiert werden. Dies ist vor allem darauf zurückzuführen, dass den vorher noch allgemeinen Zahlen mit Hilfe der Regeln eine Bedeutung gegeben wurde. Diese Tatsache lässt sich ebenfalls anhand der Statistik in Darstellung 5.8 erkennen. Weiterhin konnten Mehrdeutigkeiten von Token aufgrund der Regeln eingeschränkt werden.

5.5 Informationsstrukturierung

Das Resultat der Anwendung der in Anhang B dargestellten Strukturierungsregeln auf die Ergebnisdokumente der semantischen Analyse zeigt Abbildung 5.9. Demnach ist nach der Informationsstrukturierung nahezu die Hälfte aller identifizierten Daten logisch gruppiert.

Vorwiegend kamen dabei die Regeln zur Gruppierung von Personendaten und Zeitangaben zum Einsatz, wie in Abbildung 5.10 zu erkennen ist. Des Weiteren sind die Regeln zur Zuordnung von Informationen zu anderen, wie beispielsweise die Währung einem Geldbetrag, und der Zusammenfügung von mehrteiligen Begriffen von besonderer Bedeutung.

[47] Die Gründe hierfür wurden in Kapitel 3.8.3 auf Seite 61 angeführt.

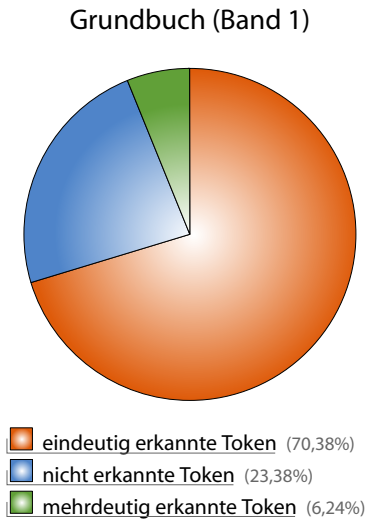


Abbildung 5.6: *Statistik zur semantischen Analyse*

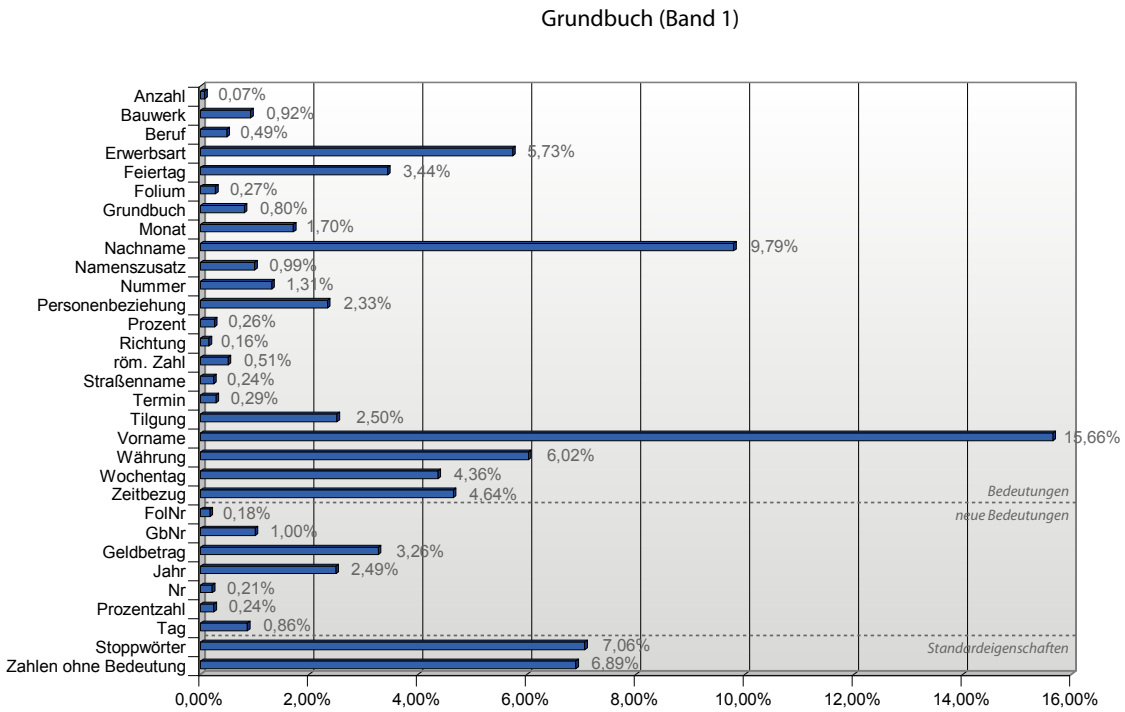


Abbildung 5.7: *Verteilung der Bedeutungen nach der semantischen Analyse*

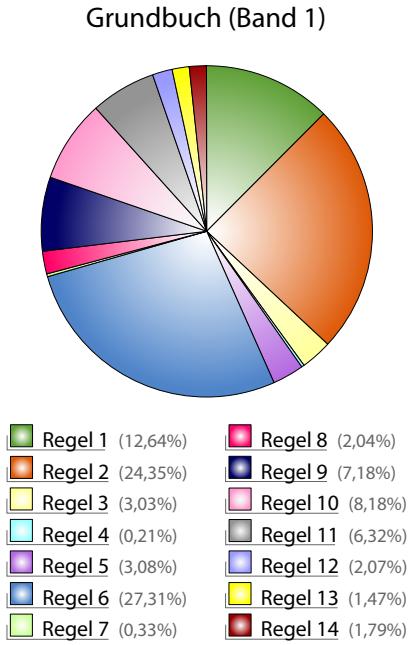


Abbildung 5.8: *Häufigkeit der Anwendung der semantischen Regeln*

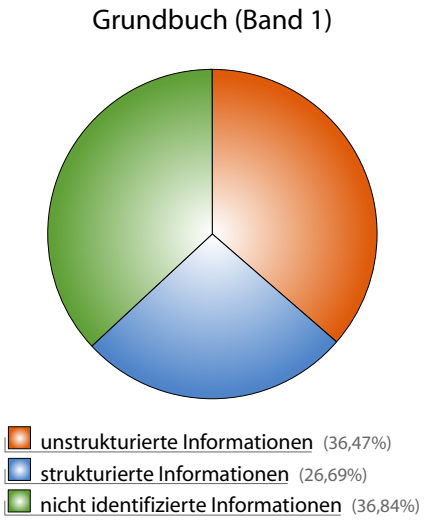


Abbildung 5.9: *Statistik zur Informationsstrukturierung*

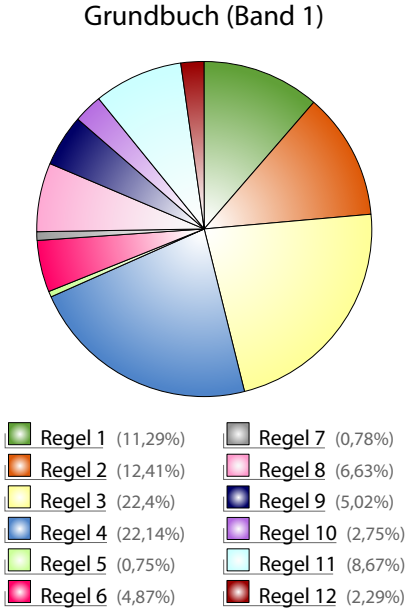


Abbildung 5.10: *Häufigkeit der Anwendung der Strukturierungsregeln*

Kapitel 6

Zusammenfassung & Ausblick

Zusammenfassend lässt sich feststellen, dass die Aufgabenstellung durch die Konzeption und Umsetzung eines Algorithmus zur Extraktion von Informationen sowohl aus den regulären als auch aus den Volltextanteilen der Eingabedateien erfüllt werden konnte. Dazu wurden zunächst für die Interpretation der Inhalte relevante Strukturen und Informationstypen der Ausgangsdokumente ermittelt. Darauf aufbauend wurde ein mehrstufiger Transformationsprozess konzipiert, der schrittweise die Dokumentinhalte klassifiziert und strukturiert. Dafür erfolgt zu Beginn die Umwandlung der Eingabedateien in besser auswertbare Formate, auf deren Grundlage dann eine Abbildung der Layout- und Strukturinformationen auf Bedeutungen der Inhalte durchgeführt wird. Anschließend findet eine Normalisierung der Dokumente statt, welche unter anderem die Ersetzung von verschiedenen Schreibweisen und Flexionsformen von Wörtern durch eine Grundform anhand von nutzerdefinierbaren Regeln beinhaltet. Die regulären Dokumentinhalte werden danach durch Grammatikregeln und den Einsatz von Lexern und Parsern ausgewertet, woran sich die Analyse der Volltextanteile auf der Grundlage von Wörterbüchern anschließt. Dabei werden buchstaben- und sprachorientierte Verfahren zur Wortabstandsberechnung und Normalisierung der Rechtschreibung angewendet. In der semantischen Analyse erfolgt die Identifikation derjenigen Worte, welche mit Hilfe der Wörterbücher mehrdeutig oder nicht erkannt wurden. Die so gewonnenen Informationen werden im darauf folgenden Verarbeitungsschritt entsprechend ihrer logischen Zusammengehörigkeit strukturiert, woran sich die Abbildung der identifizierten Inhalte auf relationale Datenbanktabellen anschließt.

Als Unterstützung für den Nutzer wurde zusätzlich ein Prüfmechanismus in den Verarbeitungsprozess integriert, der die Erkennung von Fehlern und Inkonsistenzen ermöglicht. Durch eine konsequente Einbeziehung von Logmeldungen in den Transformationsablauf und die Validierung der in den einzelnen Schritten generierten Strukturen ist der Anwender in der Lage, eventuell auftretende Fehler zu lokalisieren und wenn nötig manuelle Nachbearbeitungen vorzunehmen.

