Diplomarbeit

Torben Hoffmann

Entwicklung einer Webapplikation zur automatischen Erstellung von Signifikanzvektoren in klassifizierten Texten

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Torben Hoffmann

Entwicklung einer Webapplikation zur automatischen Erstellung von Signifikanzvektoren in klassifizierten Texten

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Thomas Schmidt
Zweitgutachter : Prof. Dr. Ing. Martin Hübner

Abgegeben am 23. August 2005

Torben Hoffmann

Thema der Diplomarbeit

Entwicklung einer Webapplikation zur automatischen Erstellung von Signifikanzvektoren von Schlüsselworten in klassifizierten Texten

Stichworte

Automatische Klassifikation, Klassifikationsschemata, Textanalyse

Kurzzusammenfassung

Diese Arbeit behandelt die Konzeption und Erstellung einer Webapplikation zur automatischen Erstellung von Signifikanzvektoren von Schlüsselworten in klassifizierten Texten am Beispiel des Klassifikationsschemas der ACM CSS. Eingangs werden die theoretischen Grundlagen der Klassifikationsschemata und einer automatische Klassifikation, sowie die Erstellung von Signifikanzvektoren erarbeitet. Auf der Grundlage der gewonnen Kenntnisse und einer spezifischen Problemdiskussion wurde sodann ein Lösungskonzeptes für das behandelte Online-Klassifikationssystem entwickelt.

Die Umsetzung des Systems besteht im Kern aus einer Textanalyse, die mittels Filterung des Klartextes unterschiedlicher Dateiformate für die Extraktion betroffener Schlüsselworte bis hin zur Speicherung der analysierten Daten verantwortlich ist. Die Visualisierung der Ergebnisse erfolgt über die Oberfläche der Webapplikation.

Torben Hoffmann

Title of the paper

Development of a Web Application for automated evaluation of keywords significances in classified papers

Keywords

Automated content classification, classification schemes, text analysis

Abstract

This diplom thesis treats the concept and implementation of a Web application for automated evaluation of keywords significances in classified papers. As an example of the classification pattern the ACM CSS is used.

After introducing the theoretical bases of the classification patterns and automatic classification, the problem of harvesting signifance vectors from an online classification system is discussed.

The main part of the system consists in a text analysis. This analysis entails the filtering of the plain text of different file formats. Moreover it is responsible for the extraction of the used keywords up to the storage of the analyzed data. The visualization of the results is made by the surface of the Web application.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

1 Einleitung

Heutzutage haben wir nicht nur den Zugang zu gewaltigen Informationsmengen, sondern wir produzieren und veröffentlichen selbst eine enorme Flut an Informationen. Privatpersonen veröffentlichen 600 mal mehr Informationen als der kommerzielle bzw. öffentliche Sektor. Es gibt also nicht nur eine Massenproduktion an Informationen, sondern auch eine Informationsproduktion durch die Massen. [LV00]

Blickt man in die Geschichte, so gab es im Mittelalter noch Gelehrte, die sich mit fast dem gesamten Wissen ihrer Zeit beschäftigen konnten. Das ist heute undenkbar.
Ein Beispiel: Die jährliche, weltweite Informationsproduktion liegt bei ein bis zwei Milliarden Gigabyte an Daten. Das entspricht einem Durchschnitt von 250 Megabyte pro Person weltweit. Alleine die Information auf die man über das Internet Zugriff hat, liegt bei 7500 Terrabyte[1]. Dazu kommen rund 1 Million Bücher (welches einer Menge von ca. 8 Terrabyte an Daten entspricht), die jährlich weltweit veröffentlicht werden. [LV00]

20 000 wissenschaftliche Artikel werden täglich weltweit zu den unterschiedlichsten Themen veröffentlicht. Das entspricht einer Gesamtanzahl von ca. 4 Millionen Fachveröffentlichungen pro Jahr. 1950 lag dieser Wert bei gut einem Zehntel davon. Weltweit erscheinen heute zwischen 100 000 und 200 000 Fachzeitschriften. [MG02]

Hieraus kann man nur erahnen mit was für einer Menge an Informationen man es heutzutage zu tun hat. Vor allem im frei zugänglichen Internet. Doch wie findet man nun seine persönlich verwertbare Information in dieser Masse von unstrukturierten Informationen? Das Ziel ist es, möglichst schnell eine passende Information zu seinem individuellen Problem in dieser Gesamtmenge von Daten zu finden. Diesen Vorgang kann man gut mit der berühmten „*Suche nach der Stecknadel im Heu*“ vergleichen.

[1] Basierend auf Rohdaten, inklusive Duplikationen und mit dem Internet verbundenen Datenbanken

Keith Davidson[2] hat in seinen Untersuchungen Interessantes herausgefunden: Arbeitnehmer benötigen etwa 3 Stunden pro Woche um nach bestimmten Informationen zu suchen. [DAV94]

Je schneller man also an die gesuchte Information kommt, desto besser ist es für ein Unternehmen oder eine Person. Denn noch nie war das Sprichwort „*Zeit ist Geld*" so wahr wie heute. Daher ist heutzutage auch laut [NOH] die Information der entscheidende Wettbewerbsfaktor in allen Branchen und Märkten.

Doch nicht nur in kommerziellen Unternehmen kämpft man mit der Informationsflut, sondern auch in der Bildung bzw. Ausbildung wird man mit diesem Problem konfrontiert. Zu dem Überangebot kommt auch noch das Problem des Wahrheitsfaktors, sprich der Qualität, hinzu. Ziel muss es also sein, die Informationen so zu erschließen, dass sie auffindbar werden, um die Qualität der Angebote entsprechend zu verbessern. [DIEPO]

Um zu einer Lösung zu kommen, muss man sich also überlegen, welche Möglichkeiten es gibt Informationen so zu verwalten, zu strukturieren und zur Verfügung zu stellen, dass das Auffinden der gesuchten Information beschleunigt und vereinfacht wird. So gibt es heute viele Ansätze zur Informationsgewinnung bzw. zu deren Auffindung. Diese kann man im Wesentlichen in folgende Gruppen einteilen: in die Gruppe mit dem Ansatz über Worte und Begriffe, den statischen und den semantischen Ansatz. Beim ersten Ansatz können einerseits alle Wörter in die Betrachtung miteinbezogen werden, oder aber es kommt zu einer Filterung der Wörter. Die Filterung geschieht mit Hilfe von Stemming oder Verwendung von Thesauri und Einbeziehung von Strukturinformationen in einem Dokument; so geht dieser Ansatz in Richtung semantischer Ansatz. Benutzt man hingegen Häufigkeitsanalysen von Wörtern oder die Kategorisierung, so befindet man sich in der Gruppe der statischen Verfahren.

Um nun die Auffindung gespeicherter, relevanter Informationen zu verbessern, müssen diese, anhand bestimmter Kriterien, identifizierbar gemacht werden. In der traditionellen Wissensspeicherung, wie z.B. in Bibliotheken oder Archiven, geschieht

[2] Executive Director der Firma Xplor International, http://www.xplor.com

diese Identifizierung durch Angabe von Schlagwörtern, Autoren oder Titeln. Diese zusätzlichen Informationen werden dem Dokument (den eigentlichen Informationen), beigefügt, um die spätere Auffindung zu erleichtern. Bei diesen Informationen handelt es sich um so genannte Metadaten. Genau dieser Ansatz wird auch im elektronischen Bereich verfolgt. Da man Metadaten prinzipiell für alle Formen von Multimedia-Objekten verwenden kann, stellen sie in Kombination mit der automatischen Klassifikation und Content Managment Systemen, einen realen Lösungsansatz für das Problem der Informationsauffindung dar.

Ein großes Problem liegt dabei jedoch in der Verwendung von Metadaten. Leider werden sie bis heute von den publizierenden Autoren selten benutzt. Ein Lösungsansatz für dieses Problem ist die automatische Annotation mit Metadaten dieser Dokumente. Hiermit beschäftigt sich unter anderen das HyLOS Projekt. [EHS05]

Das Hypermedia Learning Object System (HyLOS) ist eine Plattform, die Lernobjekte verarbeitet. Es werden unter anderem aus vorlesungsbegleitenden Aufnahmen durch Analyse und automatische Klassifikation IEEE LOM Lernobjekte[3] geschaffen. Die Basis für diese automatische Klassifikation und eine hypermediale Präsentation in Online-Lernsystemen bildet dabei den Inhalt von identifizierenden und beschreibenden Metadaten, die aus kontrollierten Vokabularen zunächst den Lernobjekten hinzugefügt werden müssen. Daraufhin können dann diese Lernobjekte durch Abgleich der hinzugefügten Schlüsselworte mit einzelnen Knoten einer Taxonomie in die entsprechenden Klassen automatisch klassifiziert werden. Die Knoten werden dabei durch die repräsentativsten Schlüsselworte aus den klassifizierten Lernobjekten abgebildet. [EHS05]

Für eine gut funktionierende Klassifikation oder der automatischen Annotation mit identifizierenden Metadaten ist eine Gewichtung bzw. die Signifikanz der einzelnen Schlüsselworte in ihren Klassen notwendig. Mit so entstehenden Signifikanzvektoren der Taxonomieknoten bzw. ihrer Klassen, kann nun mittels Vektorenvergleiches das neu einzuordnende Objekt in die Klassen mit größter Übereinstimmung eingeordnet werden.

[3] Auch eLearning Objects (eLOs) genannt

Wie man den Schlüsselworten jedes Knotens einer Taxonomie eine Signifikanz zukommen lassen kann, sowie die Vorrausetzungen als auch die Möglichkeiten einer automatischen Klassifikation sind Inhalt dieser Diplomarbeit.

1.1 Zielsetzung

In dieser Diplomarbeit wird eine Webapplikation realisiert. Diese errechnet auf der Basis von manuell erstellten, klassifizierten Schlüsselwortlisten und digitalen Dokumenten die Signifikanz jedes einzelnen Schlüsselwortes der Schlüsselwortliste einer Taxonomie. Als Klassifikationsschema soll hier, aufgrund ihrer Verwendung im HyLOS Projekt, die ACM CSS angewendet werden. Die Dokumente werden manuell über eine Klassifikationsmaske einer oder mehreren Klassen dieser Taxonomie zugeordnet. Mittels Textanalyse wird dann die Signifikanz der einzelnen Schüsselworte jeder Klasse herausgefunden. Auf diesem Weg können Signifikanzvektoren für jede Klasse der Taxonomie gebildet werden. Diese können sodann von anderen Systemen oder Applikationen, beispielsweise als Basis einer automatischen Klassifikation mittels Vektorenvergleiches, verwendet werden.

Um möglichst viele zu klassifizierende Dokumente für die Signifikanzerrechnung zu finden, wird neben einem Datei Upload auch eine Websuche mittels des Google-API in die Applikation implementiert.

1.2 Gliederung

Die Diplomarbeit gliedert sich in 9 Kapitel. Nach der Einleitung soll im zweiten Kapitel ein Überblick zu dem Thema Metadaten gegeben werden, sowie eine Vorstellung des HyLOS Projektes. Im dritten Kapitel werden Strukturierungsmöglichkeiten mittels existierender Klassifikationsschemata beschrieben. Das vierte Kapitel zeigt Möglichkeiten und Verfahren einer automatischen Klassifikation. Abschließen werden die theoretischen Grundlagen im fünften Kapitel mit der Herleitung zur Berechnung der Signifikanzvektoren anhand der sich ergebenden Problematik.

Basierend auf diesen theoretischen Grundlagen wird dann im sechsten Kapitel mittels der herausgefundenen Anforderungen ein Konzept für die zu realisierende

Webapplikation vorgestellt. Folgend von dem Design und der Gesamtarchitektur der Applikation im siebten Kapitel, werden hier zusätzlich die genauen Funktionen der Applikation vorgestellt und erläutert. Im achten Kapitel wird die entwickelte Applikation unter den Aspekten der Funktionalität und der Performance getestet. Abschließen wird die Diplomarbeit mit einer Zusammenfassung und einem Ausblick auf mögliche Erweiterungen dieser Applikation.

1.3 Begriffsdefinitionen

Da sich im Laufe der Zeit in der Klassifikationstheorie häufig Synonyme für Begriffe aus diesem Themengebiet entwickelt haben, sollten die grundlegendsten Wortbedeutungen genauer definiert werden. Dieses wird in den folgenden Absätzen getan und soll für alle folgenden Kapitel Gültigkeit haben. In vielen Situationen ist die Definition nicht immer so eindeutig einzuhalten, wie die des Deutschen Instituts für Normung e. V. in der Norm 32 705. Sie treffen den Inhalt, meiner Meinung nach, aber sehr passend und werden daher im Folgenden zitiert:

- *„Ein Gegenstand ist ein beliebiger Ausschnitt aus der wahrnehmbaren oder vorstellbaren Welt Dieser kann durch Sprache dargestellt werden. (Gegenstände können nicht nur materieller Art (z.B. eine Tasse) sondern auch nichtmaterieller Art (z.B. ein Gedanke) sein)*
- *Eine Klasse ist die Zusammenfassung derjenigen Begriffe, die mindestens ein identisches Merkmal haben.*
- *Ein Merkmal ist ein Begriffselement, das durch Aussage über die Eigenschaft eines Gegenstandes festgelegt wird.*
- *Ein Klassem (Synonym: Klassifikatorisches Merkmal) ist dasjenige gemeinsame Merkmal von Begriffen, das zur Bildung einer Klasse benutzt wird und diese von anderen Klassen unterscheidet.*
- *Ein Klassifikationssystem ist die strukturierte Darstellung von Klassen und der zwischen ihnen bestehenden Begriffsbeziehung.*
- *Eine Notation im Klassifikationssystem ist eine nach bestimmten Regeln gebildete Zeichenfolge, die eine Klasse, einen Begriff oder eine Begriffskombination repräsentiert und deren Stellung im systematischen Zusammenhang abbildet.“*

[DIN32705]

Durch diese Definition wird aber nicht ausgeschlossen, dass z.B. die Wörter „Klassifikation“ und „Systematik“ synonym für „Klassifikationssystem“ verwendet werden können. So wird eine Bestimmung einer bestimmten Systemstelle eines Dokumentes auch als „klassifizieren“ bezeichnet, deren Bedeutung so auch in dieser Arbeit Verwendung finden soll. [HAC00]

Wenn in dieser Arbeit von Dokumenten die Rede ist, soll immer ein „digitales Dokument“ gemeint sein. Da es hierfür keine einheitliche Definition gibt, sollen hiermit Materialien bezeichnet werden, die im digitalen Zustand vorliegen. Ich denke die Definition von Endres und Fellner passt in diesem Beziehung sehr gut und soll im weiteren Verlauf verwendet werden:

> *„Ein digitales Dokument ist eine in sich abgeschlossene Informationseinheit, deren Inhalt digital codiert und auf einem elektronischen Datenträger gespeichert ist, so dass er mittels eines Rechners genutzt werden kann.“*
> [EF00]

Prinzipiell kann der Begriff „*Klassifikation*“ als Oberbegriff für alle Verfahren gesehen werden, welche sich mit der Einteilung von Daten und Objekten beschäftigen. Daher wird auch in dieser Arbeit, wenn von „*Klassifikation*“ oder von „*klassifizieren*“ die Rede ist, eine Einteilung von Daten bzw. dem Dokument in eine bereits von vornherein bekannte Klasse gemeint sein. Unterschieden werden muss hier vom Clustering.

2 Metadaten

Um die Auffindung digitaler Dokumente verbessern zu können, müssen diese zunächst anhand bestimmter Kriterien identifizierbar gemacht werden. In der klassischen Wissensspeicherung, wie beispielsweise den Bibliotheken, geschieht dieses durch Angabe von Schlagwörtern, Autoren oder Titeln. Es werden den eigentlichen Informationen also weitere beschreibende Informationen hinzugefügt. Hierbei handelt es sich um die so genannten Metadaten. Auch bei digitalen Dokumenten kann die Anreicherung oder Auswertung der Metadaten sehr nützlich sein.

Daher soll dieses Kapitel einen kurzen Überblick über das Thema Metadaten geben sowie die Vorgehensweise des eingangs erwähnten HyLOS Projektes beschreiben.

2.1 Allgemeine Metadaten

Laut begrifflicher Definition sind Metadaten Daten über Daten. Ihre Aufgabe ist es, eine Informationseinheit so zu charakterisieren, dass der User den Inhalt, ihren Zweck, ihre Herkunft und evtl. sogar die Art ihrer Verwendung versteht. [BOH00]

Metadaten sind „Informationen über andere Daten (Dokumente, Datensammlungen, Bilder, Server, etc.), die in einer Form gehalten werden, dass sie die Recherche, das Retrieval und die Nutzung der Primärdokumente ermöglichen, erleichtern und ggf. bestimmen.“ Weitere Definitionen weisen darauf hin, dass es sich bei Metadaten um ein Surrogat handelt, welches dem Benutzer Schlüsse zu dem beschriebenen Objekt erlaubt, ohne dass er umfassende vorherige Kenntnis über dessen Existenz oder Charakteristika haben muss. [BOH00]

Somit können alle Arten von bibliographischen Informationen, Zusammenfassungen, Abstracts etc., die die ursprünglichen Objekte auf eine bestimmte Art beschreiben, Metadaten sein. Obwohl der Begriff der elektronischen Informations- und Datenverarbeitung entstammt, trifft er auch auf Instrumente zu, die zumindest ursprünglich mal außerhalb des Internets angesiedelt waren. Katalogkarten oder Titelaufnahmen in Bibliotheken sind beispielsweise typische Formen von Metadaten,

die unter anderen zur Beschreibung, Verwaltung und zur Auffindung von Medien dienen. Die Literaturbeschreibung in Bibliothekskatalogen ist im Grunde nichts anderes als die Auszeichnung mit Metadaten, wobei bei der Formalerschließung eher nach formalen und bei der Sacherschließung nach inhaltlichen Kriterien beschrieben wird. [HAC00]

Da im Laufe der Zeit der Begriff Metadaten immer allgemeiner benutzt wurde, ist es hilfreich, diese stets vor dem Hintergrund der Prozesse zu betrachten, die sie unterstützen oder erleichtern sollen. So kann man in Anlehnung an Bearman und Sochats eine Unterscheidung von Metadaten in folgende Arten vornehmen:

- Identifikations- und Nachweiszwecke (z.B. Titel, Autor)
- Nutzungs- und Beschaffungskonditionen (z.B. Nutzungsrechte, Preise)
- Strukturellen Aspekte (z.B. Art des Dokuments, Gliederung in Kapitel)
- Kontext (z.B. Querverweise, ähnliche Quellen)
- Inhalt (z.B. Schlagwörter, Abstract)
- Nutzungs- und Wirkungsgeschichte (z.B. frühere Versionen)
- Technische Aspekte (z.B. Datenformat, Speicherbedarf)

[BS95]

Mit der Grundidee zusätzlich zu den eigentlichen Ressourcen beschreibende Informationen abzulegen, die sowohl für den Menschen als auch für Computer gestützte Systeme lesbar sind, ist eine Lösung gefunden worden, den Herausforderungen des Wissens- und Dokumentenmanagements entgegen zu treten. Insbesondere die Qualität beim Navigieren und beim Suchen kann mit Hilfe von Metadaten erheblich gesteigert werden. [BS95]

Ein Problem im Bereich der Metadaten ist ihre Verwendung in digitalen Informationen bzw. Dokumenten. Die meisten Autoren vergessen oder verzichten auf eine Einteilung nach einem Schema oder der Verwendung von Metadaten. Und selbst, wenn Metadaten verwendet werden, ist die Konsistenz ihrer Formate nicht sichergestellt, wodurch eine spätere Auswirkung erheblich erschwert wird. [PIE01]

Ein Ansatz zur Lösung dieses Problem ist, die Autoren schon bei der Veröffentlichung ihrer Dokumente dazu anzuregen, aus einer vorgegebenen Menge von Metadaten auszuwählen. Durch die vorgegebenen Eingabemöglichkeiten kann sich der Autor an existierenden Themenstrukturen orientieren und gleichzeitig einen Standard zur Themenzuordnung einzuführen. Ein anderer Ansatz wäre die automatische Erzeugung der Metadaten durch Analyse der Inhalte der jeweiligen Dokumente. Die Kombination beider Ansätze wäre ein weiterer Lösungsansatz. So werden dem Autor durch eine automatische Analyse des Dokumentes entsprechende Vorschläge bezüglich der im Dokument gefundenen Schlüsselworte angeboten. Hier kann der Benutzer dann die seiner Meinung nach relevanten Metadaten auswählen. [HOF02]

Basis hierfür ist jedoch ein fester Wertevorrat der Metadaten. Im Laufe der Zeit haben sich so einige Standards zur Speicherung und Verwaltung von Metadaten entwickelt.