Bei der Implementierung des Algorithmus wurde insbesondere auf die modulare Umsetzung der einzelnen Verarbeitungsschritte Wert gelegt. So sind etwa alle nutzerdefinierbaren Komponenten — wie beispielsweise die Wörterbücher in Form von externen Dateien oder die semantischen Regeln in Form von Java-Klassen — streng von der internen Realisierung getrennt und dadurch komfortabel anpassbar und beliebig erweiterbar. Zudem wird dem Nutzer durch die Festlegung von bestimmten Parametern die Möglichkeit gegeben, auf den Verarbeitungsprozess Einfluss zu nehmen.

Damit der Algorithmus seine volle Leistungsfähigkeit erreicht, müssen im Anschluss an diese Arbeit die Wörterbuchbegriffe vervollständigt und die unterstützenden Regeln für die Ersetzung einzelner Worte erweitert werden. Außerdem ist die Anpassung respektive Ergänzung der semantischen Regeln empfehlenswert. Sind diese abschließenden Erweiterungen umgesetzt, wird durch den Verarbeitungsprozess eine Vielzahl der in den Dokumenten enthaltenen Informationen erkannt werden.

Eine nützliche Erweiterung des Wörterbuchverfahrens kann die Integration von so genannten Wildcards in die Definition der Begriffe darstellen. Dadurch könnte beispielsweise in das Wortverzeichnis für die Straßennamen der Begriff „*straße“ aufgenommen werden und die Angabe einer Vielzahl von Straßenbezeichnungen damit entfallen. Dies würde einerseits eine Erleichterung für den Anwender und andererseits einen Performance-Gewinn bei der Dokumentverarbeitung darstellen, da die Geschwindigkeit der Inhaltsanalyse stark abhängig vom Umfang der Wörterbücher ist. Weiterhin wäre die eingeschränkte Anwendung der Wortverzeichnisse auf die Eingabedateien denkbar. Dabei würde dem Nutzer die Möglichkeit gegeben, die Anwendung einzelner Wörterbücher auf bestimmte Inhalte auszuschließen, wodurch die Kontrolle über die analysierten Informationen realisierbar wäre. Für die Untersuchung der einzelnen Token der Volltextanteile in Bezug auf die Standardeigenschaften „Stoppwort“ und „Zahl“ wäre ebenfalls die Ergänzung um weitere Merkmale vorstellbar, wie beispielsweise die Zugehörigkeit einer Zahl zu einem bestimmten Wertebereich.

Für eine dem unerfahrenen Nutzer angemessene Applikation zur Steuerung des Transformationsprozesses und der Fehlersuche ist eine Erweiterung der graphischen Oberfläche angebracht. Denkbar wäre hier zum Bespiel eine Komponente, welche die Verarbeitungsfehler, die momentan als Logbuchnachrichten ausgegeben werden, entsprechend aufbereitet, die verursachenden Stellen im Eingabedokument markiert und dem Anwender in einem separaten Fenster präsentiert. Dadurch wäre der Nutzer in der Lage, die Fehlerquellen schnell und einfach zu lokalisieren. Ebenso wäre eine Ergänzung der Statistiken über die Ergebnisse der Volltextanalyse in der Hinsicht nützlich, dass zusätzliche Informationen über die Mehrdeutigkeit von Begriffen bereitgestellt werden. So könnten beispielsweise diejenigen Wortkategorien identifiziert werden, die sich häufig überschneiden, aufgrund dessen dann eine Anpassung der Wörterbuchbegriffe oder semantischen Regeln möglich wäre.

Wie in Kapitel 5 beschrieben, konnten mit Hilfe des umgesetzten Algorithmus bereits eine Vielzahl an Daten identifiziert werden. Diese extrahierten Informationen, die nach Abschluss des Verarbeitungsprozesses in Form einer Datenbank vorliegen, stellen eine wichtige Quelle für darauf aufbauende Applikationen dar. Diese können aufgrund der Daten unterschiedlichste Visualisierungen über Wismars historische Aspekte realisieren. Durch die Speicherung der Originaldokumente in der Datenbank und die Bezüge der extrahierten Informationen zur Quelle ist weiterhin die Möglichkeit gegeben, den Ursprung der Informationen jederzeit nachzuvollziehen. Die zusätzliche Auswertung des Abkürzungsverzeichnisses gestattet es späteren Anwendungen zudem, automatisch Erklärungen zu verwendeteten Kurzworten bereitzustellen. Die Erweiterung des Konzepts um die angegebenen Funktionalitäten würde das in dieser Arbeit realisierte Tool weiter abrunden, welches bereits aufgrund der nutzerdefinierbaren Einstellungen, beliebig erweiterbaren Regeln und modularen Anpassung einzelner Komponenten ein äußerst flexibles Werkzeug darstellt.