Das Ressource Description Framework (RDF) definiert beispielsweise eine standardisierte, XML-basierte Form um die Metadaten zu repräsentieren. Der Standard selbst definiert keine neuen Attribute wie beispielsweise HTML-Metatags. Er stellt Möglichkeiten zur Verfügung, diese Attribute und die zugehörige Semantik selbst zu entwickeln. Für Bibliothekskataloge wären dieses, wie schon erwähnt, Attribute wie „Autor“ oder „Titel“. Die Definition dieser Attribute wird durch das RDF Schema realisiert. Das RDF Schema erweitert RDF um Modellierungskonstrukte, ähnlich wie XML Schema bei XML. [PKK]

Während RDF also selbst nur das Grundmodell definiert, also die Tatsache, dass eine Ressource „*Properties*“ mit Werten haben kann, ist es dem RDF Schema möglich, die zu verwendenden Metadatenelemente genauer festzulegen. Jedoch muss dieses je nach Anwendungsgebiet individuell geschehen. Um die Interoperabilität, insbesondere bei digitalen Dokumenten im Internet, zu ermöglichen, ist es wünschenswert, gewisse universelle Metadatenelemente als eine Art Grundwortschatz festzulegen. Somit könnte das RDF Schema auch als eine Art „*Schemata-Spezifikationssprache*“ gesehen werden. [PKK]

Auf dieser Idee baut die Dublin-Core-Initiative[4] auf. Die Idee ist es, mit einer festgesetzten Menge von Elementen das Auszeichnen von Dokumenten und anderen Ressourcen zu standardisieren. [PKK]

2.1.1 Dublin Core

Ziel dieser Initiative war zunächst eine Kernmenge semantischer Beschreibungselemente, den Dublin Core Metadata Element Set, für webbasierte Ressourcen zu schaffen. Diese sollten allgemein die Auszeichnung im Web und damit die Suche und das Retrieval erleichtern. Hierzu wurde im März 1995 in Dublin, Ohio, ein erster Workshop abgehalten, deren Teilnehmer in erster Linie aus den Bereichen der Bibliotheken, Archive, Geisteswissenschaften und Geographie kamen. [DCMI02]

Nach kurzer Zeit zeigte sich bei der Entwicklung des Internets ab, dass eine einfache Menge von ursprünglich 13 Beschreibungselementen den unterschiedlichsten Anforderungen, die die verschiedenen Objekttypen und Communities an einen Metadatenstandard stellen, nicht Stand halten und somit nicht als ausschließlicher Standard für das Internet dienen kann. Daher hat sich die Initiative heute unter anderen das Hauptziel gesetzt, den gemeinsamen semantischen Kern, der in den verschiedenen Disziplinen fortexistierenden, spezifischen Metadatenstandards zu repräsentieren und eine übergreifende Suche zu ermöglichen. Dublin Core soll als der kleinste, gemeinsame Nenner verschiedener Standards dienen, der eine semantische Interoperabilität zwischen ihnen ermöglicht. [BOH00] [WEI97]

Das Dublin Core Element Set umfasst aktuell nach der Version 1.1 folgende Kernelemente:

[4] http://www.dublincore.org

Dublin Core Element Identificator	Bedeutung
Title	Name der Ressource
Creator	Für die Ressource verantwortlicher Autor
Subject	Thema der Ressource; typischerweise Schlagworte, Klassifikationscodes
Description	Inhaltszusammenfassung oder Abstract.
Publisher	Verleger, bzw. Herausgeber der Ressource
Contributers	An der Entstehung der Ressource beteiligte Personen oder Organisationen
Date	Datum für ein bestimmten Bearbeitungsstand; typischerweise Erstellungs- oder Bereitstellungsdatum, Verwendung eines Notationsschemas empfohlen
Type	Art oder Genre der Ressource
Format	Form der physischen oder digitalen Ausprägung der Ressource (Format, Dateityp)
Identifier	Eine eindeutige Identifikation für die Ressource (URL, ISBN, etc.)
Source	Quelle, von der die Ressource abgeleitet ist
Language	Sprache des intellektuellen Inhalts; Verwendung eines Notationsschemas wird empfohlen
Relation	Referenz zu verwandten Ressourcen
Coverage	Von der Ressource erfasster geographischer oder zeitlicher Bereich
Rights	Rechtliche Aspekte bezogen auf die Ressource und deren Inhalt

Tabelle 2-1 Kernelemente des Dublin Core Element Sets [DCMI1999]

Alle diese Eigenschaften sind optional und wiederholbar. Die Reihenfolge bei Angabe der Eigenschaften ist beliebig. Die Einbettung in HTML erfolgt beispielsweise mit Meta-Tags, die Einbindung in XHTML und XML mit RDF.

2.1.2 Learning Object Metadata (LOM)

Nicht nur bei der Repräsentation von Wissen, sondern insbesondere auch bei der Strukturierung von Lerninhalten wird die Notwendigkeit von Standards und Interoperabilität von Informationssystemen für Lernsysteme zwingend notwendig. So wurde 1999 von dem Learning Technology Standard Committee (LTSC) der IEEE ein

Standard P1484.12 unter dem Namen LOM (Learning Objects Metadata) vorgestellt. [IEEE]

Hier werden Lernobjekte als beliebig digitale und auch nicht digitale Einheiten definiert, wie z.B. Videoobjekte, Bilder, Audioobjekte, Texte, Software-Tools, Simulationen etc. LOM definiert nicht ein generelles Schema für Lernobjekte, sondern stellt neun unterschiedliche Kategorien zur Verfügung:

LOM Element	Bedeutung
1. General	Allgemeine Informationen über ein Lernobjekt
2. Lifecycle	Informationen über den Lebenszyklus und aktuellen Stand des Lernobjekts
3. Meta-Metadata	Informationen über die verwendeten Metadaten über das Lernobjekt
4. Technical	Informationen über technische Details des Lernobjekts
5. Educational	Informationen über pädagogische und speziell didaktische Eigenschaften des Lernobjekts
6. Rights	Informationen über die Nutzungsbedingungen des Lernobjekts
7. Relation	Informationen über Beziehungen zu anderen Lernobjekten
8. Annotation	Kommentare zum pädagogischen und didaktischen Einsatz des Lernobjekts und zusätzliche Informationen
9. Classification	Information über die Zuordnung des Lernobjekts in eine bestimmte Lernobjekt-Klasse eines Klassifikationsschemas

Tabelle 2-2: Elemente des Metadatenstandards Learning Objects Metadata (LOM) [IEEE]

Die Elemente in LOM sind somit hierarchisch in Kategorien angeordnet. Sie besitzen verschiedene, festgelegte und standardisierte Datentypen, die jeweils obligatorisch oder optional anzugeben sind. Es besteht eine gewisse Kompatibilität zwischen LOM und Dublin Core. LOM ist zwar komplexer als der Dublin Core Standard, doch lassen sich Dublin Core Elemente komplett auf die LOM Unterelemente abbilden. Die Kategorien bilden eine Art Baumstruktur, bestehend aus Stamm und Zweigen, wobei die einzelnen Werte nur in den Blättern stehen dürfen. Die Struktur ist der folgenden Abbildung zu entnehmen. [HOD01]

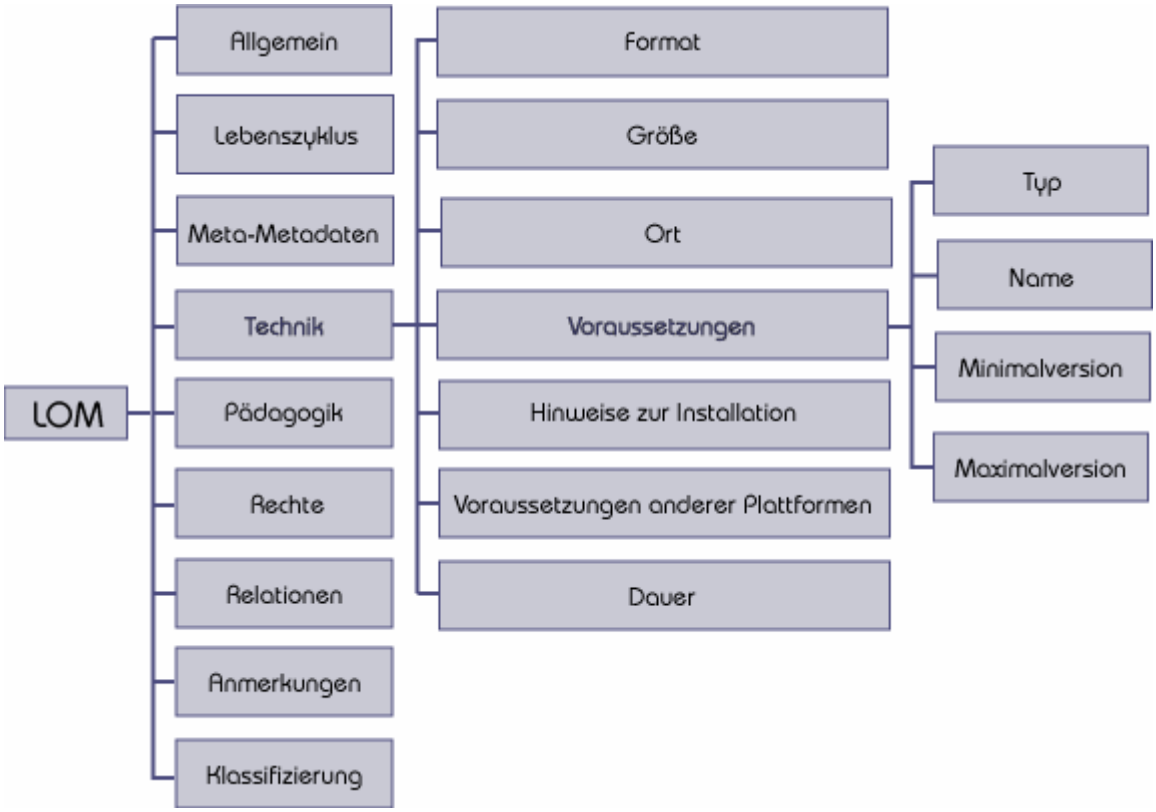

Abbildung 2-1 Struktur des Metadatenstandards LOM [IEEE]

2.2 Hypermedia Learning Object System (HyLOS)

Hierbei handelt es sich um ein Content Management System, welches konsequent den Ansatz der E-Learning Objekte gem. dem IEEE LOM folgt und auf der Entwicklungsplattform Media Information Repository basiert. Hierbei handelt es sich um ein offenes Hypermediasystem, welches die Standards XML, JNDI und Corba unterstützt. Für die Autoren von Lernobjekten stellt diese Plattform unter anderem Autorenwerkzeuge wie die verteilte Autorenumgebung und ein flexibles Content Management zur Verfügung. [EHS05]

Der LOM Standard hat ein international akzeptiertes und facettenreiches Vokabular zum Aufbau strukturierter Metadaten etabliert. Trotzdem versuchen viele der gegenwärtigen Autorenanwendungen die Menge der Metadaten zu reduzieren, weil angenommen wird, dass die Reichhaltigkeit der LOM Daten als ein Problem bei der Inhaltsproduktion angesehen wird. Daher verfolgt diese Autorenumgebung den Ansatz, dem Autor eine effektivere Inhaltsangabe zu ermöglichen, indem die Anzahl der notwendigen Eigenschaften reduziert wird. Zur Einhaltung des LOM Standard

kann aber trotzdem zusätzlich die gesamte Metadatenmenge bearbeitet werden. [EHS05]

Eines der primären Ziele des Projektes ist die schnelle und mühelose Gewinnung von Lernobjekten aus automatisierten Vorlesungsaufzeichnungen. Hierbei bildet die Annotation der Lernobjekte mit Metadaten die Grundlage für zugrunde liegende semantische Prozesse. Bei der Aufzeichnung einer Vorlesung werden beispielsweise die präsentierten Medien erfasst und in einzelne Lernobjekte eingeteilt. [EHS05]

Das Problem, welches sich für die aufgezeichneten Daten ergibt, ist das Fehlen der für eine semantische Auswertung notwendigen Metadaten. Deshalb müssen diese Lernobjekte zunächst mit Schlagwörtern angereichert werden. Für diesen Zweck wird jede zur Verfügung stehende Quelle wie die Informationen der in den Folien enthaltenden Texte oder in den Sprachaufnahmen enthaltenden Sprachsequenzen genutzt. Ist in diesen Quellen eine ausreichende zusammenhängende Textmenge vorhanden, so können hier die etablierten Verfahren des Information Retrieval zum Einsatz kommen, um die Schlüsselworte zu extrahieren. Ist jedoch nicht genügend Text vorhanden, was beispielsweise in den meisten Fällen bei Präsentationsfolien der Fall ist, so sind statistische Verfahren hier unbrauchbar und es müssen andere Lösungen gefunden werden. Hier wird der Ansatz verfolgt, die aufgezeichneten Lehrmaterialien gegen kontrollierte Schlüsselwortvokabulare zu testen. [EHS05]

Ein Problem beim Aufbau dieser kontrollierten Schlüsselwortvokabulare besteht jedoch in der Findung repräsentativer Schlagwortlisten. Diese müssen zum einen klein genug sein um erfolgreich sprecherunabhängige Spracherkennung durchführen zu können, zum anderen aber auch allgemein genug, um das Lernobjekt uneingeschränkt mit passenden Schlüsselworten anzureichern. In der Praxis ist nämlich davon auszugehen, dass die Spracherkennung nicht auf den jeweiligen Sprecher trainiert werden kann und folglich eine kontinuierliche Spracherkennung von verwertbarer Qualität nicht gegeben ist. [EHS05]

Eine brauchbare Qualität wird hingegen aber bei der Erkennung von einzelnen, sprecherunabhängigen Schlüsselworten in eingeschränkten Vokabularen erreicht. So

kann durch eine desto präzisere Eingrenzung der zu erkennenden Schlagworte eine desto bessere Erkennungsrate erzielt werden.

Zur Bildung solcher Vokabulare wird ein adaptives Verfahren eingesetzt, deren Grundlage eine Menge bereits existierender Lernobjekte oder anderer Dokumente bilden, die mit Schlüsselworten versehen sind oder durch Extraktionsverfahren bestimmt werden können. Neue, mit Schlüsselworten versehene Lernobjekte können später nach manueller Verifizierung das Ergebnis weiter verfeinern, indem diese in die Trainingsmenge aufgenommen werden. [EHS05]

Die Schüsselworte werden analog zu dem Algorithmus von Maly, Zubair und Anan [MZA01] dazu verwendet, um Knoten eines Taxonomiebaumes mit repräsentativen Schlüsselworten zu versehen. Dabei werden die Schlüsselworte vom Allgemeinen hin zum Speziellen absteigend gemäß dem Hierarchiebaum der Taxonomie angeordnet.

Die so vom Spracherkennungssystem zu findenden Schlüsselworte werden nun für eine automatische Klassifizierung herangezogen. Das Lernobjekt wird dann mittels der erkannten Worte innerhalb der Taxonomie identifiziert und durch die selektierten Knoten der Taxonomie zugeordnet. Das für die Klassifizierung der Lernobjekte verwendete Klassifikationsschema ist die auf den Informatikbereich spezialisierte ACM CSS. [EHS05]

3 Klassifikationsschemata

Eine thematische Themenzuordnung ist eine weitere gute Möglichkeit, die Auffindbarkeit von Informationen bzw. Dokumenten zu verbessern. Durch eine vordefinierte Struktur wird zum einen die Navigation in Themenbereiche ermöglicht, zum anderen können so auch themenverwandte Dokumente schnell aufgefunden werden.

Zu Beginn jeder Klassifikation stellt sich zunächst immer die Frage, was überhaupt klassifiziert werden soll und anhand welcher Kriterien diese Entscheidung bzw. Klassifizierung erfolgen soll. So könnte man z.B. nach einer Marke oder einer Farbe klassifizieren. Daten in Unternehmen könnten hingegen nach örtlichen Aspekten (z.B. nach Abteilungen) oder zeitlichen Aspekten (z.B. Eingang der Dokumente) sortiert werden. Elektronisch verfügbare Daten könnten hingegen wieder nach Dateinamen, Größe oder Dokumenttyp (Text, HTML usw.) eingeteilt werden. Diese Beispiele sollen aufzeigen, dass eine nahezu unendliche Menge an Klassifikationskriterien und damit auch an Klassifikationsschemata vorhanden ist.

In diesem Kapitel soll eine Übersicht über die grundsätzlichen Möglichkeiten der Themenzuordnung gegeben werden. Eine Themenstruktur legt fest, welche Beziehung die einzelnen Themengebiete zueinander haben. Es werden Klassifikationsschemata als eine Möglichkeit der Themenzuordnung mit hierarchisch organisierten Themenbereichen vorgestellt, sowie ihre praktische Anwendung anhand der Beispiele der DDC und der ACM CSS gezeigt.

Ein Klassifikationsschema schafft die strukturellen Vorraussetzungen für eine Klassifikation. Die Klassen, bzw. die Themen stehen dabei in einem hierarchischen Über- bzw. Unterordnungsverhältnis zueinander. Die Aspekte der höher liegenden Hierarchieebene werden dabei auf die darunter liegenden Klassen vererbt, so dass man von Merkmalserbung spricht. [NOH]

So lässt sich z.B. das Themengebiet „*Computer*“ weiter unterteilen in „*Software*“ und „*Hardware*“, die „*Software*“ wiederum in „*Anwendersoftware*“ und „*Entwicklungssoftware*“.

Generell kann man vier verschiedene Arten von Klassifikationsschemata unterscheiden, die Universal-Schemata, die fachspezifischen Schemata, die nationalen Schemata sowie selbst entwickelte Schemata.

3.1 Universal Schemata

Die Universal Schemata wurden größtenteils im 19. Jahrhundert für die klassischen Bibliotheken zur Verbesserung ihrer Organisation entwickelt. Sie umfassen daher einen großen Themenbereich und versuchen auf diese Weise das gesamte Wissen der Menschheit mit einzubeziehen. Eine Anforderung von diesem Schema war unter anderem eine universale Anerkennung und die mögliche Verwendung durch alle Menschen zu schaffen. [NOH] Auslöser ihrer Entstehung war die rasante Entwicklung des wissenschaftlichen Sektors. Die in diesem Zusammenhang stehende große Anzahl neu erschienener Bücher verlangte damals eine optimale thematische Einteilung. Einer der bekanntesten Vertreter eines Universal Schema ist unter anderem die DDC.

Weitere Beispiele wären die Library of Congress Classification (LCC), welche unter anderen von der WWW Virtual Library[5] verwendet wird und die Universal Decimal Classification (UDC).

3.1.1 Dewey Decimal Classifikation (DDC)

Die Dewey Decimal Classifikation wurde 1873 von Melvil Dewey konzipiert und 1876 nach dreijähriger Erprobung in der Bibliothek des Amherst College unter dem Titel *„A classification and subject index for cataloguing and arranging books and pamphlets in a libery*“ veröffentlicht und zählt heute zu eines der meist verbreiteten Klassifikationsschemata. Heute wird sie in ca. 90% aller Bibliotheken der USA verwendet. Weltweit wird die DDC in ca. 200 000 Bibliotheken in 135 Ländern angewendet. [OCLC]

Aus der Tabelle 3.1 wird ersichtlich, dass auch die DDC versucht im Sinne eines Universalschemata das gesamte Wissen zu klassifizieren.

[5] URL: http://w3.org/pub/DataSources/bySubject/LibraryOfCongress.html

Es stehen 10 Hauptklassen (siehe Tabelle 3.1) zur Verfügung, welche wiederum in 100 weitere Klassen (Divisions) bzw. 1000 Unterklassen (Sections) unterteilt werden können. Die Notation besteht nur aus arabischen Ziffern, welche als Dezimalbrüche behandelt und auch geordnet werden. Prinzipiell sollte jede einzelne Ziffer für eine weitere Unterteilung bzw. Hierarchiestufe stehen. In der Praxis ist dieses Prinzip jedoch nicht immer einzuhalten.

Es hat sich eingebürgert, die DDC-Zahlen immer mit mindestens drei Ziffern zu notieren und die fehlenden Stellen mit Nullen aufzufüllen. Zur optischen Untergliederung der Notationen wird nach der dritten Stelle ein Punkt eingefügt. Sollte die Notation länger sein als sechs Stellen, werden die folgenden Ziffern in Dreiergruppen notiert, die durch einen Punkt getrennt werden. Diese Zeichen haben keine Bedeutung und sind nicht relevant für die Ordnung der Dokumente. [HOE02]

0	Allgemeines
1	Psychologie und Philosophie
2	Religion, Theologie
3	Sozialwissenschaften
4	Sprache
5	Naturwissenschaften und Mathematik
6	Technik und Medizin
7	Kunst und Kultur
8	Literatur
9	Geschichte und Geographie

Tabelle 3-1: Hauptklassen der DDC [DDC02]

3	Sozialwissenschaften
330	Volkswissenschaften
336	Finanzen, Bank- und Geldwesen
336.700	Geldwesen
336.760	Börsenwesen
336.763	Wertpapiere, Effekten
336.763.300	Obligationen, Schuldverschreibungen
336.763.310	Allgemeines
336.763.311	Verzinsliche Schuldbriefe
336.763.311.100	Langfristig verzinsliche Schuldbriefe

Tabelle 3-2 Beispiel des hierarchischen Aufbaus einer DDC [DDC02]

Tabelle 3.2 zeigt eine beispielhafte Klassifikation mittels DDC. Hierbei ist ersichtlich, wie ausgehend von einem allgemeinen Thema eine immer feinere Unterteilung bis hin zu einem ganz speziellen Themengebiet vorgenommen wird. Die oben sichtbare Zehnerteilung wird solange fortgeführt bis eine angemessene Klasse bzw. Hierarchieebene erreicht ist. Die Unterteilungen bzw. Hierarchiestufen werden durch das Hinzufügen einer Ziffer direkt an der Notation sichtbar.