Literaturverzeichnis

[Apa] Apache Software Foundation
„log4j“
`http://logging.apache.org/log4j/docs/` (am 28. September 2004)

[BD] Martin Böhm, Jean-Jacques Dubray
„Dom4J performance versus Xerces/Xalan“
`http://www.dom4j.org/benchmarks/xpath/` (am 28. September 2004)

[Bog] Alexander Bogomolny
„Distance Between Strings“
`http://www.cut-the-knot.org/do_you_know/Strings.shtml`
(am 2. September 2004)

[CCG] Centre for Computational Geography (CCG), University of Leeds
„SPIN! — Spatial Mining for Data of Public Interest“
`http://www.ccg.leeds.ac.uk/spin/` (am 25. Juni 2004)

[Com] Community of Knowledge
„Knowledge Discovery in Databases, Teil II — Web Mining“
`http://www.community-of-knowledge.de/cp_artikel.htm?artikel_id=145`
(am 25. Juni 2004)

[Dom] *„dom4j“*
`http://www.dom4j.org/` (am 28. September 2004)

[Dud01] *„Der Duden Band 5 — Das Fremdwörterbuch“*
Bibliographisches Institut, Mannheim
2001

[Fri96] Marc Friedrich
„Entwurf und Implementierung von Verfahren zur Bestimmung von Textinhalten aus Verkehrsmeldungen für die Überführung in das digitale RDS-TMC-Format“
Diplomarbeit, Technische Universität Darmstadt
`http://www.smile-datentechnik.de/projekte/dipl/index.html`
(am 2. September 2004)
1996

[FU] FSF/UNESCO
„Free Software Directory“
`http://www.gnu.org/directory/webauth/htmlconvert/` (am 26. Juni 04)

[GG] Institut für Geodäsie und Geoinformatik (GG), Universität Rostock
„Geoinformatik-Lexikon“
`http://http://www.geoinformatik.uni-rostock.de/lexikon.asp`
(am 25. Juni 2004)

[Gil] Michael Gilleland
„Levenshtein Distance, in Three Flavors“
`http://www.merriampark.com/ld.htm` (am 2. September 2004)

[Goh00] Rolf Gohla
„Integrierte WWW-Anfragesichten“
Diplomarbeit, Universität Rostock, Fachbereich Informatik
2000

[HG03] Anke Holler, Stefan Geißler
„Wenn aus Information Wissen wird — Eine kurze Einführung in die Welt des Text Mining“
`http://www.cl.uni-heidelberg.de/kurs/ss03/textmengen/einfuehrungTM.pdf`
(am 25. Juni 2004)
2003

[HK01] Jiawei Han, Micheline Kamber
„Data Mining — Concepts and Techniques“
Morgan Kaufmann Publishers, San Francisco et al.
2001

[Hof] Christian Hoffmann
„Der Zusammenhang zwischen Lexer, Parser und TreeParser“
`http://www.c-hoffmann.de/WinPL/antlr/` (am 28. September 2004)

[IBM] IBM
„XML Extender — Administration and Programming“
`http://www-306.ibm.com/software/data/db2/extenders/xmlext/downloads.html`
(am 6. Oktober 2004)

[Kle04] Manfred Klenner
„Computerlinguistik-Techniken für Text Mining und Information Extraction“
`http://www.ifi.unizh.ch/ddis/fileadmin/teaching/current/BI/klenner/Vortrag_Klenner.pdf` (am 25. Juni 2004)

[KM03] Meike Klettke, Holger Meyer
„XML & Datenbanken — Konzepte, Sprachen und Systeme“
dpunkt.verlag, Heidelberg
2003

[KT02] Stefan Kuhlins, Ross Tredwell
„Toolkits for Generating Wrappers — A Survey of Software Toolkits for Automated Data Extraction from Websites“
Lecture Notes In Computer Science, Revised Papers from the International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World, S. 184 – 198
Springer-Verlag, London
2002

[Lan02] Martin Lang
„Heterogene Informationssysteme — Ein Überblick“
`http://www.m-lang.de/data/mediator/hiscore.PDF` (am 5. September 2004)
2002

[Log03] Logictran
„R2Net: RTF to HTML and XML converter“
`http://www.logictran.net/products/r2net.html` (am 5. September 2004)
2003

[LRS+02] Alberto H. F. Leander, Berthier A. Ribeiro-Neto, Altigran S. da Silva, Juliana S. Teixeira
„A Brief Survey of Web Data Extraction Tools“
ACM SIGMOD Record, Volume 31, Issue 2 (Juni 2002), S. 84 – 93
ACM Press, New York
2002

[Mar] Hans-Jürgen Martin
„Geschichtlicher Abriß der Rechtschreibung“
`http://www.schriftdeutsch.de/orth-his.htm` (am 2. August 2004)

[Mic99] Microsoft Corporation
„Rich Text Format (RTF) Specification, version 1.6“
`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp` (am 29. August 2004)
1999

[Mue01] Stefan Münz
„SELFHTML — Version 8.0“
`http://de.selfhtml.org` (am 30. August 2004)
2001

[Mue02] Ernst Münch (Hrsg.)
„Das Wismarer Grundbuch (1677/80 – 1838)“
Verlag Schmidt-Römhild, Rostock 2002

[Par] Terence Parr
„ANTLR — ANother Tool for Language Recognition“
`http://www.antlr.org` (am 28. September 2004)

[PIP] PIPER
„PIPER Toolkit“
`http://piper.ntua.gr/toolkit.html` (am 5. September 2004)

[PQ96] Terence Parr, Russell Quong
„LL and LR Translator Need k“
ACM SIGPLAN Notices, Volume 31, Issue 2 (Februar 1996), S. 27 – 34
ACM Press, New York
1996

[Rep] Republica Corporation
„X-Fetch Suite“
`http://www.x-fetch.com` (am 1. Juli 2004)

[SA98] Arnaud Sahuguet, Fabien Azavant
„W4F: a WysiWyg Web Wrapper Factory for Minute-Made Wrappers“
`http://db.cis.upenn.edu/DL/wapi.pdf` (am 5. September 2004)
1998

[SA99] Arnaud Sahuguet, Fabien Azavant
„Web Ecology — Recycling HTML pages as XML documents using W4F“
ACM International Workshop on the Web and Databases (WebDB'99)
`http://db.cis.upenn.edu/DL/webdb99.pdf` (am 5. September 2004)
1999

[SA01] Arnaud Sahuguet, Fabien Azavant
„Building Intelligent Web Applications Using Lightweight Wrappers“
Data & Knowledge Engineering archive, Volume 36, Issue 3 (März 2001), S. 283 – 316
Elsevier Science Publishers B. V., Amsterdam
2001

[Sch02] Dr. Lars Schmidt-Thieme
„KDD, Data Mining und Web Mining“
`http://www.informatik.uni-freiburg.de/cgnm/lehre/wm-02w/webmining-1.pdf`
(am 27. August 2004)
2002

[Sof04] Softinterface, Inc.
„Convert Doc“
`http://www.softinterface.com/Convert-Doc/Convert-Doc.htm`
(am 5. September 2004)
2004

[Ver02] Fonny Cendekia Vera
„Text Mining : Grundlagen, Verfahren, und Anwendungen“
`http://www-i5.informatik.rwth-aachen.de/lehrstuhl/lehre/WebIntelligenz/Beitraege/Text%20Mining%20Ausarbeitung.pdf` (am 25. Juni 2004)
2002

[W3C] W3C — World Wide Web Consortium
`http://www.w3.de` (am 30. August 2004)

[Wik] Wikipedia — Die freie Enzyklopädie
`http://de.wikipedia.org/wiki/Hauptseite` (am 29. August 2004)

[XML02] XML und Publizieren
„Glossar ‚XML und Publizieren‘“
`http://www.xml-newsletter.de/glossar.html` (am 5. September 2004)
2002

[Zie97] Marco Zierl
„Entwicklung und Implementierung eines Datenbanksystems zur Speicherung und Verarbeitung von Textkorpora“
`http://www.linguistik.uni-erlangen.de/tree/html/corsica/zierl97/zierl97.html` (am 27. August 2004)
1997

Abbildungsverzeichnis

1.1 Taxonomie des Web Mining 11
1.2 Verallgemeinerte Architektur eines IE-Systems 12

2.1 Aufbau des Abkürzungsverzeichnisses 14
2.2 Aufbau des Personenverzeichnisses 16
2.3 Aufbau der Grundbucheinträge 17
2.4 Evaluierung der Informationstypen im Volltextanteil — Teil 1 19
2.5 Evaluierung der Informationstypen im Volltextanteil — Teil 2 20
2.6 Evaluierung der Informationstypen im Volltextanteil — Teil 3 21

3.1 Überblick über die Architektur 24
3.2 Beispiel eines RTF-Dokuments im Klartext 26
3.3 Merkmale des Abkürzungsverzeichnisses 29
3.4 Merkmale der Personenindexe 30
3.5 Merkmale der Grundbücher 30
3.6 Genereller Ablauf der Layoutanalyse 32
3.7 Ablauf der Normalisierung 34
3.8 Ermittlung aller Abkürzungsvarianten 35
3.9 Algorithmus der Wortnormalisierung 38
3.10 Priorisierung der Ersetzungsregeln 39
3.11 Kaskadierende Ersetzungen 40
3.12 Transformation und Einpassung des abstrakten Syntaxbaums 42
3.13 Beispiel für die Auswertung der regulären Anteile 43
3.14 Genereller Ablauf der Volltextauswertung 44
3.15 Funktionen des Punktes innerhalb des Textes 45
3.16 Identifikation eines Tokens mit Hilfe eines Wörterbuches 48
3.17 Möglichkeiten für die Interpretation der Wörterbucheinträge 49
3.18 Mehrdeutigkeiten bei der Tokenidentifikation 49
3.19 Mehrdeutige Identifizierung von zusammengesetzten Begriffen 50
3.20 Beispiel für die Berechnung der Levenshtein-Distanz 52
3.21 Tokenliste für die Regelanwendung 59
3.22 Ablauf der semantischen Analyse 59
3.23 Wiederholung der Regelanwendung 60
3.24 Gruppierung von Informationen 64
3.25 ER-Modell für das Abkürzungsverzeichnis 65

3.26 ER-Modell für das Personenverzeichnis 66
3.27 ER-Modell für das Grundbuch 67
3.28 Kandidaten für die IST-Beziehung 68
3.29 Konkretes Beispiel für die Abbildung von XML-Daten auf Entitäten 69
3.30 Beispiel einer Statistik aus dem Logbuch 70

4.1 RTF-HTML-Konvertierung mit Word 73
4.2 Word-HTML mit Dreamweaver optimiert 74
4.3 Konvertierungsergebnis des eingesetzten RTF-Konverters 76
4.4 Phasen eines Wrappers 78
4.5 Architektur des W4F 80
4.6 Architektur des X-Fetch Wrappers 80
4.7 GUI des X-Fetch Wrappers 82
4.8 Zerlegung des Eingabestroms durch den Lexer 82
4.9 Aufbau einer Grammatikspezifikation 83
4.10 Überblick über den XML Extender 87
4.11 Eigener Ansatz zur Speicherung der Daten in DB2 88

5.1 Statistik zum Wrapping-Prozess 93
5.2 Statistik zur Auswertung durch Grammatiken 94
5.3 Statistik zur Volltextauswertung 95
5.4 Verteilung der Bedeutungen nach der Volltextanalyse 95
5.5 Anteil der Grundbucheinträge mit zuordbaren Wörterbüchern 96
5.6 Statistik zur semantischen Analyse 97
5.7 Verteilung der Bedeutungen nach der semantischen Analyse 97
5.8 Häufigkeit der Anwendung der semantischen Regeln 98
5.9 Statistik zur Informationsstrukturierung 98
5.10 Häufigkeit der Anwendung der Strukturierungsregeln 99

A.1 Veranschaulichung der ersten semantischen Regel 110

E.1 Auszug aus einem Grundbucheintrag 118
E.2 XML-Dokument nach der Normalisierung 119
E.3 XML-Dokument nach der Auswertung der Volltextanteile 119
E.4 XML-Dokument nach der semantischen Analyse 122
E.5 XML-Dokument nach der Informationsstrukturierung 122

Tabellenverzeichnis

3.0 Ersetzungstabelle des Soundex-Verfahrens 54
3.1 Phonetische Kodierung für Buchstabenkombinationen 55
3.2 Phonetische Kodierung für einzelne Buchstaben 55
3.3 Angepasste Kodierung von Buchstabenkombinationen 56
3.4 Angepasste Kodierung von einzelnen Buchstaben 56

4.0 Package de.diplom.transformer 88
4.1 Package de.diplom.normalizer 88
4.2 Package de.diplom.irreganalyzer 89
4.3 Package de.diplom.semanalyzer 89
4.4 Package de.diplom.db 90
4.5 Package de.diplom.util 90

A.0 Semantische Regeln 109

B.0 Strukturierungsregeln 111

C.0 Regeln zur Überprüfung der Informationsstrukturen 113

D.0 Relationenschemata für das Abkürzungsverzeichnis 114
D.1 Relationenschemata für das Personenverzeichnis 115
D.2 Relationenschemata für das Grundbuch 115

Anhang A

Semantische Regeln

Diese Tabelle gibt einen Überblick über die in dieser Arbeit beispielhaft definierten Regeln für die semantische Analyse, welche in Kapitel 3.8 beschrieben wurde. Für eine Verdeutlichung des Konzepts stellt Abbildung A.1 die Anwendung der ersten angeführten Regel graphisch dar.