Verantwortlich für die Änderungen und Anpassung der DDC ist die Decimal Classification Division. Neben der Klassifikation von jährlich mehr als 100.000 Dokumenten werden in unregelmäßigen Abständen neue Editionen der DDC herausgegeben. Die Dewey Datenbank wird von der OCLC Forest Press mit Hilfe der Editorial Support Systems (ESS) gewartet. Die DDC besitzt für jede Klasse eine Menge an Schlüsselwörtern, welche diese Klasse identifizierend beschreiben soll. [DDC02][NOH]

Die DDC weist, wie auch viele andere Universalschemata, einige Nachteile auf. So ist sie beispielsweise auf Grund ihrer Entstehung sehr amerikanisch geprägt und hiermit nur bedingt auf viele Bereiche des europäischen Lebens anwendbar. Hier lässt sich z.B. die Klasse 34, das Recht nennen. Es gibt einen essentiellen Unterschied zwischen dem amerikanischen und dem europäischen Rechtsystem. So kann das EU-Recht mit den gegebenen Strukturen nicht korrekt beschrieben werden. Des Weiteren ist die DDC thematisch orientiert, das Recht ist jedoch in erster Linie ein regionalorientiertes Fach. Aus diesem Grund wurde auch seit der Revision 19 eine so genannte „*Option B*“ eingeführt, die zuerst eine regionale und erst in weiterer Folge eine thematische Gliederung aufweist. Seit dieser Revision steht die Klasse 344 nicht mehr für „*Sozialrecht*“, sondern für „*Europa*“. Dieses hat aber wiederum zur Folge, dass ohne Kenntnis der verwendeten Option eine korrekte Anwendung nicht mehr möglich ist. [KNU99]

Ein Vorteil der DDC ist die hohe Klassenintegrität. Es kommt kaum zu Überschneidungen von einzelnen Themengebieten. Die Themendefinition in der Datenbank ist überall eindeutig, obwohl es mehr als 30.000 definierte Klassen gibt. [DDC02]

3.2 Fachspezifische Schemata

Fachspezifische Schemata wurden für eine bestimmte Gruppe von Benutzern oder zur Organisation eines speziellen Themengebietes entwickelt. Die zugrunde liegende Struktur bzw. Terminologie ist stark mit dem jeweiligen Themengebiet verbunden. Sie werden meist in jenen Bereichen eingesetzt, welche im Gegensatz zu Universalklassifikationen eine feinere Strukturierung benötigen. [HEB00] Ein bekannter Vertreter von diesem Schema in der Praxis ist die ACM CSS.

3.2.1 ACM Computing Classification System (ACM CCS)

Die ACM (Association for Computing Machinery) CSS ist, wie erwähnt, ein Vertreter eines fachspezifischen Schema. Sie ist auf das Fachgebiet der Computerliteratur und allen mit Computern im Zusammenhang stehenden Ressourcen spezialisiert. Die ACM CSS besitzt 11 Hauptklassen (siehe Tabelle 3.3), welche aber selten direkt zur Bewertung von Dokumenten herangezogen werden, da sie meist zu allgemein gehalten sind. Die Notation der Hauptklassen werden durch einen großen Buchstaben (A – K) dargestellt. Die zweite und die dritte Ebene werden wiederum durch arabische Ziffern gekennzeichnet. Alle Ebenen werden mit einem Punkt abgeschlossen. „*D.4.0*“ steht beispielsweise für „*Betriebssysteme allgemein*“. [ACM98]

Die komplette Struktur des ACM CSS besteht aus insgesamt 4 Klassifizierungsebenen, wobei die ersten drei Ebenen, wie schon beschrieben, fest durch einen Buchstaben oder eine Zahl definiert sind. Die vierte Ebene ist eine freie Beschreibung ohne Signatur. Entsprechend dieser Ebene kann der Inhalt eines Dokuments genau eingeordnet werden.

A	General Literature
B	Hardware
C	Computer Systems Organisation
D	Software
E	Data
F	Theory of Computation
G	Mathematics of Computation
H	Information System
I	Computing Methologies

J	Computer Applications
K	Computing Milieux

Tabelle 3-3 Hauptklassen der ACM [ACM98]

Die folgende Tabelle zeigt ein Beispiel einer Kategorisierung anhand der Klasse „D." nach der ACM CSS.

D.	Software und Software Engineering
D.4.	Betriebssysteme
D.4.1.	Prozessmanagement
D.4.1.b	Deadlocks

Tabelle 3-4 Bsp. einer Themenbeschreibung nach ACM [ACM98]

Entwickelt wurde das Schema in den 60er Jahren und erstmals 1964 veröffentlicht. Im Laufe der Zeit wurde das Schema zweimal verändert, wobei das Grundkonzept jedoch immer gleich geblieben ist. Die Klassifizierung wurde jedes Mal überarbeitet. Die derzeitige Version stammt aus dem Jahr 1998. [ACM98][HOE02]

Weitere Beispiele für fachspezifische Schemata sind die International Classification for Standards (ICS) und die FkDigBib, eine Fachklassifikation für eine digitale Bibliothek.

3.3 Nationale Schemata

Nationale Schemata sind zwar bezüglich der Themen allgemein gehalten, jedoch meistens nur für ein Land oder eine geographische Region entwickelt worden. Zu ihnen zählen unter anderen die BC (Nederlandse Basisclassificatie), SAB[6] (Sveriges Allmänna Biblioteksförening) und die GHBS (Systematik der Gesamthochschulbibliotheken in Nordrhein-Westfalen)

3.4 Weitere Schemata

Neben den beschriebenen Klassifikationsschemata gibt es noch weitere Klassifikationsmethoden, die meiner Meinung nach im Folgenden noch kurz erwähnt werden sollten. Hierbei handelt es sich um die Facettenklassifikation und um die

[6] http://www.molnda.se/bibl/subject.htm

Semantischen Netze. Neben diesen gibt es noch eine Reihe anderer Methoden zur Klassifikation, auf welche ich aber auf Grund meiner Aufgabenstellung nicht näher eingehen möchte.

3.4.1 Facettenklassifikation

Eine Weiterentwicklung der Klassifikationsschemata in Bezug auf Flexibilität stellen die so genannten Facettenklassifikationen dar. Die Grundkategorien sind vergleichbar mit den Hauptklassen der oben beschriebenen Klassifikationsschemata. Sie entsprechen dabei den gleichen Gesichtspunkten, unter denen Dokumente betrachtet werden können. Für jede dieser Grundkategorien werden dann als mögliche Werte so genannte Facetten angegeben. Die einzelnen Facetten können wiederum hierarchisch strukturiert werden. Die folgende Tabelle 1.3 zeigt als Beispiel die Klassifikation von Obstbäumen. [FEB00]

Facette A Fruchtart	**Facette B Stammart**	**Facette C Erntezeit**
A1 Apfel	B1 hochstämmig	C1 früh
A2 Birne	B2 halbstämmig	C2 mittel
A3 Kirsche	B3 niederstämmig	C3 spät
A4 Pfirsich		
A5 Pflaume		

Tabelle 3-5 Bsp. einer Facettenklassifikation von Obstbäumen [FUH02]

Bei „*A1B3C1*“ handelt es sich demnach laut Tabelle 3.5 um einen niederstämmigen Frühapfelbaum.

Die Facettenklassifikation kann als mehrdimensionales System angesehen werden, bei dem in jeder Dimension eine von den anderen Dimensionen mehr oder weniger unabhängige Beschreibung durchgeführt wird. [FEB00]

Aufgebaut werden die Facetten, indem versucht wird, aus ausgewählten Dokumenten Einfachklassen zu extrahieren und zu gruppieren, bzw. zu Facetten zusammen zu fassen. Wird ein neues Dokument klassifiziert, werden die behandelten Themen analysiert und die entsprechenden Einfachklassen zusammengefügt. Durch Synthese der Einfachklassen werden nach diesem Ablauf

neue Klassen gebildet. Auf Grund dieses Vorgangs nennt man die Facettenklassifikation auch synonym „*analytisch-synthetische Klassifikation*“. [BUC89]

3.4.2 Semantische Netze

Semantische Netze bestehen im Wesentlichen aus Knoten und Kanten. Knoten stellen Dokumente, Begriffe oder Konzepte dar. Kanten symbolisieren die Relationen zueinander. Zu einem bekannten Vertreter von semantischen Netzen gehört beispielsweise „*WorldNet*“[7]. Hierbei handelt es sich um ein System von Synonymbeziehungen. Der Vorteil der semantischen Netze gegenüber anderen hierarchischen Klassifikationen lässt die Möglichkeit entstehen, verschiedene Arten der Verfeinerung der Hierarchie zu berücksichtigen. Einige Klassifikationsschemata versuchen dieses ansatzweise mit der Implementierung von „Anhängezahlen bzw. -buchstaben“ wie in dem Beispiel der ACM CSS. Hierdurch wird das gesamte System zwar mächtiger, allerdings auch wesentlich komplizierter. Semantische Netze ermöglichen durch die verschiedenen Relationen auch verschiedene Sichtweisen, wohingegen bei den zuvor vorgestellten Klassifikationsschemata die Art der Spezialisierung für die gesamte Hierarchie vorgegeben ist. [FEB00]

Ein Beispiel soll anhand der Einteilung verschiedener Bücher, nach dem Buchtyp, sowie nach dem Autor verdeutlicht werden. Durch die Relation „*Der Telekinet*“ auf „*H.Maurer*“ und „*H.Maurer*“ auf „*Science-Fiction*“ ergibt sich, dass das Buch mit dem Titel „*Der Telekinet*“ aus dem Bereich „*Science Fiction*“ stammt.
[HOF02][KNO02]

[7] URL: http://www.cogsci.princeton.edu/~wn

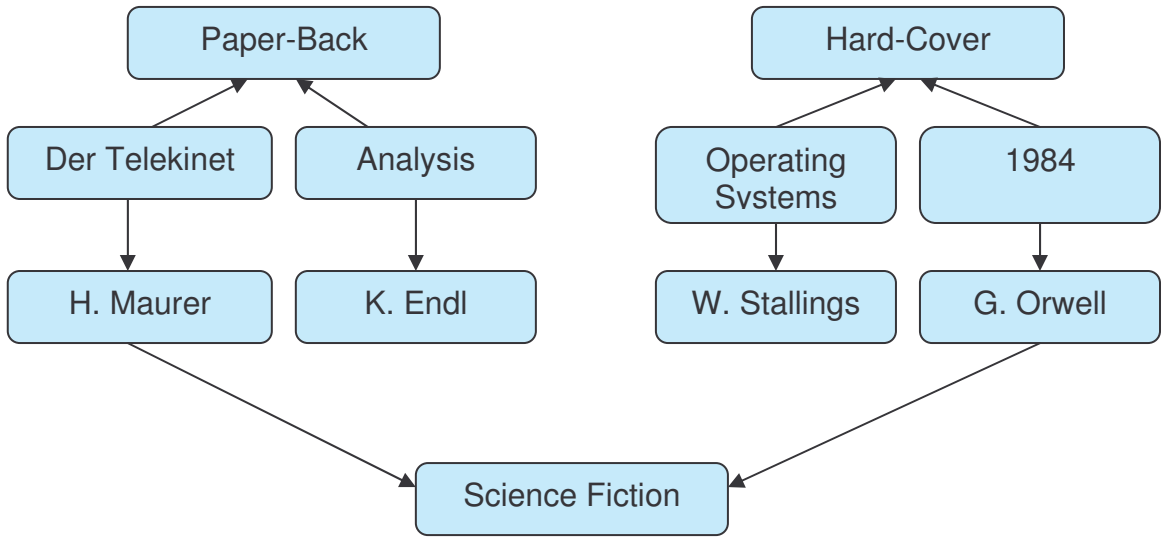

Abbildung 3-1 Einfaches Beispiel für ein semantisches Netz [HOF02]

3.4.3 Topic Maps

Eine weitere Möglichkeit sind die Topic Maps. Hier handelt es sich um eine Weiterentwicklung der semantischen Netze. Sie bestehen aus „Topics“ (den Themen bzw. Knoten) und den „*Topics Occurances*“ (den statischen Dokumentzuordnungen zu einem jeweiligen Thema). Verweise (die Assoziationen bzw. Kanten) verbinden dann diese Themen mit ihren Dokumenten und machen ihre Beziehung deutlich. Laut dem ISO Standard 13250:2000 definieren Topic Maps ein Modell und eine Architektur für ein strukturiertes Netzwerk von Hyperlinks, welches mit entsprechenden Informationsobjekten (den Dokumenten) verknüpft werden kann. Daraus entsteht aus den ehemals unstrukturierten Dokumenten eine Art „virtuelle“ Struktur. [KNO02]

4 Automatische Klassifikation

Bei der Zuweisung einer Informationseinheit zu einer Struktur kann man zwischen zwei Arten unterscheiden. Besteht die Struktur schon vor der Einfügung des Informationsobjektes in diese Struktur, so handelt es sich um eine Klassifikation. Entsteht diese Struktur erst nach einer neuen Ausgangsbasis durch die Einteilung des Objektes, so handelt es sich um Clustering.

Im traditionellen Sinn bedeutet Klassifikation die Einteilung in bestimmte Kategorien oder Klassen. Es werden Zugehörigkeitsentscheidungen getroffen, die auf ermittelten Daten basieren, nach denen die Informationseinheit bzw. das Dokument eingeteilt wird. Ein Klassifikationsprozess ist somit die wiederholte Anwendung dieser Entscheidungsfindung in neuen Situationen. [MST94]

Im letzten Kapitel wurde ein Überblick über die Klassifikationsmöglichkeiten gegeben. In diesem Kapitel soll jetzt eine Übersicht gegeben werden, wie man Dokumente in solche Schemata automatisch eingliedern kann.

4.1 Klassifikationsverfahren

Die folgende Abbildung 4.1 gibt einen Überblick über eine mögliche Einteilung der Klassifikationsverfahren.

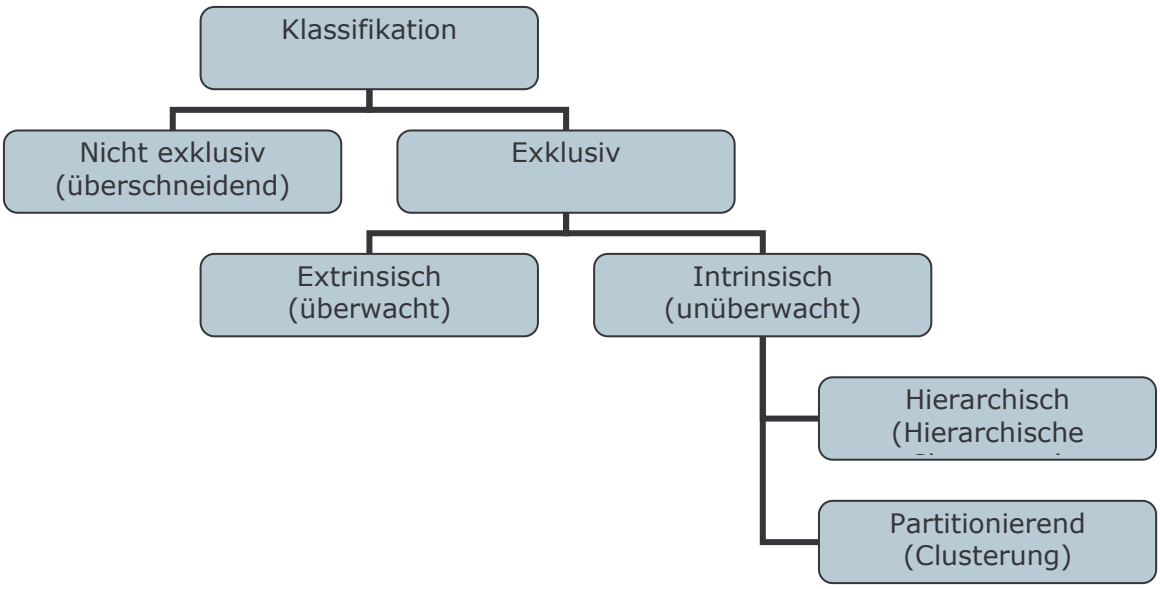

Abbildung 4-1 Einteilung der Klassifikationsverfahren [RH98]

Unter nicht exklusiven Klassifikationen (Polyhierarchien) sind diejenigen zu verstehen, bei denen die Dokumente mehreren Klassen zugeordnet werden können. Deshalb wird sie auch als überlappende Klassifikation bezeichnet. Bei exklusiven Klassifikationen (Monohierarchien) hingegen wird das Dokument genau einer Klasse zugeordnet. Wenn im Allgemeinen von Klassifikation gesprochen wird, ist in der Regel von der extrinsischen bzw. der überwachten Klassifikation die Rede. Hier wird, wie schon oben erwähnt, in eine a priori festegelegte Klassenhierarchie hinein klassifiziert. Bei der intrinsischen Klassifikation handelt es sich hingegen um post priori festgelegte Klassen, dem schon erwähnten Clustering. Beim hierarchischen Clustering entstehen zusätzlich Subklassen, wodurch eine Hierarchieebene gebildet wird. Im Gegensatz zu dem partitionierendem Clustering, wo nur eine Ebene existiert. [RH98]

Ein großer Vorteil der extrinsischen Klassifikation ist die Verwendung bereits bekannter Klassen. Das Zurückgreifen auf bekannte Klassen erleichtert die Entwicklung eines effektiven Algorithmus erheblich. Ein Nachteil der extrinsischen Klassifikation hingegen ist, dass ein möglichst großes Trainingsset zusammen gesucht werden muss, welches mit hohem Zeitaufwand bzw. Kosten verbunden ist. Beim Clustering ist das Vorhandensein eines großen Trainingssets nicht notwendig. Der Grund dafür ist, dass hier keine Vorgaben gemacht werden müssen, weil der Algorithmus für die Klassenbildung verantwortlich ist. Um jedoch sicher zu gehen,

dass passende Klassen gefunden wurden, bedarf es auch hier einer anschließenden Kontrolle. Diese ist ebenfalls mit einem erheblichen Arbeitsaufwand verbunden.

Die Basis aller Klassifikationen ist eine Analyse der Dokumente, die es im Folgenden gilt in eine entsprechende Klasse einzuordnen. Die ersten Schritte auf diesem Gebiet wurden schon sehr früh entwickelt. So schrieb schon 1958 H. P. Luhn:

„Die Häufigkeit eines Wortes in einem Artikel ist eine brauchbare Maßeinheit für die Signifikanz des Wortes. Weiter ist die relative Position von Wörtern mit bestimmter Signifikanz innerhalb eines Satzes eine Maßeinheit um die Signifikanz des Satzes zu bestimmen.“ [LUH58]

Es gibt grundsätzlich 2 Phasen bei einer Klassifikation, die Lernphase und die Anwendungsphase. Jede Phase besteht aus mehreren Schritten. Bei der Lernphase wird anhand von einem Trainingsset ein Modell erstellt. Das zugrunde liegende System wird auf dieses Trainingsset abgestimmt bzw. angepasst. In der Anwendungsphase werden anschließend anhand des erstellten Modells und des entwickelten Klassifikators neue Dokumente klassifiziert. [KLI98]

4.2 Dokumentanalyse

Die Textanalyse der Dokumente kann im Wesentlichen auf verschiedene Arten passieren. In dem Fall, wenn es um die Erkennung von verschiedenen Phänomenen der Sprache geht, wie beispielsweise unterschiedliche Wortformen, Suffixe oder Phrasen, so handelt es sich um eine linguistische Analyse. Zählt man z.B. die relevanten Attribute bzw. Wörter im Verhältnis zu der Gesamtanzahl im Dokument, handelt es sich um die statistische Analyse. Außerdem gibt es noch die begriffsorientierte Analyse. [NOH]

4.2.1 Linguistische Analyse

Die linguistische Analyse basiert auf morphologischen, syntaktischen, sowie semantischen Verfahren. Sie lässt sich durch die Techniken, welche sie verwendet, in zwei Verfahren einteilen, das wörterbuchbasierte und das regelbasierte Verfahren.

Die linguistische Analyse baut auf der Wortebene auf, beispielsweise durch Stoppwortelimination oder Wortstammbildung, welche beide in den folgenden Absätzen noch näher erläutert werden. Eine weitere Möglichkeit der morphologischen Analyse ist die Flexionsformengenerierung, bei der ein Wörterbuch angelegt wird, indem alle grammatikalisch möglichen Formen aller Wörter eingetragen werden. Ziel hiervon soll sein, schnell gleiche Grundformen zu erkennen. (z.B. dass aus den Verben „schrieb“ und „geschrieben“ die Grundform „schreiben“ abgeleitet werden kann)

Weiter gibt es die Kompositärzerlegung, bei der Mehrwortbegriffe in ihre Wortgrundformen zerlegt werden. Bei der Derivation werden die Wörter bzw. Wortklassen mit derselben Grundform zusammengefasst. Ein Beispiel wäre z.B. „Klassifikation“ und „klassifizieren“. [FUH02]

Die syntaktische Analyse baut auf der Satzebene auf und untersucht den Satzbau bzw. seine einzelnen Komponenten (Subjekt, Prädikate, Adjektive, etc.). Bei der semantischen Analyse wird letztendlich das gesamte Dokument mit in die Betrachtung einbezogen. [FUH02]

4.2.2 Statistische Analyse

Die statistische Analyse setzt sich größtenteils mit der Vorkommenshäufigkeit von Wörtern bzw. deren statistischer Verteilung auseinander. Die Bedeutung der einzelnen Wörter spielt hierbei keine Rolle. Denn das Problem der Klassifikation von Dokumenten kann nur so, vor allem mathematisch, einigermaßen in den Griff bekommen werden. Zum Aufbau eines solchen brauchbaren Modells werden in der Lernphase im Wesentlichen 5 Schritte benötigt: Die Textnormierung, die Termgenerierung, die Attributauswahl, die Attributgewichtung und die Trainingsphase. [KLI98]

4.2.2.1 Die Textnormalisierung

Bei der Dokumentanalyse kommt es ausschließlich auf den Inhalt des Dokumentes an. Aus diesem Grund müssen bei der Textnormalisierung alle unerwünschten Zeichen bzw. Zeichenketten (Token) entfernt werden. Um welche Zeichen es sich hierbei handelt, wird von dem jeweiligen Dokumenttyp bestimmt. So werden

beispielsweise bei HTML oder XML neben Satz- und Sonderzeichen auch die entsprechenden Formatierungs-Tags entfernt. [KLI98]

4.2.2.2 Die Termgenerierung

Hier kommt es in erster Linie darauf an, wie das Dokument im späteren Modell repräsentiert werden soll und in wie weit die enthaltenen Wörter zur Weiterverarbeitung genutzt werden sollen. Bei den meisten Verfahren spielt jedoch die Reihenfolge des Auftretens der Wörter keine Rolle. Anstatt einer Sequenz von Wörtern, wird nur eine so genannte Multimenge von Wörtern[8] betrachtet. Natürlich gehen so Informationen des Dokumentes verloren, allerdings ist dieser Weg ein guter Kompromiss zwischen tatsächlichem Inhalt und der mathematisch, formalen Handhabbarkeit. [KLI98]

Eine weitere Möglichkeit das Dokument intern darzustellen sind n-Gramme. Hierbei werden nicht die einzelnen Wörter betrachtet, sondern immer nur Zeichenketten der Länge n. So entstehen beispielsweise aus dem Wort „*Wort*“ mittels Trigramme[9] die Zeichenketten „*_Wo*“, „*ort*“, „*rt_*“. Zu den Vorteilen dieses Verfahrens zählen unter anderen die Robustheit gegenüber Rechtschreibfehlern sowie vor allem die sprachunabhängige Erkennung von Wortbestandteilen, wodurch eine sprachübergreifende Einsatzmöglichkeit ermöglicht wird. [KLI98]

4.2.2.3 Die Attributauswahl

Bei der Attributauswahl geht es um die Bestimmung des Informationsgehaltes eines einzelnen Wortes im Bezug auf das Dokument. Dazu wird der Text in einzelne Wörter bzw. Attribute (Token) zerlegt.

Jedes einzelne Token wird dann mittels Stoppwortelimination bearbeitet. Alle Wörter bzw. Token, die das spätere Ergebnis störend beeinflussen könnten, werden entfernt. Für diese Vorgehensweise gibt es wiederum verschiedene Ansätze. Es können beispielsweise alle Wörter entfernt werden, deren Wortlänge bzw. Buchstabenanzahl eine vorgegebene Anzahl nicht überschreiten. Bei einer Anzahl von 3 wären das beispielsweise u.a. die Bindewörter „*und*“, „*der*“, „*die*“, „*das*“ etc.

[8] Engl.: bag of words
[9] Trigramme sind n-Gramme der Länge 3

Es können aber auch alle Wörter entfernt werden, die in einer Stoppwortliste eingetragen sind. Hieraus würde jedoch automatisch auch eine Sprachabhängigkeit des Systems resultieren. Aus diesem Grund können Stoppwortlisten auch automatisch, beispielsweise anhand der Wortfrequenzanalyse oder des Rauschmaß, erstellt werden. Bei der zu erst genannten Methode werden alle Wörter anhand ihres Vorkommens bzw. ihrer Auftretenshäufigkeit sortiert. Wichtig sind nun alle jene Wörter, die eine Klasse am eindeutigsten beschreiben können. Es wird hier bei den Wörtern davon ausgegangen, dass bei einer sehr niedrigen bzw. einer sehr hohen Auftretenshäufigkeit, oder bei Vorkommen in vielen Klassen gleichzeitig, sie keinen wichtigen Informationsgehalt für die Klassifizierung des Dokumentes enthalten und sie demnach entfernt werden können. Es werden somit diejenigen Wörter entfernt, deren bestimmte Häufigkeit einen Schwellwert über- oder unterschreitet.

Bei der Erstellung mittels Rauschmaß wird geprüft, ob das Auftreten eines Wortes (Token) über alle Dokumente relativ gleich verteilt ist. Ist dieses der Fall, so ist die enthaltene Information bezüglich der Klassifikation sehr gering und das Wort kann somit entfernt werden. Ist das Vorkommen des Wortes hingegen selten, kann man davon ausgehen, dass das entsprechende Wort einen sehr hohen Informationsgehalt hat. [WEI01]

In einem weiteren Schritt, in dem man weitere Verbesserungen der Ergebnisse erzielen kann, ist die Stammformreduktion. Unter der Stammformreduktion versteht man, dass alle Endungen der Wörter bzw. Token entfernt werden. Das hat zur Folge, dass es so meist zu einem nicht in der Realität existierenden Wort kommt. Auf diesem Wege werden beispielsweise die Wörter „*computer*“, „*computerization*“, „*compute*“ und „*computation*“ auf ihre Stammform „*comput*“ reduziert. [FUH02]

Ein Problem bei der Attributauswahl stellen mehrsprachige Dokumente dar. Meist reichen schon einzelne englische Fachausdrücke in einem deutsch verfassten Dokument aus, um negative Auswirkungen auf das mit Hilfe dieses betroffenen Dokuments zu entwickelnde Modell haben. Diese Wörter werden bei der Ermittlung der Worthäufigkeit mit berücksichtigt und verfälschen so die Ergebnisse. Eine mögliche Lösung wäre die Benutzung der n-Gramme zur Termgenerierung.

4.2.2.4 Die Attributgewichtung

Nachdem die Auswahl der Attribute getroffen ist, können die Attribute ein weiteres Mal gewichtet werden. Entscheidend hierbei ist wie oft die Attribute auftreten. Die Verfahren der Attributgewichtung sind sehr vielfältig.
Ein Verfahren ist das TFIDF, die „Term Frequency Inverse Document Frequency“ [KLI98].

Bei dem TFIDF Verfahren wird die Gewichtung anhand der Termfrequenz und der Dokumentfrequenz berechnet. Die Termfrequenz TF (w,d) gibt an, wie oft ein Wort w in einem Dokument d auftritt. Die Dokumentfrequenz DF (w) hingegen gibt die Anzahl derjenigen Dokumente an, in denen das Wort w mindestens einmal auftritt. Dieses Verfahren basiert auf der Annahme, dass nur die Wörter aussagekräftig sind, welche einerseits eine hohe Termfrequenz aufweisen sowie zusätzlich eine geringe Dokumentfrequenz besitzen. Dieses sind somit diejenigen Worte, die in einem Dokument selbst häufig auftreten, gleichzeitig aber nur in wenigen Dokumenten insgesamt anzufinden sind. Die Dokumentfrequenz wird mit der nach Klinkenberg beschriebenen IDF, der inversen Dokumentfrequenz berechnet. [KLI98]

$$IDF(w) = \frac{\log |D|}{DF(w)}$$

Formel 4-1: IDF

D steht hier für die gesamte Trainingsmenge und |D| ist somit folglich die Anzahl aller Trainingsdokumente. w steht für ein Attribut bzw. Wort des Dokumentes. Das TFIDF Gewicht eines Wortes w im Dokument d ergibt sich somit wie folgt [Klinkenberg 1998].

$$TFIDF(w,d) = TF(w,d) * IDF(w)$$

Formel 4-2: TFIDT

Ein weiterer Aspekt, den man sich bei der Klassifikation von Programm spezifischen Dokumenten zu Nutze machen kann, ist die Attributsgewichtung der Wörter entsprechend ihrer Position im Dokument. Die Gewichtung der Wörter ist somit abhängig davon, ob die Wörter beispielsweise im Titel, in der Überschrift oder im Text auftreten. Wörter, die dann in der Überschrift vorkommen, können beispielsweise so ein höheres Gewicht haben, als jene, die nur im normalen Text auftreten. Um diese Positionen genau zu bestimmen, helfen die jeweiligen Formatierungs-Zeichen bzw. –Tags. So stehen z.B. in einem HTML-Dokument die Überschrift in den Tags <H1> bis <H6> und der normale Text im <body> Tag.

Des Weiteren können die Attribute bzw. Wörter auch anhand von Metainformationen gewichtet werden, wenn diese vorhanden sind. Sind sie vorhanden, so kann davon ausgegangen werden, dass diese Worte für die Informationsbeschreibung des Dokumentes sehr wichtig sind und sie somit ein höheres Gewicht bekommen müssen als diejenigen, die hier nicht vorkommen.
Außerdem besteht die Möglichkeit, dass die Gewichtung von Attributen manuell vorgenommen werden kann. [FEB00]

4.2.2.5 Die Trainingsphase

In dieser Phase wird nun mit Hilfe der, aus dem eben beschriebenen Verfahren, gefundenen und bearbeiteten Attributen und eines darauf angewendeten Algorithmus, ein Modell erstellt. Ziel ist jetzt das Trainieren eines Klassifikators für eine bekannte Anzahl an Klassen eines Trainingsdokumentessatzes. Auf diese Weise ist später (nach den Trainingsdurchläufen) auch dann auch das Klassifizieren von unbekannten Dokumenten möglich. [KLI98]

4.2.3 Begriffsorientierte Verfahren

Ein Problem der eben beschriebenen linguistischen und statistischen Verfahren ist, dass beide nicht die Bedeutung einzelner Wörter verstehen können. So handelt es sich bei den Worten „Klavier“ und „Piano“ um je einen Term, welche aber von der sprachlichen Repräsentation eine gleiche Bedeutung haben. Einige linguistische Verfahren können zwar beispielsweise durch die Stammformbildung die Worte „Klavier“ und „Klaviere“ auf ein Wort zurückführen, jedoch auf eine sprachunabhänige, auf die Bedeutung aufbauende Analyse geben sie nicht.

Begriffsorientierte Verfahren abstrahieren nun die Bedeutung der vorgefundenen Worte und versuchen so den Inhalt des Dokumentes zu erfassen. Die ermittelte Bedeutung wird später mittels eines festen Vokabulars (z.B. eines Thesaurus oder Wörterbuches) repräsentiert. Auf diesem Wege kommt man der tatsächlichen Bedeutung des Dokumentes, wie sie durch eine manuelle Analyse gegeben wäre, zwar einen Schritt näher, jedoch kann man noch keinesfalls vom wirklichen Verständnis bzw. Bedeutung des Dokumentes reden. Die heutige Sprachwissenschaft geht von der Annahme aus, dass die Bedeutung eines Wortes nur über den Kontext seines jeweiligen Gebrauches erschlossen werden kann. Für optimale Analyseverfahren wäre somit die Berücksichtigung des Kontextes unumgänglich. Aber weder die linguistischen, noch die statistischen Verfahren kommen dieser Forderung entgegen. Die Begrifforientierten Verfahren gehen deshalb in eine wissensbasierte Richtung, wo beispielsweise mit Modellen aus der künstlichen Intelligenz durch Einbeziehung des Weltwissens und Wissensakquisition versucht wird, dieser Forderung nachzukommen. Hier ist allerdings der Implementierungsaufwand sehr erheblich, so dass schon die Realisierung kleinerer Systeme extrem aufwendig und kostenintensiv ist. [NOH]

4.3 Klassifikationsalgorithmen

Aufgrund des hohen Implementierungsaufwandes auf Seiten der begriffsorientierten und linguistischen Verfahren, sollen im Folgenden einige wichtige Algorithmen vorgestellt werden, welche deshalb vor allem bei der statistischen Dokumentauswertung zum Einsatz kommen. Vor jeder Klassifikation bedarf es der Entscheidung, ob ein Dokument einem Thema oder mehreren Themen zugeordnet werden kann. Ein Beispiel für ein binär separierbares[10] Problem wäre der medizinische Bereich. Basierend auf Untersuchungsergebnissen bzw. Krankheitsmerkmalen soll nun bestimmt werden, ob eine Krankheit vorliegt oder nicht. Algorithmen wie der KNN hingegen können einem Dokument auch mehrere Themen zuweisen.

[10] Es gibt nur zwei Entscheidungsmöglichkeiten, richtig oder falsch.

4.3.1 Das Vektorraummodell

Viele Klassifikatoren verwenden für die Entscheidungsfindung bzw. Klassifikation das Prinzip des Vektorraum-Modells (siehe Abbildung 4.1). Es gehört wohl zu den bekanntesten Modellen in der Forschung des Information Retrieval. Entwickelt wurde es von G. Salton und seine formale Definition lautet wie folgt:

Definition: Sei $T = \{t_1, \ldots, t_n\}$ eine endliche Menge von Termen und $D = \{d_1, \ldots, d_m\}$ eine Menge von Dokumenten. Für jedes Dokument $d_i \in D$ sei zu jedem Term $t_k \in T$ ein Gewicht $w_{i,k} \in R$ gegeben. Die Gewichte des Dokumentes D_i lassen sich zu einem Vektor $w = (w_{i,1}, \ldots, w_{i,n}) \in R^n$ zusammenfassen. Dieser Vektor beschreibt das Dokument im Vektorraummodell: er ist seine Repräsentation und wird Dokumentvektor genannt. [FEB00]

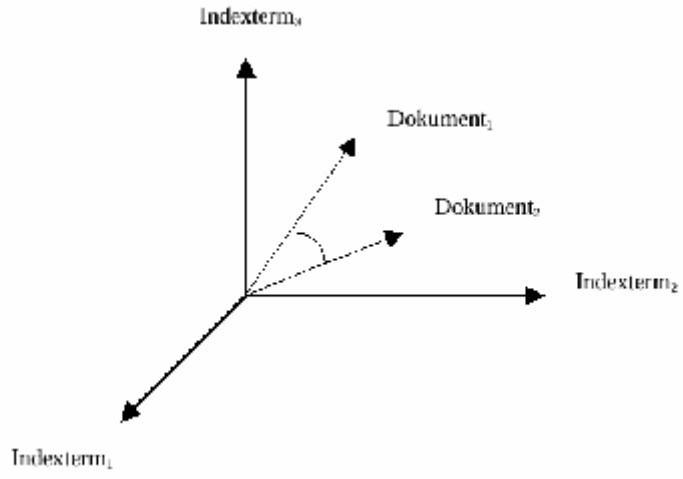

Abbildung 4-2 Vektorraummodell

Im Klassifizierungsschritt werden dann auch die neu zu klassifizierende Dokumente durch Vektoren $q \in R^n$ dargestellt. Wie bei der Repräsentation der Trainingsdokumente werden diese durch eine Menge gewichteter Terme dargestellt. Darauf wird mittels einer Vergleichsfunktion $s: R^n \times R^n \rightarrow R$ definiert, mit der jedem Paar aus zwei Vektoren $x, y \in R^n$ ein reeller Ähnlichkeitswert $s(x, y)$ zugewiesen wird. Die formalen Definitionen sagen also aus, dass Trainingsdokumente bzw. die durch sie definierten Klassen über Vektoren repräsentiert werden. Gleiches gilt für das neu zu klassifizierenden Dokument. Dann wird beispielsweise mittels des Skalarproduktes zweier Vektoren die Ähnlichkeit der beiden ausgerechnet und

verglichen. Dort, wo die größte Ähnlichkeit zwischen neuem Dokument und bestehender Klasse vorliegt, wird das Dokument hineinklassifiziert. Zur Berechnung der Ähnlichkeit kann man wie schon erwähnt das Skalarprodukt oder das Kosinusmaß benutzen. [FEB00]

$$Skalar = x * y = \sum_{i=1}^{n} x_i * y_i$$

Formel 4-3: Skalarprodukt [PAP97]

$$\cos(x, y) = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} * \sqrt{\sum_{i=1}^{n} y_i^2}}$$

Formel 4-4: Kosinusmaß [PAP97]

Bei dem Ergebnis mittels Skalarprodukt entsteht eine große Abhängigkeit in der Größe und Anzahl der einzelnen Werte in den Vektoren bzw. der Länge der Dokumente. Würde man den Dokumentinhalt unabhängig von der Gewichtung der einzelnen Attribute verdoppeln, so würde sich aufgrund der Linearität auch die Ähnlichkeit der Dokumente verdoppeln, obwohl kein zusätzlicher Informationsgewinn existieren würde. Trotzdem kann das Skalarprodukt überall eingesetzt werden, wo die Länge eines Dokumentes bekannt ist. Das Kosiniusmaß hingegen ist unabhängig von der Länge der Dokumente. Das Maß gibt lediglich die Richtung der Vektoren an. Die zwei Vektoren, welche am ehesten in die dieselbe Richtung zeigen, stimmen auch am ehesten miteinander überein. [FEB00]

Außer diesen beiden Möglichkeiten gibt es auch noch weitere Maße wie beispielsweise das Overlap-Maß, das Dice-Maß, das Jaccard-Maß oder die euklidische Distanz. Für detaillierte Beschreibungen dieser Maße möchte ich auf [FEB00] verweisen.

4.3.2 Der Rocchio Algorithmus

Auch bei diesem Algorithmus werden die Dokumente nach dem Vektormodell als Vektoren repräsentiert. Die Attributgewichtung erfolgt nach dem TFIDF Verfahren und wird anschließend in der Regel auf die euklidische Länge 1 normiert. Aus diesem Grund ist dieser Algorithmus auch unter dem Namen „*TFIDF-Algorithmus*“ bekannt. Am Ende des Algorithmus wird dann letztlich die Ähnlichkeit der Vektoren wieder über das Kosinus-Maß ermittelt. [KLI98]

4.3.3 Der Naive-Bayes Algorithmus

Dieser Algorithmus zählt mit zu den bekanntesten innerhalb der statistischen Verfahren. Ausgehend von der Fragestellung, mit welcher Wahrscheinlichkeit ein Dokument d in eine Klasse C_j zugeordnet wird“, wird der folgende Ansatz aufgebaut. Mathematisch kommt man mit Hilfe der bedingten Wahrscheinlichkeit hier zu einer Antwort der Frage. Diese gibt die Wahrscheinlichkeit an, dass ein Ereignis A auftritt, wenn ein Ergebnis B aufgetreten ist. Die Gleichung hierfür lautet wie folgt:

$$P(C_j / d) = \frac{P(C_j \cap d)}{P(d)}$$

Formel 4-5: Bedingte Wahrscheinlichkeit

$$\textit{Wobei } P(C_j \cap d) = P(d / C_j) * P(C_j)$$

In der Formel wird *P (Cj/d)* als „*a posteriori*“ Wahrscheinlichkeit bezeichnet und definiert die Wahrscheinlichkeit für die Zuweisung in die Klasse C_j unter der Bedingung, dass das Dokument auch klassifiziert wurde. *P (d)* gibt dabei die Wahrscheinlichkeit an, mit der dieses Dokument klassifiziert wird. Da dieser Faktor konstant ist bzw. bleibt, kann er für die weitere Berechnung vernachlässigt werden. P *(d/Cj)* definiert die bedingte Wahrscheinlichkeit unter der sich ein Dokument in der Klasse C_j befindet. Der Faktor *P (Cj)* ist eine „*a priori*“ Wahrscheinlichkeit und bezeichnet hier die vor der eigentlichen Klassifikation bekannte Wahrscheinlichkeit, mit der die Klasse C_j dem Dokument zugeordnet werden kann. Diese Wahrscheinlichkeit ist im Normalfall für alle Klassen gleich groß und stellt somit auch

eine vernachlässigbare Konstante dar, da es am Anfang jeder Klassifikation gleich wahrscheinlich ist, dass eine Klasse einem Dokument zugeordnet wird.

Mit Hilfe der Gleichung lässt sich jetzt das „*bayes'sche Theorem*" definieren, welches als Ausgangpunkt für den Klassifikator des Naive-Bayes-Algorithmus dient. Er berechnet die Wahrscheinlichkeit $P(C_j/d)$, also die, mit der ein Dokument *d* in die Klasse C_j gehört. Dabei wird auch hier das Dokument als Vektor von Attributen bzw. Worten repräsentiert.

$$P(C_j / d) = \frac{P(d / C_j) * P(C_j)}{P(d)}$$

Formel 4-6: Bayes'sches Theorem

Laut der Bayes'schen Regel gehört ein Dokument *d* jetzt in jene Klasse, bei der die höchste Wahrscheinlichkeit ermittelt wird. Daraus folgt:

$$NB(d) = \arg\max_{C_j \in C} P(C_j / d)$$

Durch Umformung nach Bayes'schem Theorem entsteht

$$NB(d) = \arg\max_{C_j \in C} \frac{P(d / C_j) * P(C_j)}{P(d)} \cong \arg\max_{C_j \in C} P(d / C_j) * P(C_j)$$

Der Nenner *P(d)* kann jetzt vernachlässigt werden, da er im Normalfall keinen Einfluss auf das Ergebnis hat.