Priorität	Regel
1	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Feiertag`. **Aktion:** Ordne dem Token die Bedeutung `Jahr` zu.
2	**Bedingung:** Das Token hat ausschließlich die Bedeutungen `Vorname` und `Nachname` `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Vorname` oder `Nachname`. **Aktion:** Ordne dem Token die Bedeutung `Vorname` zu.
3	**Bedingung:** Das Token hat ausschließlich die Bedeutungen `Vorname` und `Feiertag` `UND` das Nachfolgertoken hat keine Bedeutung `UND` das Nachfolgertoken ist eine Zahl. **Aktion:** Ordne dem Token die Bedeutung `Feiertag` zu.
4	**Bedingung:** Das Token hat ausschließlich die Bedeutungen `Vorname` und `Nachname` `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Nachname`. **Aktion:** Ordne dem Token die Bedeutung `Vorname` zu.
5	**Bedingung:** Das Token hat ausschließlich die Bedeutungen `Vorname` und `Nachname` `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Vorname` `UND` das Nachfolgertoken hat nicht die Bedeutung `Vorname`. **Aktion:** Ordne dem Token die Bedeutung `Nachname` zu.
6	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Währung`. **Aktion:** Ordne dem Token die Bedeutung `Geldbetrag` zu.
7	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Bauwerk`. **Aktion:** Ordne dem Token die Bedeutung `Anzahl` zu.
8	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Prozent`.

Priorität	Regel
	Aktion: Ordne dem Token die Bedeutung `Prozentzahl` zu.
9	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Nachfolgertoken hat ausschließlich die Bedeutung `Monat`. **Aktion:** Ordne dem Token die Bedeutung `Tag` zu.
10	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Monat`. **Aktion:** Ordne dem Token die Bedeutung `Jahr` zu.
11	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Nummer` `UND` das Vorvorgängertoken hat ausschließlich die Bedeutung `Grundbuch`. **Aktion:** Ordne dem Token die Bedeutung `GbNr` zu.
12	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Vorgängertoken hat ausschließlich die Bedeutung `GbNr`. **Aktion:** Ordne dem Token die Bedeutung `GbNr` zu.
13	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Folium`. **Aktion:** Ordne dem Token die Bedeutung `FolNr` zu.
14	**Bedingung:** Das Token hat keine Bedeutung `UND` das Token ist eine Zahl `UND` das Vorgängertoken hat ausschließlich die Bedeutung `Nummer`. **Aktion:** Ordne dem Token die Bedeutung `Nr` zu.

Tabelle A.1: *Semantische Regeln*

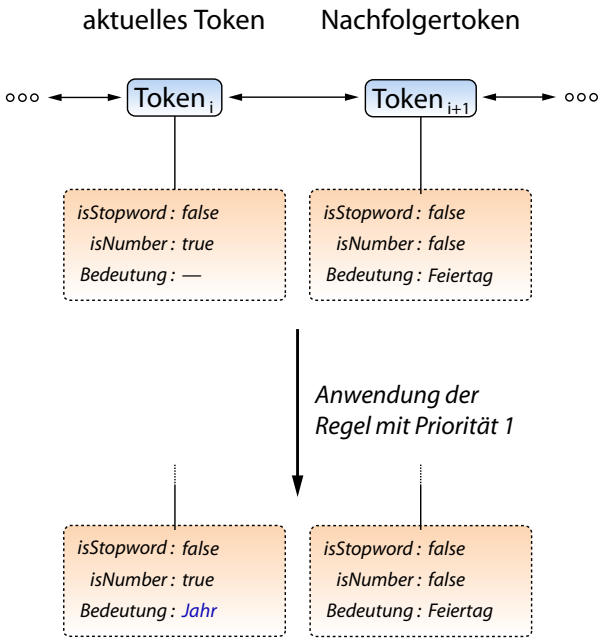


Abbildung A.1: *Veranschaulichung der ersten semantischen Regel*

Anhang B

Strukturierungsregeln

Diese Auflistung beinhaltet eine Beschreibung aller in dieser Arbeit umgesetzten priorisierten Strukturierungsregeln, welche die in Kapitel 3.9 beschriebene Informationsgruppierung realisieren.

Priorität	Regel
1	Reduziere alle jeweils aufeinander folgenden Vornamen, Feiertage, Straßennamen, Namenszusätze, Währungen und Erwerbsarten zu einer Information.
2	Ordne einen Namenszusatz dem folgenden Nachnamen, einen Zeitbezug dem folgenden Feiertag, eine römische Zahl dem vorangegangenem Straßennamen oder der vorangegangenen Richtung, dem Bauwerk die vorangegangene Anzahl, dem Geldbetrag die folgende Währung und der Prozentzahl die nachfolgende Prozentangabe zu.
3	Bilde aus einem Nachnamen eine Person-Gruppe. Wenn vor diesem Nachnamen schon eine Person-Gruppe existiert, dann ordne diesen Nachnamen der vorangehenden Person zu.
4	Füge einen Vornamen der nachfolgenden Person-Gruppe hinzu, soweit eine vorhanden ist und diese noch keinen Vornamen beinhaltet. Andernfalls lege eine neue Person mit diesem Vornamen an.
5	Füge eine Personenbeziehung der nachfolgenden Person hinzu, soweit eine vorhanden ist und diese noch keine Personenbeziehung beinhaltet.
6	Füge eine Personenbeziehung der vorangehenden Person hinzu, soweit eine vorhanden ist und diese noch keine Personenbeziehung beinhaltet. Andernfalls lege eine neue Person mit dieser Beziehung an.
7	Füge einen Beruf der nachfolgenden Person hinzu, soweit eine vorhanden ist. Andernfalls lege eine neue Person mit diesem Beruf an.
8	Bilde aus einem Jahr eine Zeitangabe-Gruppe.
9	Füge einen Feiertag der nachfolgenden Zeitangabe hinzu, soweit eine vorhanden ist und diese noch keinen Feiertag beinhaltet. Andernfalls lege eine neue Zeitangabe mit diesem Feiertag an.
10	Füge einen Monat der nachfolgenden Zeitangabe hinzu, soweit eine vorhanden ist und diese noch keinen Monat beinhaltet. Andernfalls lege eine neue Zeitangabe mit diesem Monat an.

Priorität	Regel
11	Füge einen Wochentag der nachfolgenden Zeitangabe hinzu, soweit eine vorhanden ist und diese noch keinen Wochentag beinhaltet. Andernfalls lege eine neue Zeitangabe mit diesem Wochentag an.
12	Füge einen Tag der nachfolgenden Zeitangabe hinzu, soweit eine vorhanden ist und diese noch keinen Tag beinhaltet. Andernfalls lege eine neue Zeitangabe mit diesem Tag an.

Tabelle B.1: *Strukturierungsregeln*

Anhang C

Regeln zur Überprüfung der Informationsstrukturen

In diesem Abschnitt werden diejenigen Überprüfungsregeln kurz vorgestellt, die im Rahmen dieser Arbeit beispielhaft für den Prüfmechanismus umgesetzt wurden.