Der Algorithmus basiert dabei auf der Annahme, dass alle Attribute bzw. Wörter stochastisch vollkommen von einander unabhängig sind. Dieses sieht in der Realität natürlich ganz anders aus. Doch wie des Öfteren in der Klassifikation müssen auch hier Zugeständnisse gemacht werden, um ein komplexes Problem auf eine übersichtliche, verarbeitbare Größe zu minimieren. Mit Hilfe der stochastischen Unabhängigkeit lässt sich nun die Formel folgendermaßen umformen:

$$NB(d) = \arg\max_{C_j \in C} P(C_j) * \prod_{i=1}^{n} P(a_i / C_j)$$

Dabei steht a_i für ein Attribut bzw. Wort aus dem Dokument *d*. Jetzt müssen noch die Wahrscheinlichkeiten bestimmt werden. Eine Formel hierfür ist:

$$P(a_i / C_j) = \frac{n_{ai}(C_j)}{n(C_j)}$$

Wobei $n_{ai}(C_j)$ für die Häufigkeit bzw. das Gewicht des Attributes *i* in der Klasse C_j steht und $n(C_j)$ für die Häufigkeit bzw. das Gewicht aller Attribute in C_j. Hierdurch wird nun auch ein wesentliches Problem sichtbar. Kommt ein Attribut nämlich aus der zu klassifizierenden Instanz in der gerade betrachtenden Klasse nicht vor, so ist der Faktor n_{ai} somit *0* und in weitere Folge dann auch $P(a_i/C_j)$. Damit ergibt sich auch dann eine Wahrscheinlichkeit für diese Klasse $P(C_j/d)$ von *0*. Hilfe schafft hier die Glättung des Ergebnisses mittels so genannten m-estimate, womit sich folgende Gleichung ergibt:

$$P(a_i / C_j) = \frac{n_{ai}(C_j) + m * p}{n(C_j) + m}$$

Die Variable m steht hier für eine Konstante zur Steuerung der Glättung und *p* für eine a priori Wahrscheinlichkeit. Meist wählt man $p = 1/k$, wobei k für die Anzahl der Attribute steht oder man setzt $k=m$. Durch diese Hilfe erweitert man sozusagen die Wahrscheinlichkeitsberechnung p um m virtuelle Attribute. Damit verhindert man die Wahrscheinlichkeitserrechnung von *0*, welche man sonst für die Klassenwahrscheinlichkeit herausbekommen würde, nur aufgrund des Fehlens eines einzelnen Attributes innerhalb dieser Klasse. [KLI98]

4.3.4 Der KKN Algorithmus

Das K-Nearest-Neighbor Verfahren (KNN) verfolgt einen anderen Ansatz. Hier wird das zu klassifizierende Dokument nicht mit den jeweiligen Klassen verglichen, sondern es werden bei diesem Verfahren grundsätzlich alle Dokumente gespeichert und dann Dokumentweise verglichen. Soll ein neues Dokument klassifiziert werden, werden die *k*-ähnlichsten (ein variabel festlegbarer Faktor) Trainingsdokumente über

ein Ähnlichkeitsmaß (in der Regel das Kosinusmaß) bestimmt und dann jener Klasse zugeordnet, aus der die meisten Übereinstimmungen der Trainingsdokumente stammen. Wenn die Klasse C_j diejenige ist, die die Anzahl k_{cj} an Beispielen aus den k-ähnlichsten Dokumenten hat, so wird das neue Dokument mit einer Konfidenz k_{cj}/k in dieses Klasse C_j eingeordnet. Daraus ergibt sich die folgende mathematische Formel für das KNN-Verfahren:

$$KNN(d) = \arg\max_{C_j \in C} k_{C_j}$$

Formel 4-7: KNN-Verfahren

Ein Vorteil dieses Verfahrens ist, dass es gegenüber verrauschten Daten wie beispielsweise Dokumenten mit falscher Klassenzuordnung als Nachbarn recht unempfindlich ist. [KLI98]

Ein Nachteil hingegen ist neben der hohen Anzahl von irrelevanten Attributen auch die gleichmäßige Gewichtung dieser Attribute, da der Klassifikator nicht zwischen den einzelnen Attributen unterscheidet. Dieses Problem ist auch unter dem Namen „*Fluch der Dimensionen*“ bekannt. Ist das Verhältnis zwischen relevanten und irrelevanten Attributen recht groß, so wird das Ergebnis wie an dem folgenden Beispiel verdeutlicht, verfälscht.

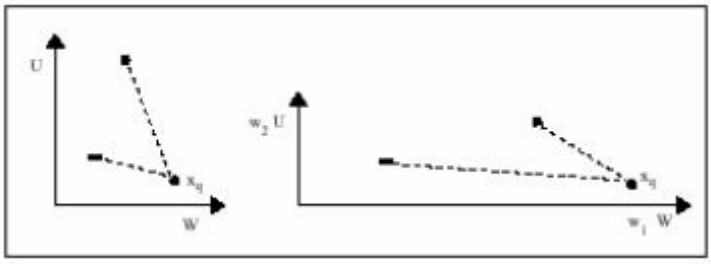

Abbildung 4-3 Fluch der Dimensionen [KLI98]

U und *W* stehen hier für zwei gleich gewichtete Attribute und außerdem gibt es zwei verschiedene Klassen „*Kreis*“ und „*Rechteck*“. Aufgrund der Einfachheit wird als Ähnlichkeitsmaß die euklidische Distanz gewählt. Das neu zu klassifizierende Element x_q wird jetzt aufgrund der kürzeren Distanz im linken Bild fälschlicher Weise

der Klasse „*Rechteck*“ zugeordnet. Erst durch eine Gewichtung der Attribute mittels der Faktoren w_1 und w_2 an den jeweiligen Achsen wird dieses Problem behoben und führt so zu einer korrekten Klassifikation.

4.3.5 Support Vektor Maschinen (SVM)

Ein weiteres Verfahren zur Klassifikation bieten die Support Vector Machines. Sie basieren auf dem Structural-Risk-Minimation-Prinzip der Computational Learning Theory. In ihrer Grundform werden die Support Vektor Maschinen für binäre Klassifikation eingesetzt. Es existiert jedoch inzwischen auch eine Reihe von Abwandlungen, die die Definitionen mehrerer Klassen erlauben. Es werden lineare Grenzfunktionen erlernt, um in einem n-dimensionalen Raum eine Hyperebene zu finden, welche den Raum anhand der Klassenzuordnung in zwei Hälften teilt.
[PKK]

5 Erstellung der Signifikanzvektoren

Nachdem in den letzten Kapiteln die theoretischen Grundlagen bzw. die Möglichkeiten einer automatischen Klassifikation gezeigt wurden, sollen in diesem Kapitel der Weg zu den Signifikanzvektoren, sowie die Probleme aufgezeigt werden, die bei der Berechnung der Signifikanzvektoren der einzelnen Taxonomieknoten beachtet werden müssen.

Wie eingangs dieser Arbeit bereits erwähnt, ist die Basis für die Errechnung der Signifikanzvektoren eine bereits existierende klassifizierte Schlüsselwortliste. Diese besteht aus einer Menge einzelner Schlüsselworte, die den einzelnen Klassen einer Taxonomie zugeordnet wurden. Jede Klasse bzw. jeder Knoten der Taxonomie besitzt dabei eine undefinierte Menge an ihr beschreibenden, ungewichteten Schlüsselworten. Man kann also sagen, dass jede Klasse durch einen Schlüsselwortvektor dargestellt wird.

Eine solch ungewichteter Schlüsselwortvektor kann jedoch nicht für die automatische Klassifizierung genutzt werden, wenn beispielsweise die Klassifizierung wie bei dem HyLOS Projekt anhand der Ähnlichkeitsübereinstimmung von Vektoren realisiert werden soll. (siehe Abschnitt 2.3)

Die Basis für ein solches Verfahren ist, wie in Absatz 4.3.1 bereits beschrieben, eine Gewichtung der einzelnen Schlüsselworte jeder Taxonomieklasse. Denn nur mittels dieser Gewichtungen ist eine Klassifizierung durch Vergleich zwischen dem Wortvektor des Dokuments und den Schlüsselwortvektoren der einzelnen Taxonomieklassen möglich. Bei den Klassen mit der größten Ähnlichkeit wird das neu zu klassifizierende Dokument schließlich automatisch hinein klassifiziert.

Ziel ist es folglich aus den einzelnen Schlüsselwortvektoren Signifikanzvektoren von klassifizierten Dokumenten zu erstellen. Dazu muss die Signifikanz jedes einzelnen Schlüsselwortes der jeweiligen Klasse errechnet werden. Die Datenbasis ist neben den einzelnen Klassen mit ihren beschreibenden Schlüsselworten, klassifizierte Texte, bzw. Dokumente, die in diese Klassen eingeordnet worden sind. Um aus diesen Daten eine aussagekräftige Signifikanz zu errechnen, sind einige Kriterien zu

beachten. Diese werden im folgende Abschnitt näher erläutert, sowie existierende Lösungsmöglichkeiten zur Vermeidung dieser Probleme.

5.1 Probleme bei der Gewichtung von Schlüsselworten

Es gibt einige Kriterien wie die Dokumentgröße bzw. -länge und der mit ihnen abhängigen Häufigkeitsverteilung, die bei der Gewichtung von Worten bzw. bei der Errechnung ihrer Signifikanz beachtet werden müssen. Hinzu kommen durch die Verwendung Digitaler Dokumente Punkte wie die Persistenz, Duplikate oder das zugrunde liegende Dateiformat.

5.1.1 Persistenz

Die Persistenz der Inhalte in digitalen Dokumenten ist nicht gewährleistet. Die Inhalte des Internets verändern sich heutzutage sehr schnell. So verschwinden viele Ressourcen einfach und der User stößt dann auf so genannte dead links[11]. Dieses sind Dokumente, die entweder nicht mehr existieren oder deren Verweisung sich geändert hat.

Zum anderen können die Inhalte dieser Dokumente prinzipiell jederzeit geändert werden, indem dem Dokument neue Inhalte hinzugefügt oder entfernt werden. Sie sind dann unter dem gleichen Verweis weiter aufzufinden.

So zeigt W. Koehler in einer über drei Jahre hinweg durchgeführten Untersuchung, dass in 97% aller untersuchten Dokumente Veränderungen statt fanden. Dieses waren neben überwiegend inhaltlichen Änderungen, auch strukturelle Änderungen. [KOE00]

Ein Problem, was aus diesen Veränderungen resultieren kann, ist eine mögliche falsche thematische Zuordnung zu einer früher zugewiesenen Klasse. Das Dokument kann sich durch die Änderungen im Laufe der Zeit so sehr verändern, dass diese früher richtig zugewiesene Klasse jetzt völlig falsch sein kann. Dieses würde folglich dazu beitragen, dass auch die Signifikanz dieser Klasse hinterlegten Schlüsselworte nun nicht mehr korrekte Ergebnisse besitzen würden. Solche Fälle gilt es somit zu vermeiden.

[11] dead links: tote Verweise

Abhilfe kann hier durch lokale Speicherung der jeweiligen Dokumente geschaffen werden. Das bedeutet, dass bei der erstmaligen Analyse des Dokuments, bezüglich verwendeter Schlüsselworte, eine unabhängige Kopie des originalen Dokuments lokal abgespeichert wird und mit dieser Kopie weitergearbeitet wird. Alle aus dieser Kopie gewonnenen Daten und Klassenzuordnungen bleiben somit korrekt. Findet man jetzt später neu erstellte bzw. geänderte Auflagen dieses Dokuments, so kann dieses als ein neues, eigenständiges, neu klassifiziertes Dokument in das System aufgenommen werden.

Ein Weg zur Auffindung solcher Änderungen bzw. Neuauflagen von Dokumenten ist die Überprüfung von automatisch generierten Checksummen, welche mittels einer Hash-Funktion (siehe Abschnitt 5.2.1) von den Dokumenten errechnet werden können.

5.1.2 Duplikate

Digitale Dokumente lassen sich besonders einfach kopieren. Schätzungen besagen, dass es sich bei etwa 30% des verfügbaren Inhaltes im Internet um Duplikate handelt. Hier sind beispielsweise so genannte Mirror-Sites zu nennen, unter denen viele Software-Unternehmen ihre Software oder Dokumentationen zum Download zur Verfügung stellen. Hier liegen auf mehreren, verschiedenen Servern identische Dateien bzw. Dokumente.

Im Laufe der Recherche-Arbeiten zu dieser Diplomarbeit sind mir zudem hin und wieder Dokumente begegnet, die strukturell und inhaltlich vollkommen identisch waren, aber unter verschiedenen Dokumentnamen aufzufinden waren bzw. sind.

Das hieraus resultierende Problem bezüglich der Signifikanz ist die Verfälschung der Ergebnisse bei der Berechnung der Signifikanz. Denn wird ein Dokument doppelt oder öfters einer identischen Klasse zugeordnet, so geht dieses Ergebnis auch doppelt oder mehrfach in die Gewichtung der übereinstimmenden Schlüsselworte zwischen Dokument und Taxonomieklasse ein. Dieser Effekt würde zu einer unnötigen Verfälschung der Signifikanz und der hieraus später resultierenden Klassifikation führen und sollte daher verhindert werden.

Zur Auffindung solcher Duplikate kann wiederum der Hash-Wert eines Dokuments herangezogen werden.

5.1.3 Dateiformate

Digitale Dokumente liegen in den unterschiedlichsten Dateiformaten vor. Dabei soll unter „Dateiformat“ die unterschiedliche Kodierung eines Dokuments verstanden werden.

Somit muss ein Klassifizierungssystem bzw. Indexierungssystem Filter für die verschiedenen Dateiformate besitzen. Ohne solche Filter könnten sonst Dateiformate wie beispielsweise Microsoft Word, PDF oder andere Formate aufgrund ihrer Kodierung nicht eingelesen bzw. analysiert werden. Erst durch die Extraktion des reinen Textes mittels dieser Filter können die Inhalte solcher Formate eingelesen und ausgewertet werden. Auch müssen Formatierungszeichen, welche im reinen Text vorliegen, wie beispielsweise das HTML-Format, über Filter entfernt werden, damit diese Zeichen nicht das Ergebnis der Auswertung verfälschen. Denn dieser Formatierungscode würde natürlich automatisch mit in die Bewertung einfließen und somit die Ergebnisse der Signifikanz der einzelnen Schlüsselworte verfälschen, aber über den Inhalt des Dokuments nichts aussagen.

Das bedeutet also, dass für jedes existierende Dateiformat spezielle Filter benutzt werden müssen, um an den Klartext bzw. den Inhalt der Dokumente zu gelangen, um diese auszuwerten zu können.

5.1.4 Dokumentgröße

Ein weiteres Problem, natürlich nicht nur bei digitalen Dokumenten, sondern auch allen anderen Dokumenten, ist die Dateigröße bzw. seine Länge. Es macht schließlich bei seiner Analyse einen Unterschied, ob ein Dokument A nur 200 Wörter besitzt oder ein Dokument B 50 000 Wörter. Betrachtet man das Dokument auf Dokumentebene und möchte so eine Aussage über die Signifikanz eines Wortes *w*, welches im Dokument A 20 mal und im Dokument B 60 mal vorkommt, machen, so kann man auf Dokumentebene nur sagen, dass in beiden Dokumenten das Wort *w* vorkommt. Statistisch gesehen, bezüglich der Signifikanz des Wortes *w,* kann man die beiden Dokumente also nicht unterscheiden. Sie sind beide gleich relevant.

Betrachtet man die Dokumente hingegen auf Wortebene, so lässt sich sagen, dass im Dokument A das Wort *w* 20 mal und im Dokument B 60 mal auftritt. Statistisch gesehen, bezüglich der Signifikanz des Wortes *w,* lässt sich aus dieser Sicht sagen, dass das Dokument B relevanter ist, da hier die Wortanzahl des Wortes *w* bei 60 liegt. Dieses ist zumindest schon einmal eine Aussage bzw. ein Ergebnis, mit dem man weiterarbeiten könnte. Aber ist das auch richtig? Eventuell nicht, denn das Dokument A hat insgesamt nur 200 Worte und das Dokument B 50 000. Für ein korrektes Ergebnis muss man also noch die Gesamtgröße des Dokuments betrachten bzw. berücksichtigen.

Ein Lösungsansatz wäre beispielsweise die relative Häufigkeit des Wortes *w* in den Dokumenten zu betrachten. Hier teilt man die Häufigkeit des Wortes w durch die Gesamtwortanzahl des Dokuments. Damit ergibt sich ein Ergebnis für Dokument A von 0,1 und für Dokument B von 0,0012. Statistisch gesehen, bezüglich der Signifikanz des Wortes *w,* kann man nun sagen, dass das Dokument A erheblich relevanter ist als Dokument B. Dieses ist auch das Ergebnis, welches unter Einbeziehung der Dokumentlänge das richtige Ergebnis ist.

5.1.5 Korrigierte Häufigkeitsbewertung

Die Häufigkeit des Vorkommens desselben Wortes im Text ist generell ein fragliches Indiz für die Relevanz. Sie könnte auch Ergebnis einer sprachlichen Ungeschicklichkeit sein. Die Häufigkeit darf somit nicht einfach absolut gezählt werden, sondern muss in Beziehung gesetzt werden zur Länge des Textes und zum Umfang des verwendeten Wortschatzes.

Da es sich aber bei den zugrunde liegenden Schlüsselwortlisten um ein von Experten kontrolliertes Vokabular handelt und die zu analysierenden Dokumente nur auf diesen Inhalt hin untersucht werden müssen, kann man, meiner Meinung nach, die Wortschatzbeziehung vernachlässigen. Anders wäre es, wenn man neue Schlüsselworte anhand von häufig in Texten vorkommenden Worten, suchen würde. Hier müsste man diesen Faktor schon mitberücksichtigen, damit nicht automatisch aufgrund falscher Verwendung eines häufig benutzten Ausdruckes dieser übernommen wird.

Es muss in diesem Fall nur, wie schon im vorherigen Abschnitt angesprochen, die Länge des Textes mit einbezogen werden, sprich die Gesamtwortanzahl. So ist es möglich die relative Häufigkeit bilden zu können.

5.1.6 Betrachtung seltener Begriffe

Eine andere Sicht auf die Relevanz eines Wortes ist normalerweise auch die Betrachtung ihrer Seltenheit. Auch eher beiläufige Stichworte in einem Dokument können einen wichtigen Beitrag zu seiner Relevanz, zur Gesamtinformation, beinhalten. (Siehe auch IDF-Verfahren in Abschnitt 4.2.2.4)

Aber auch das IDF-Verfahren würde meiner Meinung nach aufgrund der zugrunde liegenden, kontrollierten Schlüsselwortliste nicht hilfreich sein. Es ist anzunehmen, dass aufgrund der relativ geringen Menge der Klasse hinterlegten, und die sie sehr spezifisch beschreibenden Schlüsselworte, diese Schlüsselworte wahrscheinlich in verhältnismäßig vielen, dieser Klasse zugehörigen Dokumenten aufzufinden sind. Und gerade die Worte, die aus der Menge der hinterlegten Schlüsselworte in der Gesamtheit nicht oft in den Dokumenten enthalten sind, sind diejenigen Worte, welche für eine Gewichtung bzw. für ihre Signifikanz innerhalb ihrer Klasse nicht von großer Relevanz sind.

5.2 Lösungsalgorithmen

Die in dem vorherigen Abschnitt schon erwähnten Lösungsalgorithmen, wie das Hashing und die relative Häufigkeit, sollen im Folgenden Abschnitt genauer erläutert werden.

5.2.1 Hashing

Wie schon erwähnt, kann man zur Auffindung von Duplikaten und geänderten Dokumenten eine automatisch generierte Checksumme heranziehen. Dieses lässt sich durch so genannte Hash-Funktionen realisieren.

Hashing[12] ist eine Transformation eines großen oder unendlichen Definitionsbereiches in einen kleineren, endlichen Zielbereich. Normalerweise wird

[12] Wörtlich übersetzt heißt „hashing“ zerhacken

z.B. die Menge aller durch 32, 64, oder bei dem md5-Algorithmus 128 Bit darstellbaren Zahlen als Zielbereich verwendet.

Ein Hashing-Algorithmus ist eine Funktion *h: h(x) = y*, für die folgende Aussagen gelten:

1. Für jedes *x* gibt es genau ein *y*.
2. Da die Menge des Definitionsbereiches größer als die des Zielbereiches oder unendlich ist, kann es für mehrere *x* ein gleiches *y* geben.
    a. Hieraus folgt, dass eine eindeutige Rück-Konvertierung von *y* zu *x* aufgrund des kleineren Wertebereiches nicht möglich ist. Dies heißt, eine eindeutige inverse Funktion *h': h'(y) = x* existiert nicht.

Aufgrund dieser Regeln folgt, dass jede Hashfunktion surjektiv, aber nicht injektiv ist.

Findet man nun ein *x*, *y*, für die gilt *h(x) = h(y)*, so spricht man von einer Kollision. Die Anzahl der Kollisionen für eine bestimmte, im Allgemeinen domänenabhängige Wertemenge, gilt gewöhnlich als „*Qualität*" einer Hashfunktion. Je weniger Kollisionen die Hashfunktion für die gegebene Wertemenge also hat, desto besser ist diese Hashfunktion.

Da 32 oder 64 Bit für die geforderte, extrem niedrige Wahrscheinlichkeit von Kollisionen nicht ausreichen, habe ich für die Hash-Wert-Berechnung des Dokuments 128 Bit gewählt, nämlich den MD5-Algorithmus[13].

Liegt nur noch der Hash-Wert vor, so ist es nicht mehr möglich, die exakte Zeichenkette zu bestimmen, die mit der Hashfunktion behandelt wurde. Anhand des Hash-Wertes des Dokumentes lassen sich nun folgende Aussagen machen:

- Wird die Existenz des Dokuments anhand des Hash-Wertes geprüft und fällt diese Prüfung negativ aus, so ist diese Aussage immer richtig.
- Fällt die Prüfung hingegen positiv aus, wird also der eben errechnete Hash-Wert schon bei einem anderen im System vorhandenen Dokument gefunden,

[13] MD5 – „Message Digest Algorithm", siehe auch http://www.faqs.org/rfcs/rfc1321.html

so ist diese Aussage mit einer sehr hohen Wahrscheinlichkeit richtig, kann aber aufgrund der Möglichkeit von Kollisionen falsch sein.

Um das Problem, bzw. die Wahrscheinlichkeit einer Kollisionen auf ein Minimum (im idealsten Fall auf *0*) zu reduzieren, könnte man den Hash-Wert desselben Dokuments mit einer zweiten, anderen Hashfunktion (z.B. dem SHA-1 Algorithmus[14]) doppelt berechnen lassen und beide Werte in der Datenbank abspeichern. Tritt jetzt bei einem der Werte eine Übereinstimmung auf, prüft man dieses auch bei dem zweiten Wert. Die Wahrscheinlichkeit, dass zufällig gerade bei zwei verschiedenen Hash-Funktionen bei einem identischen Dokument eine Kollision auftritt, ist fast gleich 0. Um auch diese fast unmögliche Wahrscheinlichkeit noch auszuschließen, kann man Hash-Funktionen mit noch höherer Qualität wählen. [BUR04]

5.2.2 relative Häufigkeit

Hier muss man zunächst zwischen Statistik und der Wahrscheinlichkeitstheorie aus der Mathematik unterscheiden. Diese beschäftigt sich mit dem Verrechnen von (vorgegebenen) Wahrscheinlichkeiten. Die Statistik hingegen ist eine in Teilen empirische Wissenschaft. Ihre Aufgabe ist es, unter anderem, aus Beobachtungsdaten (Stichproben) auf Wahrscheinlichkeiten zu schließen.

Die Definitionen der absoluten Häufigkeit und der relativen Häufigkeit lauten wie folgt:

$$H_n(E) = k$$

Formel 5-1: Absolute Häufigkeit

$$h_n(E) = \frac{k}{n}$$

Formel 5-2: Relative Häufigkeit

Wobei *n* für die Anzahl des durchgeführten Zufallsexperiments steht, bei denen das Ereignis *E k-mal* eingetroffen ist.

[14] SHA-1 – „US Secure Hash Algorithm 1“, siehe auch http://www.faqs.org/rfcs/rfc3174.html

Das „*starke Gesetz großer Zahlen*“ sagt folgendes aus: „Mit wachsender Versuchszahl stabilisiert sich die relative Häufigkeit eines gegebenen Ereignisses im Allgemeinen bei einer bestimmten Zahl *p*, die relativen Häufigkeiten unterscheiden sich immer weniger von dieser Zahl.

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} y_i = \mu$$

Formel 5-3: Starkes Gesetz großer Zahlen

Die relative Häufigkeit wird oft zur Schätzung der Wahrscheinlichkeit verwendet. Man bezeichnet Wahrscheinlichkeiten, die man derart a posteriori aus relativen Häufigkeiten erhält, daher auch als empirische (oder statistische) Wahrscheinlichkeiten. Mit Hilfe von Wahrscheinlichkeiten kann man dann auch vernünftige Prognosen abgeben. Denn wenn ein Zufallsexperiment n mal durchgeführt wird, so erwartet man auch dass ein bestimmtes Ereignis *E* mit einer Wahrscheinlichkeit $p = P(E)$ dabei $n * p$-mal eintritt, denn $p \approx \frac{k}{n}$ und folglich ist umgewandelt $k \approx n * p$. Dieses gilt natürlich auch für große *n*.

5.3 Errechnung der Signifikanzvektoren

Wie aus den letzten beiden Abschnitten ersichtlich, ist die wohl einfachste und effektivste Möglichkeit den Schlüsselworten Gewichte zukommen zu lassen bzw. eine Aussage bezüglich ihrer Signifikanz machen zu können, die Zählung der gemeinsamen Schlüsselworte von klassifizierten Dokumenten und ihren entsprechenden Klassen. Dokumente werden zunächst manuell klassifiziert, um festzustellen auf welche Schlüsselworte das Dokument analysiert bzw. welche Schlüsselworte gezählt werden müssen. Aufgrund der hinterlegten Schlüsselwortliste muss nicht jedes Wort im Dokument verglichen werden, sondern immer nur nach den dem Dokument zugeteilten Klassen hinterlegten Schlüsselworten.

Jetzt werden alle in diesem klassifizierten Dokument vorkommende Schlüsselworte gezählt, die mit dem dieser Klasse hinterlegten Schlüsselworte (aus der Schlüsselwortliste) übereinstimmen. Die auf diese Weise ermittelte Gewichtung erfolgt somit aufgrund des Häufigkeitsvorkommens jedes einzelnen,

übereinstimmenden Schlüsselwortes des Dokumentinhaltes und dem Dokument zugeordneten Klasse bzw. Klassen hinterlegten Schlüsselwortliste bzw. Vektor.

Für die Berechnung der relativen Häufigkeit eines Schlüsselwortes einer Klasse mit mehreren zugehörigen Dokumenten muss nun, aufgrund der Einbeziehung der Dokumentgröße jedes einzelnen Dokuments, die einzelne relative Häufigkeit dieses Schlüsselwortes in jedem Dokument summiert werden. Anschließend muss diese Summe durch die Gesamtanzahl der Klasse zugeteilten Dokumente geteilt werden. Würde man dieses nicht machen, würde die Beziehung der Länge jedes einzelnen Dokuments verloren gehen. Folglich würden die Dokumente mit unterschiedlichen Gewichten in die Berechnung der Signifikanz eingehen.

Da sich laut starkem Gesetz großer Zahlen aus dem vorherigen Abschnitt die Häufigkeit erst mit steigender Versuchszahl stabilisiert, muss somit versucht werden eine maximale Menge von Dokumenten einer Klasse zu finden. Nur so ist eine brauchbare Aussage, bezüglich der Signifikanz der Schlüsselworte in ihren zugehörigen Klassen, zu machen.

Damit man für jeden Taxonomieknoten bzw. für jede Klasse einen Signifikanzvektor errechnen kann, muss hierfür auch jede Klasse einzeln mit ihren zugehörigen Dokumenten betrachtet und ausgewertet werden.

Abschließend ergibt sich somit folgende Formel für die Berechnung der Signifikanz bzw. der relativen Häufigkeit für ein Schlüsselwort *w* einer Klasse *i*:

$$Signifikanz(w_i) = \frac{\sum relative\ Häufigkeit\ von\ w_i\ in\ \ jedem\ einzelnen\ Dokument\ in\ i}{\sum aller\ i\ zugeteilten\ Dokumenten}$$

Formel 5-4: Signifikanz eines Schlüsselwortes

Dabei steht w für das jeweilige Schlüsselwort der Klasse i.

Der Signifikanzvektor einer Klasse wird folglich wieder durch die einzelnen Signifikanzen dieser Klasse hinterlegten Schlüsselworte gebildet.

6 Konzept

In den vorangegangenen Kapiteln wurden die theoretischen Grundlagen für die automatische Klassifikation von Dokumenten und die Berechnung der Signifikanzvektoren bzw. der relativen Häufigkeit von Schlüsselworten erklärt. Die hieraus gewonnen Kenntnisse wurden für die Umsetzung dieses Konzeptes angewendet und berücksichtigt.

Dieses Kapitel beschreibt den Weg, wie mit Hilfe einer Webapplikation aus einem gegebenen Klassifikationsschema mit seinen Klassen und ihren hinterlegten Schlüsselwortlisten, in Kombination mit einer manuellen Klassifizierung von Dokumenten für jede Klasse ein Signifikanzvektor erstellt werden kann

Als erster Schritt für die Erzielung des Vorhabens erfolgen die Erstellung der notwendigen Anforderungen, sowie die Betrachtung des Ist-Zustandes der momentanen Situation und die Analyse existierender Systeme. Weiter wird ein ergonomisches Userinterface entwickelt. Hierauf folgend wird die zugrunde liegende relationale Datenbank entwickelt. Abschließend folgt der Schritt zur Umsetzung der Anforderungen und den damit verbundenen Funktionalitäten.

6.1 Anforderungen

Um zunächst an die einzelnen Klassen und ihre beschreibenden Schlüsselworte der Taxonomie zu gelangen, ist es notwendig die XML Datei zu analysieren und die gewünschten Daten hieraus zu extrahieren. Zudem muss berücksichtigt werden, dass sich die Klassenanzahl bzw. die Schlüsselworte der Taxonomie im Laufe der Zeit ändern können.

Da sich laut Empirischem Gesetz großer Zahlen aus dem Abschnitt 5.2.2 die Häufigkeit erst mit steigender Versuchszahl stabilisiert, wird eine Mindestmenge von Dokumenten benötigt, welche den entsprechenden Klassen zugeordnet werden kann und die auf das Vorkommen der entsprechenden Schlüsselwörter der jeweiligen Klasse hin analysiert werden können. Je mehr Dokumente pro Klasse analysiert werden, desto aussagekräftiger wird die errechnete Signifikanz der einzelnen Schlüsselworte in ihren zugehörigen Klassen. Eine Konvergenz der einzelnen

Signifikanzen ist somit mit steigender Anzahl richtig klassifizierter Dokumente anzunehmen. Ziel muss folglich sein, möglichst viele, unterschiedliche Dokumente zu den einzelnen Klassen zu finden und ihnen zuzuordnen.

Wo kommen aber diese Dokumente her? Zum einen ist vorstellbar, dass der Benutzer bereits Dokumente besitzt, sei es auf der Festplatte eines Computers oder auf externen Speichermedien. Hieraus folgt, dass eine Upload-Funktion implementiert werden sollte. Zum anderen ist es aber auch denkbar, dass der Benutzer erst Dokumente suchen möchte bzw. muss, um sie darauf folgend einer Klasse zuzuordnen. Es wäre somit praktisch, wenn eine Websuche realisiert würden werde, aus der man die gefundenen Dokumente direkt nach einer manuell durchgeführten Klassifikation auf die jeweiligen Schlüsselworte hin analysieren lassen könnte.

Um aus den Dokumenten und den einzelnen Schlüsselworten die Signifikanzvektoren der einzelnen Taxonomieknoten bzw. Klassen zu bestimmen, muss das betroffene Dokument zunächst auf seinen Inhalt hin untersucht, bzw. die der betroffenen Klasse hinterlegten Schlüsselworte in seinem Inhalt gefunden werden (siehe auch Abschnitt 4.2.2). Das Vorkommen jedes einzelnen, betroffenen Schlüsselwortes im Dokument muss daraufhin gezählt und gespeichert werden. Gleiches gilt aufgrund der benötigten relativen Häufigkeit (siehe Abschnitt 5.2.2) für die Gesamtwortanzahl der im Dokument vorkommenden Worte errechnen lässt.

Zur Extraktion der einzelnen Wörter aus dem Inhalt eines Dokuments ist es nach Abschnitt 5.1.3 notwendig, das Problem der unterschiedlichen Dateiformate zu beachten. Um an den Inhalt des Dokumentes jedes Dateiformates zu kommen, sind entsprechende Formatfilter bzw. Präparatoren erforderlich, die den reinen Klartext aus den jeweiligen Dateiformaten extrahieren und zur Weiterverarbeitung zur Verfügung stellen.

Generell kann zwar davon ausgegangen werden, dass die Applikation nur von Experten benutzt wird, sie sollte aber auch von Nicht-Experten einfach und leicht bedient werden können.

Diese Applikation soll wird am Beispiel der ACM CSS Taxonomie (Siehe Abschnitt 3.2.1) realisiert werden. Es sind aber, wie in dem dritten Kapitel der Klassifikationsschemata lesbar, auch andere hierarchische Taxonomien denkbar. Somit wäre es vorteilhaft, wenn auch mehrere verschiedene Taxonomien von der zu erstellenden Applikation bearbeitet werden können. Das heißt, für jede Taxonomie mit seinen gesamten Klassen müssen die Signifikanzvektoren voneinander unabhängig errechnet werden können. Dieses wäre aber nur möglich, wenn für jede Taxonomie eine eigenständige Webapplikation erstellt werden würde. Da dieses aber nicht im Rahmen dieser Diplomarbeit zu realisieren ist, könnte man dieses auch über einzelne, eigenständige Projekte innerhalb einer Applikation realisieren. Hier ist dann jedes Projekt für eine eigene Taxonomie verantwortlich. Dabei ist es egal, ob es sich um eine identische oder eine andere Taxonomie handelt. Denkbar wäre somit auch, dass zwei Benutzer, unabhängig voneinander, für sich an zwei verschiedenen Projekten die gleiche Taxonomie bearbeiten könnten.

Hiermit ergeben sich folgende Anforderungen, die an die zu realisierende Webapplikation gestellt werden:

- Unterstützung verschiedener Taxonomien
- Ein Projekt anlegen und löschen können
- Ein XML-Parser, der die einzelnen Klassen und ihre zugehörigen Schlüsselworte aus der XML-Taxonomie-Datei extrahiert und sie zur Weiterverarbeitung bereitstellt.
- Neue Klassen der Taxonomie zusätzlich hinzufügen
- Einzelne Schlüsselworte der Taxonomie zusätzlich hinzufügen, bzw. entfernen
- Ein Dokument-Upload aus dem Dateisystem
- Eine Dokument-Suche im Internet
- Duplikate und veränderte Dokumente laut Kapitel 5.1.1 und 5.1.2 von schon im System vorhandenen Dokumenten erkennen
- Dokumente manuell in eine beliebige Klassen einer Taxonomie klassifizieren
- Dokumente aus den hineinklassifizierten Klassen wieder entfernen
- Die Berechnung der Signifikanz aller betroffenen Schlüsselworte

- Format-Filter, die den Klartext aus den jeweiligen Dateiformaten extrahiert und der Textanalyse zur Verfügung stellt
- Eine Textanalyse, die diese Schlüsselworte auffindet und diese, wie auch die Gesamtanzahl aller analysierten Worte zählt
- Bereitstellung der Signifikanzvektoren für andere Anwendungen

6.2 Ist-Analyse

Das Klassifikationsschema liegt als XML Datei vor. In ihr stehen alle zunächst benötigten Daten wie alle Klassen der Taxonomie und ihren zugeordneten Schlüsselworten, welche sie identifizieren sollen und für die die Signifikanz zu errechnen ist.

Um die Signifikanz einzelner Schlüsselworte errechen zu können, werden wie beschrieben eine Vielzahl von Dokumenten gebraucht, die alle zur Errechnung der Signifikanz auf benutzte Schlüsselworte hin analysiert werden müssen. Einige Dokumente existieren entweder lokal auf einem Rechnersystem, auf externen Speichermedien oder, wie die meisten wahrscheinlich, auf Servern im Internet. Um die im Internet „verstreuten" Dokumente gezielt nach ihren Inhalten auffinden zu können, ist wie beschrieben eine Websuche erforderlich. Der größte und meist benutzte Suchdienst ist derzeit Google. Um an die Ergebnisse seiner Websuche heranzukommen, stellt Google neben dem üblichen Webinterface unter anderem ein API zur Verfügung. Dieses API soll die Basis für die Dokumentsuche im Internet dieser Webapplikation dienen.

Für die Benötigte Textanalyse gibt es mehrere, schon existierender und frei verfügbarer Systeme bzw. Lösungen. So beispielsweise Jakarta Lucene[15], welches bei dieser Webapplikation für die erforderliche Textanalyse integriert wird.

Für die nach Abschnitt 5.1.3 benötigten Filter für die einzelnen Dateiformate werden frei verfügbare, als Open Source zur Verfügung stehende Präparatoren verwendet, die wie in der Regain Anwendung der Jakarta Lucene Textanalyse vorgeschaltet werden.

[15] URL: http://lucene.apache.org/java/docs/

6.2.1 Das Google API

Bei der Suche wird auf das System von Google zugegriffen und deren Suchergebnisse zurückgegeben. Dieses wird mit Hilfe dem zur Verfügung stehenden API von Google realisiert. Mit Hilfe dieses API hat man direkten Zugriff auf die Datenbanken von Google.
Vorraussetzung für die Benutzung des API ist eine Registrierung unter *www.google.de/apis*. Per E-Mail erhält man nun einen Lizenzschlüssel, der bei jeder Abfrage mit übermittelt werden muss. Dieser Schlüsselcode ist einem Konto zugeordnet und erlaubt es, maximal 1000 Abfragen pro Tag durchzuführen.

Folgende Parameter müssen dem API bei einer Anfrage übermittelt werden:

- *key* (Lizenzschlüssel)
- *q* (Suchstring)
- *start* (Startwert, von dem an die Ergebnisse zurück geliefert werden)
- *maxResults* (Anzahl der zurück zugebenden Ergebnisse, Maximal 10)
- *filter* (gibt an, ob Filter gesetzt wurden)
- *restricts* (Einschränkungen, z.B. nur Seiten aus Deutschland, etc.)
- *safeSearch* (Filter für jugendgefährdete Inhalte)
- *lr* (Einschränkung Sprache, z.B. nur deutschsprachige Seiten)
- *ie* (bedeutungslos, muss aber gesetzt werden)
- *oe* (bedeutungslos, muss aber gesetzt werden)

Als Ergebnis erhält man eine Instanz der Klasse *GoogleSearchResult*. In ihr sind u.a. die Gesamtanzahl der bei Google unter diesem Suchstring gefundenen Treffer enthalten. Dieser Wert steht in der Variablen „*estimatedTotalResultCount*“. Dieses ist jedoch nur die Anzahl der angeforderten, nicht aber der tatsächlich zurück gelieferten Treffer. Eine API Anfrage liefert immer nur max. 10 Ergebnis-Objekte zurück. Um weitere Ergebnisse zu bekommen, muss der zu übergebende Startwert „*start*“ neu gesetzt werden und es muss eine neue Abfrage gestartet werden. Hieraufhin erhält man die Ergebnisse von der Stelle „*start*“ an bis „*maxResults*“ aus der Anzahl „*estimatedTotalResultCount*“ von gefundenen Gesamtergebnissen bei Google.

Die Eigenschaft „*resultElements*“ liefert die tatsächlichen Suchtreffer mit ihren zur Verfügung stehenden Details. Es besteht aus einem Array der Klasse *ResultElement*. Jedes Objekt des Arrays repräsentiert einen Suchtreffer unter anderen mit den folgenden Eigenschaften:

- *summary* (Zusammenfassung des Suchergebnisses)
- *URL*
- *Snippet*
- *Title*
- *cachedSize*
- *directoryCategory*

Der zu übergebende Suchstring „*q*“, sowie alle anderen erforderlichen Parameter der API werden mit Hilfe der Sucheinstellungen der ersten Tabelle der Websuche, fest definierten Variablen und der Suchauswahl zusammen gefügt und an das API übermittelt.
Die Parameter „*key*“, „*maxResults*“, „*filter*“, „*safeSearch*“, „*ie*“ und „*oe*” werden mit festen Variablen übergeben, da es sich hier um Werte handelt, die sich bei den Suchanfragen nie ändern. Das ist bei den anderen Parametern anders, so dass sie beliebig vom Benutzer nach seinen Wünschen gesetzt werden können.

6.2.2 Jakarta Lucene

Jakarta Lucene[16] ist ein Projekt der Apache Jakarta Gruppe und wurde 1997/98 von Doug Cutting gegründet. Lucene besteht aus reinem Java-Code. Bei der Entwicklung von Lucene wurde darauf geachtet, dass es sehr vielseitig einsetzbar ist. So werden die Eingangsdaten so abstrahiert, dass man schlicht alles, was im Entferntesten mit Text zu tun hat, als Grundlage für die Indexierung nutzt. Lucene ist daher keine feste Applikation, die nur für bestimmte Szenarien funktioniert, sondern eher eine Art API.

Die Jakarta Lucene zugrunde liegende Textanalyse wird auch in diese Webapplikation integriert werden. Die verwendete Bibliothek von ist die aktuelle „*lucene-1.4.3.jar*“ Datei.

[16] http://jakarta.apache.org/lucene

Bei Lucene ist sie für die Erstellung des Indexes zuständig. Hierzu übernimmt zunächst ein *Analyzer* die Extraktion der einzelnen im Dokument vorkommenden Worte. Dieser besteht wiederum aus einem *Tokenizer* und spezifischeren *Analyzern*. Der *Tokenizer* nimmt dabei alle Zeichen, die nicht durch Whitespaces unterbrochen werden, als ein Token bzw. Wort an.

Die so extrahierten Wörter können dann noch von den *SimpleAnalyzer*, dem *StopAnalyzer* und dem *StandardAnalyzer* gefiltert werden. Diese zusätzlichen *Analyzer* werden bei der dieser Webapplikation implementierten Version verwendet.

Der *SimpleAnalyzer* wandelt dabei alle Worte in Kleinbuchstaben um.

Der *StopAnalyzer* filtert zusätzlich Worte heraus, die in einer Stoppwortliste stehen. Dieses sind momentan: einer, eine, eines, einem, einen, der, die, das, dass, daß, du, er, sie, es, was, wer, wie, wir, und, oder, ohne, mit, am, im, in, aus, auf, ist, sein, war, wird, ihr, ihre, ihres, als, für, von, mit, dich, dir, mich, mir, mein, sein, kein, durch, wegen, wird.

Der *StandardAnalyzer* ist dann noch für die Entfernung von der verwendeten Punktierung verantwortlich. Er entfernt beispielsweise alle Punkte und Apostrophe, auch aus Wörtern (so wird U.S.A. zu USA).