Regel
Liste alle diejenigen Worte auf, denen keine Bedeutung zugeordnet werden konnte.
Liste alle diejenigen Worte auf, denen mehrere Bedeutungen zugeordnet wurden.
Liste alle Personen auf, die einzig aus einem Vornamen gebildet sind.
Liste alle Personen auf, die einzig aus einem Nachnamen gebildet sind.
Liste alle diejenigen Zeitangaben auf, die keine Angabe des Jahres beinhalten.

Tabelle C.1: *Regeln zur Überprüfung der Informationsstrukturen*

Anhang D

Struktur der Datenbank

Die aus den ER-Modellen in Abbildung 3.25 bis 3.28 ab Seite 65 resultierenden Relationenschemata werden in den folgenden Tabellen vorgestellt. Dabei wurden die einzelnen Relationen zu den verschiedenen Dokumenttypen konzeptionell voneinander abgegrenzt, indem sie verschiedenen Datenbankschemata zugeordnet wurden.

D.1 Relationenschemata für das Abkürzungsverzeichnis (Schema **abkverz**)

Relationenname	Attribute	Schlüssel
Quelldok	dokName datei	Primärschlüssel
Eintrag	eintrID dokName	Primärschlüssel Fremdschlüssel bzgl. Quelldok
Abk	abkID abk origZNr origZInh	Primärschlüssel
EintragAbk	abkID eintrID	Primärschlüssel, Fremdschlüssel bzgl. Abk Fremdschlüssel bzgl. Eintrag
Info	infoID info origZNr origZInh	Primärschlüssel
EintragInfo	infoID eintrID	Primärschlüssel, Fremdschlüssel bzgl. Info Fremdschlüssel bzgl. Eintrag

Tabelle D.1: *Relationenschemata für das Abkürzungsverzeichnis*

D.2 Relationenschemata für das Personenverzeichnis (Schema **persverz**)

Relationenname	Attribute	Schlüssel
Quelldok	dokName datei	Primärschlüssel
Eintrag	eintrID dokName	Primärschlüssel Fremdschlüssel bzgl. Quelldok
Nachname	nachnameID nachname origZNr origZInh	Primärschlüssel
EintragNachname	nachnameID eintrID	Primärschlüssel, Fremdschlüssel bzgl. Nachname Fremdschlüssel bzgl. Eintrag
Person	personID vorname persInfo origZNr origZInh	Primärschlüssel
EintragPerson	personID eintrID	Primärschlüssel, Fremdschlüssel bzgl. Person Fremdschlüssel bzgl. Eintrag
GbNr	nrID nr nrInfo	Primärschlüssel
PersonGbNr	nrID personID	Primärschlüssel, Fremdschlüssel bzgl. GbNr Fremdschlüssel bzgl. Person

Tabelle D.2: *Relationenschemata für das Personenverzeichnis*

D.3 Relationenschemata für das Grundbuch (Schema **gb**)

Relationenname	Attribute	Schlüssel
Quelldok	dokName datei	Primärschlüssel
Straße	strID dokName	Primärschlüssel Fremdschlüssel bzgl. Quelldok
Eintrag	eintrID	Primärschlüssel
StraßeEintrag	eintrID strID	Primärschlüssel, Fremdschlüssel bzgl. Eintrag Fremdschlüssel bzgl. Straße
Zeile	zeilenID zeilenTyp	Primärschlüssel

Relationenname	Attribute	Schlüssel
	origZNr origZInh origZInhForm	
EintragZeile[48]	zeilenID eintrID	Primärschlüssel, Fremdschlüssel bzgl. Zeile Fremdschlüssel bzgl. Eintrag
StraßeZeile[48]	strID zeilenID	Primärschlüssel, Fremdschlüssel bzgl. Straße Fremdschlüssel bzgl. Zeile
Info	infoID pos	Primärschlüssel
ZeileInfo	infoID zeilenID	Primärschlüssel, Fremdschlüssel bzgl. Info Fremdschlüssel bzgl. Zeile
Person	infoID vorname persBez	Primärschlüssel, Fremdschlüssel bzgl. Info
Nachname	nachnameID nachname namenszusatz	Primärschlüssel
PersonNachname	nachnameID infoID	Primärschlüssel, Fremdschlüssel bzgl. Nachname Fremdschlüssel bzgl. Person
Beruf	berufID beruf	Primärschlüssel
PersonBeruf	berufID infoID	Primärschlüssel, Fremdschlüssel bzgl. Beruf Fremdschlüssel bzgl. Person
Zeitangabe	infoID tag wochentag monat jahr	Primärschlüssel, Fremdschlüssel bzgl. Info
Feiertag	feiertagID feiertag	Primärschlüssel
ZeitFeiertag	infoID feiertagID	Primärschlüssel, Fremdschlüssel bzgl. Zeitangabe Fremdschlüssel bzgl. Feiertag
Straßenname	infoID strName nummer	Primärschlüssel, Fremdschlüssel bzgl. Info
Richtung	infoID richtung nummer	Primärschlüssel, Fremdschlüssel bzgl. Info
Bauwerk	infoID bau anzahl	Primärschlüssel, Fremdschlüssel bzgl. Info
Geldbetrag	infoID	Primärschlüssel, Fremdschlüssel bzgl. Info

Relationenname	Attribute	Schlüssel
	betrag währung	
Prozentangabe	infoID prozentzahl prozent	Primärschlüssel, Fremdschlüssel bzgl. Info
Tilgung	infoID tilung	Primärschlüssel, Fremdschlüssel bzgl. Info
Termin	infoID termin	Primärschlüssel, Fremdschlüssel bzgl. Info
Erwerbsart	infoID erwerb	Primärschlüssel, Fremdschlüssel bzgl. Info
GbNr	infoID nr	Primärschlüssel, Fremdschlüssel bzgl. Info
SchNr	infoID nr	Primärschlüssel, Fremdschlüssel bzgl. Info
FolNr	infoID nr	Primärschlüssel, Fremdschlüssel bzgl. Info
SbNr	infoID nr	Primärschlüssel, Fremdschlüssel bzgl. Info
Nr	infoID nr	Primärschlüssel, Fremdschlüssel bzgl. Info
Zahl	infoID zahl	Primärschlüssel, Fremdschlüssel bzgl. Info
Stoppwort	infoID wort	Primärschlüssel, Fremdschlüssel bzgl. Info
Unbekannt	infoID wort	Primärschlüssel, Fremdschlüssel bzgl. Info
Bedeutung	bedID bed	Primärschlüssel
UnbekBed	bedID infoID	Primärschlüssel, Fremdschlüssel bzgl. Bedeutung Fremdschlüssel bzgl. Unbekannt

Tabelle D.3: *Relationenschemata für das Grundbuch*

[48] Dieses Relationenschema hätte mit den beiden beteiligten Schemata verschmolzen werden können. Darauf wurde jedoch zu Gunsten der Übersichtlichkeit verzichtet.