Nachdem nun die *Analyzer* den Text in einzelne Worte zerlegt haben, kommt ein *Stemmer* zum Einsatz. Er ist für die Bildung der Stammform der gefundenen Worte zuständig. Lucene bietet dabei zwei verschiedene *Stemmer* an, für englisch sprachige Dokumente den *StandardStemmer* und für deutschsprachige einen *GermanStemmer*. In dieser Applikation wird der *GermanStemmer* verwendet werden, da es sich aufgrund der zugrunde liegenden deutschen Schlüsselworte auch vorwiegend um deutschsprachige zu analysierende Dokumente handeln wird.

Durch die Funktionen der *Analyzer* verringert sich die Gesamtwortanzahl erheblich, so dass die Aussage der Signifikanz der Schlüsselworte innerhalb des Dokuments noch aussagekräftiger wird.

Lucene erwartet jedoch, um diese Analysefunktionen verwenden zu können, immer einen String oder einen Reader mit reinem Klartext.

6.2.3 Verwendete Präparatoren

Um möglichst viele, unterschiedliche Dateiformate verarbeiten zu können, sind auch genauso viele Präparatoren notwendig, die den jeweiligen Klartext aus den jeweiligen Dateiformaten extrahiert. Hier stellt das Open Source Projekt *Regain* unter anderem eine Sammlung der gängigsten Open Source Präparatoren zur Verfügung. Unterstützt werden hier die Formate HTML, PDF, Plaintext Formate, XML, Word, Excel und RTF. Die einzelnen Präparatoren bzw. Bibliotheken, die das *Regain* Projekt sind unter anderen vor allem die Folgenden:

- Apache XML Xerces 2.6.2[17] mit den Bibliotheken xercesImpl.jar und xml-apis.jar. Dieser bietet einen Parser zum Lesen von XML-Dateien.
- PDFBox 0.7.1[18] mit den Bibliotheken PDFBox-0.7.1.jar. Dieser ermöglicht das Lesen von PDF Dokumenten ab Java 1.3 oder darüber.
- Jakarta POI 3.0 alpha 1[19] mit den Bibliotheken poi-3.0-alpha1-20050704.jar und poi-scratchpad-3.0-alpha1-20050704.jar. Dieser ermöglicht das Lesen von Microsoft Excel-Dokumenten und Microsoft Word-Dokumenten.

6.3 Das Datenbank-Modell

Die Applikation besteht aus zwei verschiedenen Datenbanken, der „*admin_db*“ und den jeweiligen „*Projekt-Datenbanken*“.
Beim ersten Aufruf der Applikation wird zunächst nur die Datenbank „*admin_db*“ mit einer Tabelle „*Projekt*“ angelegt. Die Projektdatenbanken werden dann bei jeder Projekterstellung erzeugt.

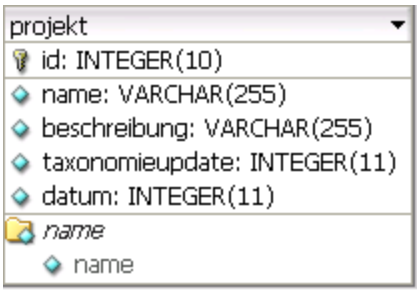

Abbildung 6-1 „admin_db“-Datenbank-Modell

[17] Siehe http://xml.apache.org/xerces2-j
[18] Siehe http://pdfbox.org
[19] Siehe http://jakarta.apache.org/poi

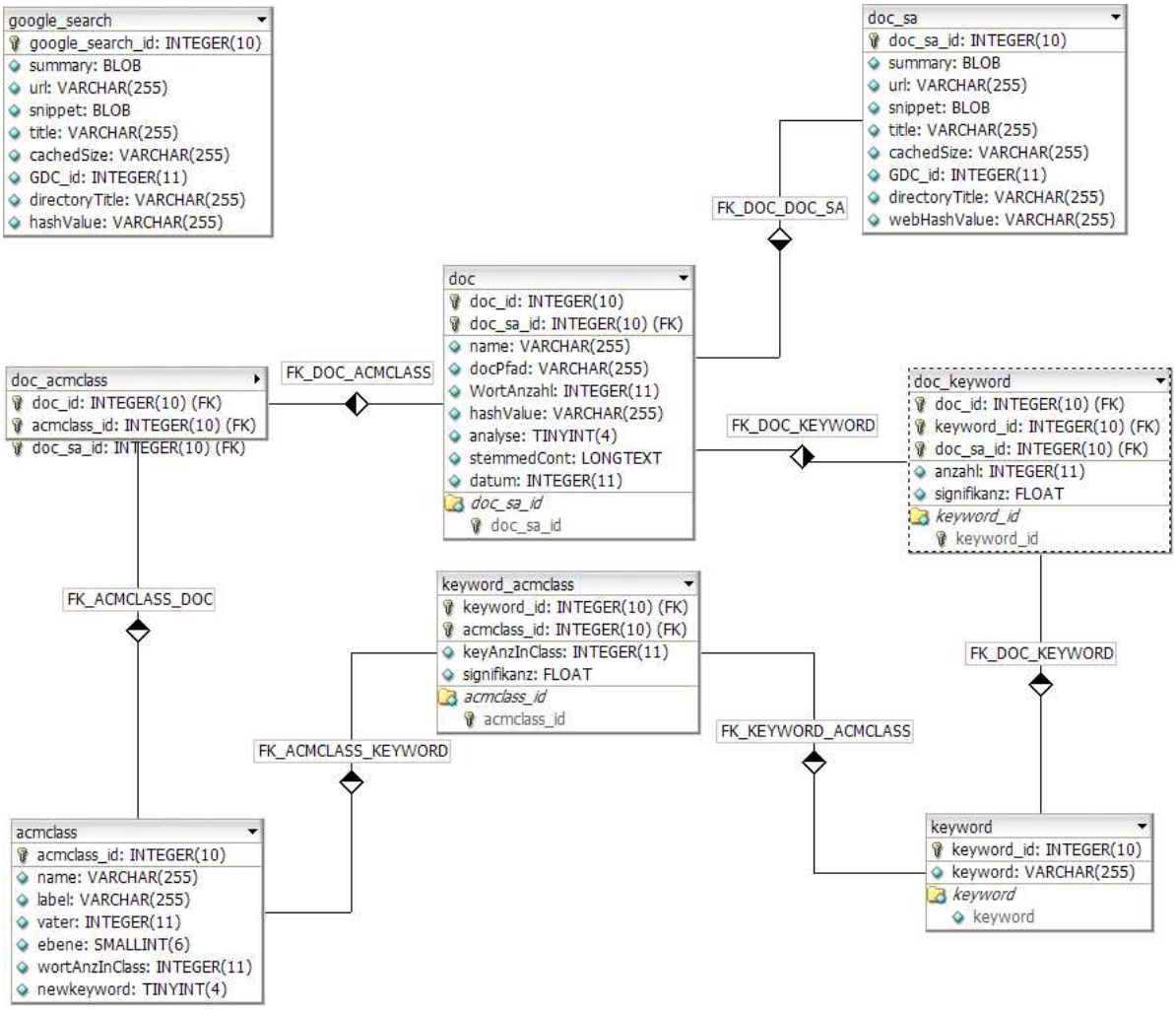

Abbildung 6-2 „Projekte“-Datenbank-Modell

6.3.1 Administrations-Datenbank

Nach dem ersten Aufruf der Installation wird eine Datenbank mit dem Namen „*admin_db*“ und einer Tabelle „*PROJEKT*“ erstellt.

PROJEKT	**Tabelle aller existierenden Projekte**	
FELD	**TYP**	**BESCHREIBUNG**
Id	INT	Eindeutige ID des Projektes
Name	VARCHAR(255)	Name des Projektes
Beschreibung	VARCHAR(255)	Beschreibung des Projektes
Taxonomieupdate	INT	Datum des letzten Taxonomieupdates
Datum	INT	Erstellungsdatum

Tabelle 6-1 Aufbau der Tabelle PROJEKT der ADMIN_DB

6.3.2 Projekt Datenbanken

Bei jeder Erstellung eines neuen Projektes wird eine neue Datenbank mit dem Namen des jeweiligen Projektes angelegt. Zusätzlich werden in der gerade erstellten Datenbank die folgenden Tabellen angelegt:

- ACMCLASS
- KEYWORD
- KEYWORD_ACMCLASS
- GOOGLE_SEARCH
- DOC_SA
- DOC
- DOC_ACMCLASS
- DOC_KEYWORD

KEYWORD	**Tabelle mit den Schlüsselworten**	
FELD	**TYP**	**BESCHREIBUNG**
Keyword_id	INT	Eindeutige ID des Schlüsselwortes
Keyword	VARCHAR(255)	Name des Schlüsselwortes

Tabelle 6-2 Aufbau der Tabelle KEYWORD der PROJEKTDATENBANK

ACMCLASS	**Tabelle mit den Klassen**	
FELD	**TYP**	**BESCHREIBUNG**
Acmclass_id	INT	Eindeutige ID der ACM-Klasse
Name	VARCHAR(255)	Name der ACM-Klasse
Label	VARCHAR(255)	Label der ACM-Klasse
Vater	INT	ID des Vater dieser ACM-Klasse
Ebene	SMALLINT	Hierarchie-Ebene dieser ACM-Klasse
wortAnzInClass	INT	Gesamtanzahl aller dieser Klasse zugeordneter Worte aus den dieser Klasse zugeordneten Dokumenten
Newkeyword	TINYINT	Gibt an, ob dieser Klasse ein neues Schlüsselwort hinzugefügt wurde

Tabelle 6-3 Aufbau der Tabelle ACMCLASS der PROJEKTDATENBANK

KEYWORD_ACMCLASS	Tabelle mit der Verbindung von Klasse & Keyword	
FELD	**TYP**	**BESCHREIBUNG**
Keyword_id	INT	ID des Schlüsselwortes aus KEYWORD
Acmclass_id	INT	ID der ACM-Klasse aus ACMCLASS
keyAnzInClass	INT	Anzahl des Vorkommens dieses Schlüsselwortes in allen Dokumenten aus der Tabelle DOC, die dieser Klasse zugeordnet wurden
Signifikanz	INT	Signifikanz dieses Schlüsselwortes in dieser Klasse

Tabelle 6-4 Aufbau der Tabelle KEYWORD_ACMCLASS der PROJEKTDATENBANK

GOOGLE_SEARCH	Temporäre Tabelle für die Suchergebnisse von Google	
FELD	**TYP**	**BESCHREIBUNG**
Google_search_id	INT	Eindeutige ID des Search-Eintrages
Summary	BLOB	Kurzbeschreibung des Search-Eintrages
url	VARCHAR(255)	URL des Search-Eintrages
Snippet	BLOB	Auszug aus der Verzeichnis-Beschreibung des Search-Eintrages
Title	VARCHAR(255)	Titel des Search-Eintrages
cachedSize	VARCHAR(255)	Grösse des Search-Eintrages
directoryTitle	VARCHAR(255)	Verzeichnisname des Search-Eintrages
webHashValue	VARCHAR(255)	Eindeutiger Hash-Wert genommen aus den Attributen des Search-Eintrages

Tabelle 6-5 Aufbau der Tabelle GOOGLE_SEARCH der PROJEKTDATENBANK

DOC_SA	Tabelle mit den Such-Attributen des Dokumentes	
FELD	**TYP**	**BESCHREIBUNG**
Doc_sa_id	INT	Eindeutige ID der Search-Attribute des Dokumentes
Summary	BLOB	Kurzbeschreibung des Search-Eintrages
url	VARCHAR(255)	URL des Search-Eintrages
Snippet	BLOB	Auszug aus der Verzeichnis-Beschreibung des Search-Eintrages
Title	VARCHAR(255)	Titel des Search-Eintrages
cachedSize	VARCHAR(255)	Größe des Search-Eintrages
directoryTitle	VARCHAR(255)	Verzeichnisname des Search-Eintrages
webHashValue	VARCHAR(255)	Eindeutiger Hash-Wert genommen aus den Attributen des Search-Eintrages

Tabelle 6-6 Aufbau der Tabelle DOC_SA der PROJEKTDATENBANK

DOC	TABELLE MIT DEN EIGENSCHAFTEN EINES DOKUMENTES	
FELD	**TYP**	**BESCHREIBUNG**
Doc_id	INT	Eindeutige ID des Dokumentes
Doc_sa_id	INT	ID der search-Attribute aus DOC_SA
Name	VARCHAR(255)	Name des Dokumentes
docPfad	VARCHAR(255)	Pfad zum Dokument im System
WortAnzahl	INT	Gesamtanzahl aller Wörter des Dokumentes
HashValue	VARCHAR(255)	Hash-Wert vom gesamten Dokument
Analyse	TINYINT	Gibt an, ob das Dokument schon analysiert wurde
Datum	INT	Datum, an dem das Dokument in System übernommen wurde

Tabelle 6-7 Aufbau der Tabelle DOC der PROJEKTDATENBANK

DOC_ACMCLASS	VERBINDUNG DES DOKUMENTES MIT DEN ACM-KLASSEN	
FELD	**TYP**	**BESCHREIBUNG**
Doc_id	INT	ID des Dokumentes aus DOC
Acmclass_id	INT	ID der ACM-Klasse aus ACMCLASS

Tabelle 6-8 Aufbau der Tabelle DOC_ACMCLASS der PROJEKTDATENBANK

DOC_KEYWORD	VERBINDUNG DES DOKUMENTS MIT DEN JEWEILIGEN SCHLÜSSELWORTEN	
FELD	**TYP**	**BESCHREIBUNG**
Doc_id	INT	ID des Dokumentes aus DOC
Keyword_id	INT	ID des Schlüsselwortes aus KEYWORD
Anzahl	INT	Anzahl des Schlüsselwortes KEYWORD_ID in dem Dokument DOC_ID
Signifikanz	FLOAT	Signifikanz des Schlüsselwortes KEYWORD_ID in dem Dokument DOC_ID

Tabelle 6-9 Aufbau der Tabelle DOC_LEXIKON der PROJEKTDATENBANK

6.4 Verzeichnisstruktur

Die Verzeichnisse sind, wie in der folgenden Abbildung zu sehen, angelegt worden.

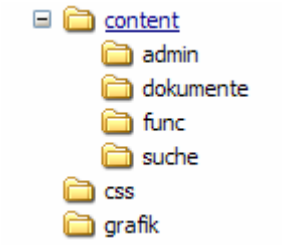

Abbildung 6-3 Verzeichnisstruktur der Applikation

Im Hauptverzeichnis der Applikation befinden sich nach der erfolgreichen Installation vier Hauptverzeichnisse namens „*content*“, „*css*“ und „*grafik*“.

- ***„content“***
  In diesem Ordner befinden sich alle für die Applikation notwendigen Inhalte, Menüs, sowie die erforderliche PHP-Funktionen. Die in diesem Verzeichnis enthaltenen Ordner bzw. Verzeichnisse sind hier wiederum wie folgt strukturiert worden. Jeder Menüpunkt der Applikation besitzt einen Ordner mit identischem Namen, der wiederum für jeden Untermenüpunkt hat eine eigene PHP-Datei beinhaltet. Auch diese Datei ist nach dem jeweiligen Untermenüpunktnamen benannt worden, so dass ein einfaches Auffinden ermöglicht wird.
  Zuzüglich zu den Verzeichnissen der Menüpunkte gibt es noch ein weiteres Verzeichnis namens „*func*“. Dieser beinhaltet alle PHP-Dateien, deren Funktionen von der Applikation auf aufgerufen werden.

- ***„css“***
  In diesem Ordner liegen die Style-Sheets der Applikation.
- ***„grafik“***
  In diesem Ordner befinden sich alle benötigten Grafiken bzw. Bilder.

Nach erfolgreichem Anlegen eines neuen Projektes wird in dem Hauptverzeichnis zusätzlich ein weiteres Verzeichnis mit dem jeweiligen Namen des Projektes angelegt.

- **Verzeichnis „*Projektname*“**
  In diesem Ordner befinden sich alle Dateien, bzw. Dokumente eines Projektes. Hier gibt es zwei Unterverzeichnisse namens „*XMLDateien*“ und „*Dokumente*“.

  - „*XMLDateien*“
    In diesem Verzeichnis werden die benötigten Taxonomie-XML-Dateien abgelegt.
  - „*Dokumente*“
    In diesem Verzeichnis werden alle Dateien, die in entsprechende Klassen hinein klassifiziert wurden, in die entsprechenden Verzeichnisse der Taxonomie-Klassen hinein kopiert. Für jede Klasse, die ein Dokument beinhaltet, wird ein Verzeichnis mit dem jeweiligen Klassennamen angelegt.

6.5 XML-Taxonomie-Datei Struktur

Wie in Abschnitt 6.2 zu lesen liegen die Taxonomie Daten als eine XML Datei vor. Damit die zu erstellende Applikation den Anforderungen entsprechend mit mehreren Taxonomien umgehen kann, muss die Taxonomie in folgender XML-Struktur vorliegen.

Alle Angaben der Taxonomie müssen innerhalb des Tags *<taxonomy>* stehen. Der Name der Taxonomie muss in dem Tag *<name>* folgen, soll eine Beschreibung mitgegeben werden, so ist diese in den Tag *<description>* zu schreiben.
Die Klassen sind innerhalb des tags *<taxon>* anzugeben. Die Attribute, die hier angegeben werden müssen, sind die Bezeichnung bzw. die Notation und der Name der jeweiligen Klasse. Die Klasse ist in den Tag *<entry>* zu schreiben und die Notation in den Tag *<id>*. Ist der Klasse ein Schlüsselwort zugehörig, so muss dieses auf die Klassentags folgen mit dem Namen *<keyword>*.

Hat eine Klasse eine Unterklasse, so ist diese wie oben beschrieben innerhalb des Tags *<taxon>* dieser Klasse einzubauen. Die folgende Abbildung zeigt ein Beispiel anhand der ACM CSS Taxonomie.

```
- <taxonomy>
    <name>ACMCCS2002</name>
    <description>ACM Classification Index 2002</description>
  - <taxon>
      <id>K.</id>
      <entry>en:Computing Milieux</entry>
    - <taxon>
        <id>K.0</id>
        <entry>en:General</entry>
      </taxon>
    - <taxon>
        <id>K.1</id>
        <entry>en:The Computer Industry</entry>
        <keyword>OSF</keyword>
      - <taxon>
          <id>K.1.a</id>
          <entry>en:Markets</entry>
        </taxon>
      - <taxon>
          <id>K.1.b</id>
          <entry>en:Standards</entry>
          <keyword>DIN</keyword>
          <keyword>Deutsche Norm</keyword>
        </taxon>
```

Abbildung 6-4 Struktur einer Taxonomie-XML-Datei

6.6 die Webapplikations-Oberfläche

Obwohl die Applikation in erster Linie für Experten vorgesehen ist, die sich mit Computersystemen und deren Anwendungen auskennen, soll bei der Entwicklung auf die Richtlinien und Normen der ergonomischen Dialoggestaltung geachtet werden.

Besonders hervorzuheben ist hier die EN ISO 9241-10 aus dem Jahr 1996, in welcher die allgemeinen Regeln und Empfehlungen festgehalten werden. [BAL00]
Hier werden unter anderen die folgenden Grundsätze genannt:

- Aufgabenangemessenheit,
- Selbstbeschreibungsfähigkeit,
- Steuerbarkeit,
- Erwartungskonformität,
- Fehlertoleranz,
- Individualisierbarkeit und
- Lernförderlichkeit
- Wartbarkeit

Eine Menge dieser Regeln sind auf die Gestaltung von Expertensystemen ausgerichtet, andere wiederum allgemeiner gültig. Deshalb sollten wie schon erwähnt auch bei der Entwicklung des Userinterfaces diese Regeln berücksichtigt werden. Vor allem die Punkte Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Erwartungskonformität und Fehlertoleranz sind meiner Meinung nach sehr wichtig. Aus diesem Grunde möchte ich diese vier Punkte hier noch einmal nennen und zitieren:

Aufgabenangemessenheit:

> „Ein Dialog ist Aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen."

Selbstbeschreibungsfähigkeit:

> „Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird."

Erwartungskonformität:

> „Ein Dialog ist erwartungskonform, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, z.B. seinen Kenntnissen aus dem Arbeitsgebiet, seiner Ausbildung und seiner Erfahrung sowie den allgemein anerkannten Konventionen."

Fehlertoleranz

> „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingabe entweder mit keinem oder minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann.“

Folgend wird jetzt ein Webfrontend für die Applikation entwickelt. Dabei wurde auf Einhaltung der genannten Ergonomie geachtet. Alle Eingabeaufforderungen werden vor der Weitergabe der Daten auf ihre Vollständigkeit und Korrektheit überprüft.
Beispielhaft zeigt beispielsweise die Abbildung 6.5 des entsprechenden Dialoges bzw. der Maske zur Klassifizierung eines Dokumentes. Zur guten Übersichtlichkeit wird auf eine übersichtliche und klar strukturierte zweistufige Menüstruktur zurückgegriffen und darauf geachtet, dass die angezeigten Informationen möglichst immer komplett auf den Bildschirm passen.