Anhang E

Laufendes Beispiel

Anhand der folgenden Darstellungen soll der konkrete Verarbeitungsprozess am Beispiel eines Auszugs aus einem Grundbucheintrag veranschaulicht werden. Dabei werden aufgrund der Übersichtlichkeit zum einen die Phasen der RTF-HTML-Konvertierung und die Vergabe von IDs zur Speicherung der Informationen in der Datenbank nicht aufgeführt, da hierzu Beispiele in den entsprechenden Kapiteln zu finden sind. Zum anderen werden die Resultate einiger Phasen übersprungen, da diese ebenfalls im jeweils darauf folgenden Verarbeitungsschritt nachvollzogen werden können[48].

E.1 Quelldokument

[...]
1 ***Grundbuch Nr. 255*** *(2), fol. 77v, neues Stadtbuch Nr. 340*
3 Bude
5 Arnoldus Gützkou. Miser. 1418.
Marten Tampz, Gläser.
Peter Beneke.
Paul Lübke. sen. empt. Thom. 1678.
Paul Lübbeke. her. ibid.
Daniel Pirstorff. empt. Galli 1684.
Sehl. Daniel Pierstörpen Erben. Luciae 1703.
deßen Sohn Daniel Pierstorp. haeredit. ibid.
[...]
6 100 m. Schw. Kloster. Miseric. 1418.
75 m. dito. Thom. 1678.
25 m. dito. Galli 1684. Delet. L. a. Luc. 1837.
[...]

Abbildung E.1: *Auszug aus einem Grundbucheintrag*

[48] Für ein besseres Verständnis kann die Darstellung des Gesamtprozesses in Abbildung 3.1 auf Seite 24 herangezogen werden.

E.2 Ergebnis nach der Normalisierung

```
<!-- ... -->
<Eintrag>
  <Grundbuchnummer>Grundbuch Nr. 255 (2), fol. 77v, neues Stadtbuch Nr. 340
    <OrigZNr value="1643"/>
    <OrigZInh value="Grundbuch Nr. 255 (2), fol. 77v, neues Stadtbuch Nr. 340"/>
    <OrigZInhForm value="&lt;p&gt;&lt;i&gt;1 &lt;b&gt;Grundbuch Nr. 255 &lt;/
        b&gt;(2), fol. 77v, neues Stadtbuch Nr. 340&lt;/i&gt;&lt;/p&gt;"/>
  </Grundbuchnummer>
  <!-- ... -->
  <Eigentümer>Daniel Pirstorff. emptio Galli 1684.
    <OrigZNr value="1650"/>
    <OrigZInh value="Daniel Pirstorff. empt. Galli 1684."/>
    <OrigZInhForm value="&lt;p&gt;Daniel Pirstorff. empt. Galli 1684.&lt;/p&gt;"/
        >
  </Eigentümer>
  <!-- ... -->
  <Eigentümer>deßen Sohn Daniel Pierstorp. haereditas ibid.
    <OrigZNr value="1652"/>
    <OrigZInh value="deßen Sohn Daniel Pierstorp. haeredit. ibid."/>
    <OrigZInhForm value="&lt;p&gt;de&amp;szlig;en Sohn Daniel Pierstorp.
        haeredit. ibid.&lt;/p&gt;"/>
  </Eigentümer>
  <!-- ... -->
</Eintrag>
<!-- ... -->
```

Abbildung E.2: *XML-Dokument nach der Normalisierung*

E.3 Ergebnis nach der Auswertung der Volltextanteile

```
<!-- ... -->
<Eintrag>
  <Grundbuchnummer>
    <GbNr value="255"/>
    <SchNr value="2"/>
    <FolNr value="77v"/>
    <SbNr value="340"/>
    <OrigZNr value="1643"/>
    <OrigZInh value="Grundbuch Nr. 255 (2), fol. 77v, neues Stadtbuch Nr. 340"/>
    <OrigZInhForm value="&lt;p&gt;&lt;i&gt;1 &lt;b&gt;Grundbuch Nr. 255 &lt;/
        b&gt;(2), fol. 77v, neues Stadtbuch Nr. 340&lt;/i&gt;&lt;/p&gt;"/>
  </Grundbuchnummer>
  <!-- ... -->
  <Eigentümer>
    <Token value="Daniel" pos="1">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Nachname" dist="0.0">
        <Begriff value="Daniel"/>
      </Wörterbuch>
      <Wörterbuch name="Vorname" dist="0.0">
        <Begriff value="Danyel"/>
      </Wörterbuch>
    </Token>
    <Token value="Pirstorff" pos="2">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Nachname" dist="0.0">
```

```
        <Begriff value="Pirstorff"/>
      </Wörterbuch>
    </Token>
    <Token value="emptio" pos="3">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Erwerbsart" dist="0.0">
        <Begriff value="emptio"/>
      </Wörterbuch>
    </Token>
    <Token value="Galli" pos="4">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Feiertag" dist="0.0">
        <Begriff value="Galli"/>
      </Wörterbuch>
    </Token>
    <Token value="1684" pos="5">
      <IstStoppwort value="false"/>
      <IstZahl value="true"/>
    </Token>
    <OrigZNr value="1650"/>
    <OrigZInh value="Daniel Pirstorff. empt. Galli 1684."/>
    <OrigZInhForm value="&lt;p&gt;Daniel Pirstorff. empt. Galli 1684.&lt;/p&gt;"/
        >
  </Eigentümer>
  <!-- ... -->
  <Eigentümer>
    <Token value="deßen" pos="1">
      <IstStoppwort value="true"/>
      <IstZahl value="false"/>
    </Token>
    <Token value="Sohn" pos="2">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Personenbeziehung" dist="0.0">
        <Begriff value="Sohn"/>
      </Wörterbuch>
      <Wörterbuch name="Vorname" dist="0.0">
        <Begriff value="Sohn"/>
      </Wörterbuch>
    </Token>
    <Token value="Daniel" pos="3">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Nachname" dist="0.0">
        <Begriff value="Daniel"/>
      </Wörterbuch>
      <Wörterbuch name="Vorname" dist="0.0">
        <Begriff value="Danyel"/>
      </Wörterbuch>
    </Token>
    <Token value="Pierstorp" pos="4">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Nachname" dist="0.0">
        <Begriff value="Pierstorp"/>
      </Wörterbuch>
    </Token>
    <Token value="haereditas" pos="5">
      <IstStoppwort value="false"/>
      <IstZahl value="false"/>
      <Wörterbuch name="Erwerbsart" dist="0.0">
        <Begriff value="haereditas"/>
```

```
      </Wörterbuch>
    </Token>
    <Token value="ibid" pos="6">
      <IstStoppwort value="true"/>
      <IstZahl value="false"/>
    </Token>
    <OrigZNr value="1652"/>
    <OrigZInh value="deßen Sohn Daniel Pierstorp. haeredit. ibid."/>
    <OrigZInhForm value="&lt;p&gt;de&amp;szlig;en Sohn Daniel Pierstorp.
        haeredit. ibid.&lt;/p&gt;"/>
  </Eigentümer>
  <!-- ... -->
</Eintrag>
<!-- ... -->
```

Abbildung E.3: *XML-Dokument nach der Auswertung der Volltextanteile*

E.4 Ergebnis nach der semantischen Analyse

```
<!-- ... -->
<Eintrag>
  <Grundbuchnummer>
    <GbNr value="255"/>
    <SchNr value="2"/>
    <FolNr value="77v"/>
    <SbNr value="340"/>
    <OrigZNr value="1643"/>
    <OrigZInh value="Grundbuch Nr. 255 (2), fol. 77v, neues Stadtbuch Nr. 340"/>
    <OrigZInhForm value="&lt;p&gt;&lt;i&gt;1 &lt;b&gt;Grundbuch Nr. 255 &lt;/
        b&gt;(2), fol. 77v, neues Stadtbuch Nr. 340&lt;/i&gt;&lt;/p&gt;"/>
  </Grundbuchnummer>
  <!-- ... -->
  <Eigentümer>
    <Vorname value="Daniel" pos="1"/>
    <Nachname value="Pirstorff" pos="2"/>
    <Erwerbsart value="emptio" pos="3"/>
    <Feiertag value="Galli" pos="4"/>
    <Jahr value="1684" pos="5"/>
    <OrigZNr value="1650"/>
    <OrigZInh value="Daniel Pirstorff. empt. Galli 1684."/>
    <OrigZInhForm value="&lt;p&gt;Daniel Pirstorff. empt. Galli 1684.&lt;/p&gt;"/
        >
  </Eigentümer>
  <!-- ... -->
  <Eigentümer>
    <Stoppwort value="deßen" pos="1"/>
    <Unbekannt value="Sohn" pos="2">
      <Bedeutung value="Personenbeziehung"/>
      <Bedeutung value="Vorname"/>
    </Unbekannt>
    <Vorname value="Daniel" pos="3"/>
    <Nachname value="Pierstorp" pos="4"/>
    <Erwerbsart value="haereditas" pos="5"/>
    <Stoppwort value="ibid" pos="6"/>
    <OrigZNr value="1652"/>
    <OrigZInh value="deßen Sohn Daniel Pierstorp. haeredit. ibid."/>
    <OrigZInhForm value="&lt;p&gt;de&amp;szlig;en Sohn Daniel Pierstorp.
        haeredit. ibid.&lt;/p&gt;"/>
  </Eigentümer>
  <!-- ... -->
</Eintrag>
<!-- ... -->
```

Abbildung E.4: *XML-Dokument nach der semantischen Analyse*

E.5 Ergebnis nach der Informationsstrukturierung

```
<!-- ... -->
<Eintrag>
  <Grundbuchnummer>
    <GbNr value="255"/>
    <SchNr value="2"/>
    <FolNr value="77v"/>
    <SbNr value="340"/>
    <OrigZNr value="1643"/>
    <OrigZInh value="Grundbuch Nr. 255 (2), fol. 77v, neues Stadtbuch Nr. 340"/>
    <OrigZInhForm value="&lt;p&gt;&lt;i&gt;1 &lt;b&gt;Grundbuch Nr. 255 &lt;/
        b&gt;(2), fol. 77v, neues Stadtbuch Nr. 340&lt;/i&gt;&lt;/p&gt;"/>
```

```
  </Grundbuchnummer>
  <!-- ... -->
  <Eigentümer>
    <Person>
      <Vorname value="Daniel" pos="1"/>
      <Nachname value="Pirstorff" pos="2"/>
    </Person>
    <Erwerbsart value="emptio" pos="3"/>
    <Zeitangabe>
      <Feiertag value="Galli" pos="4"/>
      <Jahr value="1684" pos="5"/>
    </Zeitangabe>
    <OrigZNr value="1650"/>
    <OrigZInh value="Daniel Pirstorff. empt. Galli 1684."/>
    <OrigZInhForm value="&lt;p&gt;Daniel Pirstorff. empt. Galli 1684.&lt;/p&gt;"/
        >
  </Eigentümer>
  <!-- ... -->
  <Eigentümer>
    <Stoppwort value="deßen" pos="1"/>
    <Unbekannt value="Sohn" pos="2">
      <Bedeutung value="Personenbeziehung"/>
      <Bedeutung value="Vorname"/>
    </Unbekannt>
    <Person>
      <Vorname value="Daniel" pos="3"/>
      <Nachname value="Pierstorp" pos="4"/>
    </Person>
    <Erwerbsart value="haereditas" pos="5"/>
    <Stoppwort value="ibid" pos="6"/>
    <OrigZNr value="1652"/>
    <OrigZInh value="deßen Sohn Daniel Pierstorp. haeredit. ibid."/>
    <OrigZInhForm value="&lt;p&gt;de&amp;szlig;en Sohn Daniel Pierstorp.
        haeredit. ibid.&lt;/p&gt;"/>
  </Eigentümer>
  <!-- ... -->
</Eintrag>
<!-- ... -->
```

Abbildung E.5: *XML-Dokument nach der Informationsstrukturierung*

Thesen

1. Die Formate DOC und RTF eignen sich nicht zur angemessenen Auswertung von Layout- und Strukturinformationen von Dokumenten. Die Transformation in andere, besser auswertbare Formate ist daher erforderlich.

2. Die Einbeziehung von in den Originaldokumenten erkennbaren Layout- und Strukturinformationen — wie beispielsweise Schriftformatierungen — in die Analyse kann die Interpretation der Daten und deren Bedeutung entscheidend unterstützen.

3. Durch eine Normalisierung der verschiedenen Schreibvarianten und Flexionen von Worten kann die Anzahl der nötigen Begriffe in den Wörterbüchern gesenkt werden. Zum einen kann dadurch ein Performance-Gewinn bei der Analyse der Volltextanteile erzielt werden, zum anderem wird dem Nutzer die Erstellung und Anpassung der Wörterbücher erleichtert.

4. Reguläre Dokumentinhalte können mit Hilfe von Grammatiken ausgewertet werden. Durch die Abbildung der konstruierten abstrakten Syntaxbäume auf XML-Fragmente kann die Zielstruktur der identifizierten Daten beliebig beeinflusst werden.

5. Wörterbuchbasierte Verfahren im Zusammenhang mit buchstaben- und sprachorientierten Methoden haben sich als ein effektiver Ansatz zur Auswertung von Volltextanteilen herausgestellt. Bei Vollständigkeit der Wörterbuchbegriffe kann damit ein Großteil der unstrukturierten Informationen identifiziert werden.

6. Der DB2 XML Extender weist bei der Dekomposition von XML-Dokumenten mittels der `RDB_node`-Zuordnung Beschränkungen auf, die den Einsatz bei umfangreichen Dokumenten mit vielen abzubildenden Daten verhindern. Ein eigener Ansatz zur Speicherung der Daten in relationalen Tabellen muss daher gefunden werden.

7. Semantische Regeln sind hilfreich zur Eliminierung von Mehrdeutigkeiten und zur Identifikation von Informationen, die nicht mit Hilfe der Wörterbuchbegriffe bestimmt werden konnten.

8. Die Ergebnisse der Volltextauswertung und der semantischen Analyse können entscheidend durch die Wahl der Parameter — wie Schwellenwert für die Wortabstandsberechnung, Selektionsmethode etc. — und die sinnvolle Definition von semantischen Regeln beeinflusst werden.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, 29. Oktober 2004