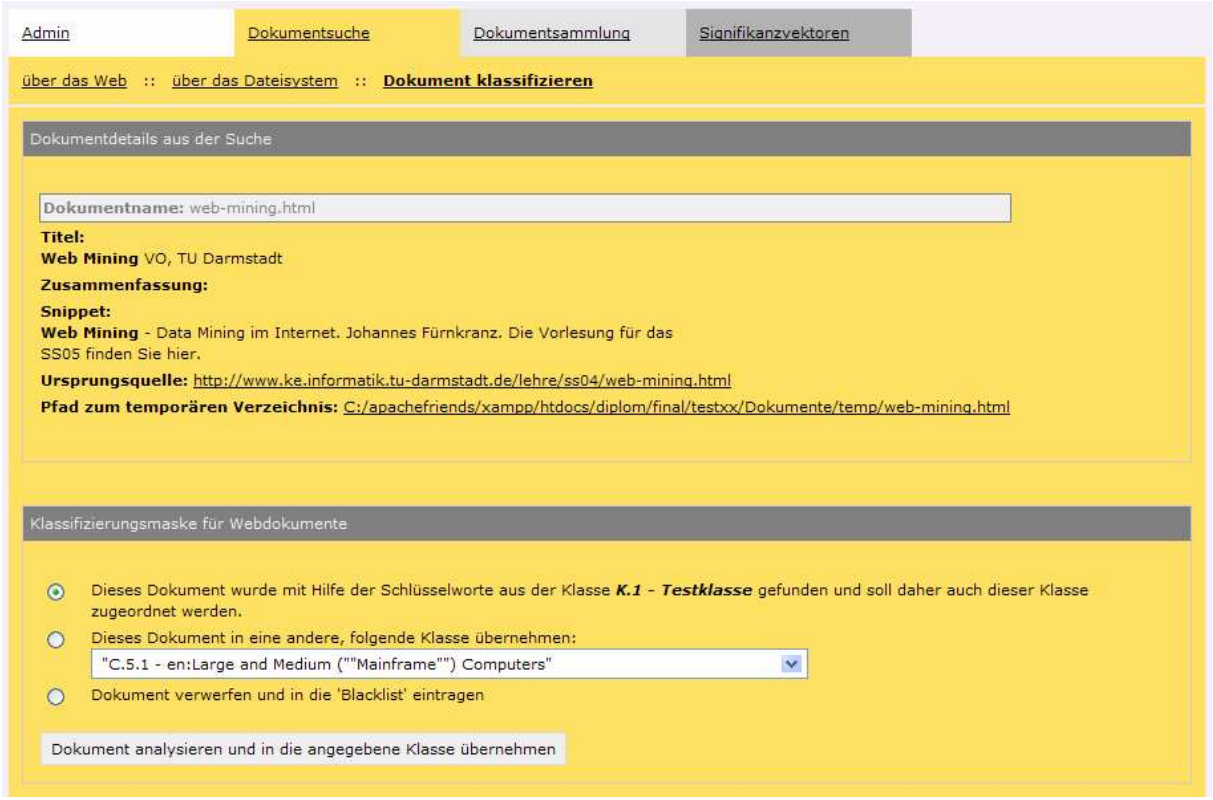

Abbildung 6-5 Klassifizierungsmaske für Webdokumente der Applikation

Alle durch die Applikationen zu machenden Aktionen können ohne Auswirkung auf die restliche Applikation rückgängig gemacht werden.

Aus dem entwickelten Userinterface ergibt sich somit zur Einhaltung der ergonomischen Richtlinien das folgende Layout für die gesamte Applikation.

6.6.1 Layout

Das gesamte Projekt besteht aus nur einer sichtbaren Standardseite namens *index.php*. Dieses bedeutet, dass ohne Frames gearbeitet wird. Dieses hat zum einen den Vorteil, dass die Applikation von allen Browsern aus, auch denen, die nicht Frames unterstützen, aufgerufen werden kann. Dieses war aber nicht ausschlaggebend für diesen Entschluss, sondern viel mehr der Vorteil der Einfachheit, der einfachen Übersichtlichkeit und vor allem der einfacheren Wartbarkeit. Da eine fortlaufende Wartbarkeit, bzw. Weiterentwicklung gut denkbar ist, sollte dieses schon gleich bei der Entwicklung mit berücksichtigt werden. Um hier eine gute Basis, bzw. Übersicht zu schaffen, werden immer nur angeforderte Dateien aus den entsprechenden Verzeichnisordnern des „*content*"-Verzeichnisses geladen und die im Browser sichtbare „*index.php"* Seite eingebunden.
Aufgrund der benutzen Verzeichnisstruktur (siehe Abschnitt 6.4) kann man somit leicht immer nur die betroffenen, eingebundenen „*content*"-Dateien auffinden und anpassen.

Die gesamten Designeigenschaften werden mit Hilfe von Cascading Style Sheets realisiert, welche in der CSS-Datei des „*content*"-Verzeichnisses abgespeichert werden und bei jedem Aufruf der „*index.php"* Seite mit in die Seite eingebunden wird. So kann man später mit Hilfe einer einzigen Datei leicht und ohne großen Aufwand das gesamte Design seinen Vorstellungen entsprechend anpassen.

Das Layout wird in vier Bereiche untergliedert, die entsprechend ihres Linkaufrufes inkludiert werden. Diese sind wie folgt angeordnet:

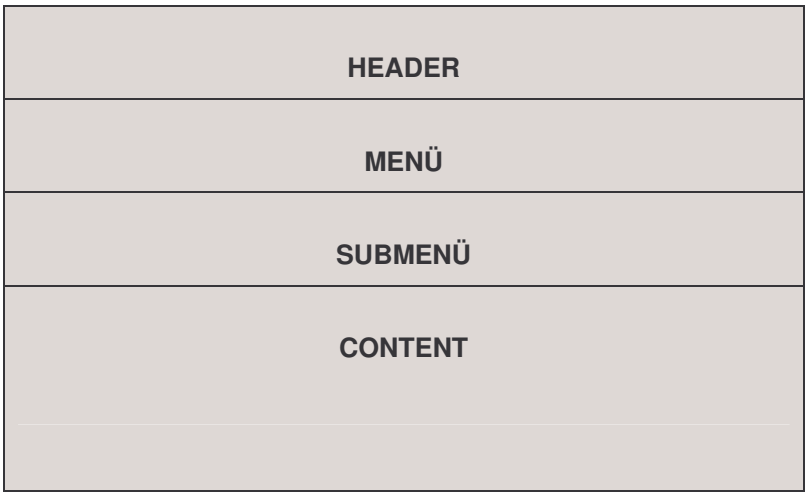

Abbildung 6-6 Webfrontend Oberflächen Strukturierung

- **Header-Modul**
  Hier stehen der Name der Applikation, sowie deren Versionsnummer. Am rechten Rand wird das aktuelle Datum und die Uhrzeit ausgegeben.

- **Menü-Modul**
  Hier werden alle Ober-/Hauptmenüpunkte als Link mit entsprechenden Parametern abgebildet.
  Jeder Menüpunkt erhält eine identifizierende Hintergrundfarbe, so dass es für den User sofort sichtbar ist, in welchem Menüpunkt er sich befindet. Die Menüpunkte lauten wie folgt:

  - Administrationsbereich
  - Dokumentensuche
  - Dokumentensammlung
  - (Signifikanz)-Analyse

- **Submenü-Modul**
  Hier werden die Submenüpunkte des jeweiligen Menüpunktes in der entsprechenden Farbe des aufgerufenen Menüpunktes abgebildet.
  Die Untermenüpunkte werden als Link mit entsprechenden Parametern dargestellt. Ein aktiver Untermenüpunkt wird durch einen fett geschriebenen

Untermenüpunktnamen verdeutlicht und verliert die Linkfunktion. So weiß der Benutzer sofort, wo er sich aktuell auf der Applikation befindet.
Die jeweiligen Sub-Menüpunkte sind in den jeweiligen Beschreibungen der einzelnen Hauptmenüpunkte zu finden.

- **Content-Modul**
  In diesem Bereich werden je nach Linkaufruf aus den Submenüpunkten mit Hilfe der übergebenden Parameter die entsprechend angeforderten Dateien inkludiert, bzw. eingebunden und geladen.

  Dieses Verfahren wird bei jedem Linkaufruf innerhalb der Applikation angewendet. Hier werden mit Hilfe der POST-Methode Parameter übermittelt, die mit entsprechenden Funktionen abgefragt werden, um die angeforderten Dateien in die sichtbare „*index.php*“-Seite einzubinden.

Wird die „*index.php*“-Seite (die Startseite) der Applikation ohne Parameter aufgerufen (dieses ist auch der Fall, wenn man die Seite das erste Mal mit dem Browser aufruft), wird immer der Submenüpunkt „*Projektauswahl*“ aus dem Menüpunkt „*Admin*“ als Standardseite geladen und es muss zunächst ein entsprechendes Projekt ausgewählt werden.

6.6.2 Administrationsbereich

Über den Administrationsbereich kann man die folgenden Submenüpunkte aufrufen:

- Projektübersicht/ -auswahl
- Neues Projekt anlegen
- Taxonomie-Update
- Klassenübersicht
- Schlüsselwortübersicht

6.6.2.1 Projektübersichtsseite

Beim Aufruf dieser Seite erhält der User eine Übersicht aller zurzeit im System (ausgelesen aus der Tabelle „*Projekt*“ aus der Datenbank „*admin_db*“) enthaltenen Projekte und seinen Attributen sowie einer Löschfunktion in folgender Reihenfolge:

- Projektname
- Beschreibung
- Erstellungsdatum
- Letztes Taxonomie-Update
- Projekt Löschen

Hier muss der Benutzer nun mittels Linkaufrufes seines gewünschten Projektes eine Auswahl treffen. Diese Auswahl ist einmalig bei jedem Start der Applikation notwendig. Nach der ersten Wahl wird eine entsprechende Variable in einer Session gespeichert, so dass die Applikation von diesem Zeitpunkt an weiß, mit welchem Projekt der Benutzer arbeiten möchte. Dieses ist erforderlich, da die jeweilige Projektauswahl für die Auswahl der entsprechenden Datenbanken verantwortlich ist, mit denen im weiteren Verlauf bei der Benutzung der Applikation gearbeitet werden soll.

Mit Hilfe des Linkes der Projektauswahl und den entsprechend übergebenen Parameter gelangt der User zum Menüpunkt „*Dokumentsammlung*“ und dem Submenüpunkt „*Übersicht nach Klassen*“.

Sind zurzeit keine Projekte im der Datenbank „*admin_db*“ der Applikation vorhanden, erhält der User einen entsprechenden Hinweis und es erscheint ein Link, der zum Submenüpunkt „Neues Projekt anlegen“ im Obermenüpunkt „Administrationsbereich“ führt.

Wird die „*admin_db*“ ohne die Existenz von dieser aufgerufen, so wird zunächst versucht diese fehlende Datenbank mit der Tabelle „*projekt*“ zu erstellen. Dieses würde beispielsweise der Fall sein, wenn die Applikation noch nicht installiert wurde, bzw. wenn sie das erste Mal nach der Installation aufgerufen wird.

6.6.2.2 Neues Projekt anlegen

Hier erscheint eine Eingabemaske mit folgenden Eingabefeldern:

- Projektname

- Beschreibung des Projektes
- Pfad zur Taxonomie-XML-Datei des Projektes

Nach Angabe des Projektnamens, der Beschreibung und der entsprechenden XML Datei werden nun die gemachten Angeben auf Richtigkeit geprüft. Bei den Feldern Projektname und Beschreibung wird allgemein auf Existenz geprüft, während bei dem Pfad zur XML Datei geprüft wird, ob es sich bei der angegebenen Datei auch wirklich um eine XML Datei handelt. Fällt die gemachte Prüfung positiv aus und es existiert noch kein Projekt mit dem angegeben Namen, wird ein neues Projekt angelegt. (siehe Abschnitt 7.4.1)

Wurde das Projekt erfolgreich angelegt, so erscheint eine Ausgabe mit dem entsprechenden Hinweis und man erhält eine Übersicht über die gefundenen und in die Datenbank übertragenen Klassen und Schlüsselworte. Darunter erscheint die „*Projektübersichtsseite*“.

6.6.2.3 Taxonomie Update

Hier erscheint eine Eingabemaske mit folgenden Eingabefeldern:

- Pfad zur Taxonomie-XML-Datei

Wurde ein Pfad angegeben und auf den Button „*Taxonomie Update durchführen*“ geklickt, wird zunächst die angegebene Datei auf Richtigkeit der Datei geprüft. War die Überprüfung positiv, startet der Update-Prozess. (siehe Abschnitt 7.4.2)

Wurde das Update erfolgreich abgeschlossen, erscheint eine Ausgabe, in der man die Anzahl der neu gefundenen Schlüsselworte und der neu gefundenen Taxonomie Klassen ablesen kann.

6.6.2.4 Klassenübersicht

Hier erhält der Benutzer eine Ebenenübersicht aller im Projekt befindlichen ACM Klassen, die er sich der Ebene nach anzeigen lassen kann. Hier wird der Inhalt

mittels eines SQL-Statements und entsprechenden Parametern aus der Tabelle „*ACMCLASS*“ gelesen und angezeigt.

6.6.2.5 Schlüsselwortübersicht

Hier erhält der Benutzer eine alphabetisch geordnete Übersicht aller im Projekt befindlichen Schlüsselworte. Dabei werden aus der Tabelle „*KEYWORD*“ alle Schlüsselworte mit dem gewünschten Anfangsbuchstaben ausgelesen und aufgelistet. Nun kann sich der Benutzer noch über einen Link „Klassenzugehörigkeit“ alle Klassen auflisten lassen, in dem dieses Schüsselwort vorkommt. Zusätzlich zu der Klassenzugehörigkeit wird hier die Anzahl des Vorkommens dieses Schlüsselwortes in den dieser Klasse zugeordneten Dokumenten aufgelistet, sowie die Signifikanz in dieser Klasse, ein Link zum Löschen des Schlüsselwortes in der aufgelistetem Klasse. Realisiert wird diese Anzeige durch ein INNER JOIN SQL-Statement aus der Datenbank-Tabelle „*ACMCLASS*“, „*KEYWORD*“ und „*ACMCLASS_KEYWORD*“.

6.6.3 Dokumentsuche

Über die Dokumentsuche kann man die folgenden Submenüpunkte aufrufen:

- Dokumentsuche über das Web
- Dokumentsuche über das Dateisystem
- Dokumentsuche über die zuletzt gemachte Webabfrage
- Dokument klassifizieren

6.6.3.1 Dokumentsuche über das Web

Der Content-Bereich der Dokumentsuche über das Web besteht aus vier Bereichen, die zur besseren Übersicht durch einzelne Tabellen dargestellt werden. Es sind die Bereiche „*Sucheinstellungen*“, „*Ebenenzugriff auf die Klassen*“, „*Schnellzugriff auf die Klassen*“ und „*uneingeschränkte Suche*“.

- **Sucheinstellungen**
  Hier besteht die Möglichkeit die folgenden Einstellungen zur Suchanfrage zu ändern:

- Anzahl der Suchergebnisse
- Spracheinschränkungen
- Suchstringverknüpfung
- Dokumenttypen

Mit diesen Einstellungen werden die nun startenden Suchanfragen an das Google-API abgeschickt werden.

- **Ebenenzugriff auf die Klassen**

  Hier hat man den Zugriff über die Ebenen des Taxonomie-Kataloges auf die jeweiligen Klassen. Man klickt sich durch die Ebenen zu seiner gewünschten Kategorie. Wenn man die gewünschte Kategorie erreicht hat, kann man nun mit den dieser Klasse zugeordneten Schlüsselworten und den in den Sucheinstellungen gemachten Angaben eine Suchanfrage auf das Google API starten. Dargestellt wird die Hierarchie aller Klassen der Taxonomie. Zunächst werden nur diejenigen der ersten Ebene mit Link zur zweiten Ebene abgebildet. Durch Anklicken dieses Linkes gelangt man zu den Kindern bzw. zur Unterklasse dieser Klasse, somit zur nächsten Ebene der Taxonomie. Dieses ist solange möglich wie es für die gewählte Klasse Unterklassen gibt. Abgebildet werden die Klassen, indem sie aus der Datenbank-Tabelle „*ACMCLASS*“ mit Hilfe entsprechender Parameter und entsprechenden SQL-Statements gelesen werden. Neben jeden abgebildeten Klassennamen wird ein zweiter Link mit dem Namen „*Für diese Klasse Dokumente suchen*“ dargestellt. Mit ihm startet man eine Suchanfrage mit dem dieser Klasse hinterlegten Schlüsselworten und gelangt anschließend zur Seite Suchergebnisse.

- **Schnellzugriff auf die Klassen**

  Optional kann auch über die Schnellauswahl die gewünschte Klasse ausgewählt werden. Diese besteht aus einem Drop-Down-Menü, indem alle Klassennamen angezeigt werden, jedoch ohne Auflistung der Schlüsselworte. Startet man hier die Suche mit dem Suche-Button, werden jedoch die hinterlegten Schlüsselworte zur Suchstringbildung herangezogen. Startet man die Suche über den Button „*Für diese Klasse Dokumente suchen*“, startet man

eine Suchanfrage mit dem dieser Klasse hinterlegten Schlüsselworten und gelangt anschließend auch zur Seite Suchergebnisse.

- **uneingeschränkte Suche**
  Hier besteht für den Benutzer die Möglichkeit über ein frei formulierbares Suchfeld eine entsprechende Suchanfrage zu starten. Anschließend gelangt er durch den Button „*Suche starten*“ zur Seite Suchergebnisse.

6.6.3.2 Suchergebnisse

Wird diese Seite mit den Parametern der Ebenensuche aufgerufen, so bekommt der User zuzüglich zu den Ergebnissen eine weitere Suchmaske angeboten. Hier werden alle aus der übermittelten Klasse und in der Suchstringbildung benutzten Schlüsselworte einzeln, untereinander aufgelistet. Dieses ist notwendig, da eine Klasse unendlich viele Schlüsselworte besitzen kann und diese im Suchstring bei einer Suchstringverknüpfung mit „*AND*“ sehr wahrscheinlich mit steigender Schlüsselwortanzahl keine Ergebnisse mehr liefern wird. So kann jedes Schlüsselwort mit einer nebenstehenden Checkbox aktiviert werden, um es für die neue Suchanfrage mit zu berücksichtigen. Um den Benutzer zusätzlich alle Optionen der individuellen Suche zu ermöglichen, ist unter dem letzten Schlüsselwort ein freies Textfeld, indem der Benutzer noch eigene Suchwörter zu den schon ausgewählten Schlüsselworten addieren kann.

Jetzt kann der User mit allen aktivierten Schlüsselworten und ggf. seinen zusätzlichen Suchworten eine neue Suchanfrage mit dem Button „*neue Suchanfrage*“ starten und kommt auf diesem Weg erneut auf diese Seite der Suchergebnisse.

Hash-Wert Berechnung der Suchergebnisse

Aus jedem zurück gelieferten Ergebnis der Suchanfrage und seinen Attributen „*summary*“, „*URL*“, „*Snippet*“, „*Title*“ und „*cachedSize*“ wird zunächst ein neuer String zusammengefügt. Aus diesen String wird dann mittels dem md5-Algorithmus ein Hash-Code (siehe 5.2.1) gebildet. Für diese Berechnung ist die PHP-Funktion md5()[20] vollkommen ausreichend. Der Hash-Wert soll dazu dienen, später bei neuen Suchanfragen schnell diejenigen Ergebnisse zu finden, welche schon bei vorherigen Suchanfragen bearbeitet wurden. Wurde ein Dokument über das Web gefunden, so hat dieses Dokument automatisch Suchattribute aus der Suchanfrage von Google und somit den errechneten Hash-Wert der oben genannten Attribute. Diese werden in der Tabelle „*DOC_SA*“ gespeichert. Bei jeder neuen Suche werden die Suchergebnisse zunächst mit ihren errechneten Hash-Werten und den in der Datenbank gespeicherten Hash-Werten der Tabelle „*DOC_SA*“ verglichen. Liegt eine Übereinstimmung dieser Werte vor, so werden diese Ergebnisse farblich hervorgehoben ausgegeben.

6.6.3.3 Dokumentsuche über das Dateisystem

Der Content-Bereich der Dokumentsuche über das Dateisystem besteht aus einer Maske mit Upload-Funktion. Hier kann man über das Dateisystem des Benutzers eine Datei auswählen und mit Hilfe des Buttons „*Datei kopieren*“ einen Kopiervorgang starten. Danach gelangt man mit entsprechenden Parametern zu der Klassifizierungsmaske.

6.6.3.4 Klassifizierungsmaske

Über einen Link „*Dokument klassifizieren*“ aus der Übersichtsliste der Suchergebnisse oder dem Dokument-Upload gelangt man zu der Klassifizierungsmaske des gefundenen Dokuments der Dokument-Suche. Ansonsten ist diese Seite nicht aufrufbar.
Das Dokument wird zunächst in das Verzeichnis „*temp*“ im Verzeichnis „*Dokumente*“ des Projektverzeichnisses gedownloadet bzw. kopiert. Existieren diese Verzeichnisse noch nicht, so werden sie erstellt.
Ich habe hier ein temporäres Verzeichnis gewählt, damit der Benutzer in der weiteren Benutzung nicht unnötig lange warten muss, wenn es sich bei dem Dokument um ein größeres Dokument aus dem Internet handelt. Der Benutzer kann parallel

[20] URL: http://www.faqs.org/rfcs/rfc1321.html

weiterarbeiten, indem er beispielsweise das gewählte Dokument liest oder sich eine passende Klasse aussucht. Somit ist im nächsten Schritt, der Dokumentanalyse dieser Schritt nicht mehr notwendig und es kann direkt mit der Analyse begonnen werden.
Auf dieser Seite erhält der Benutzer noch einmal alle Informationen des Dokumentes wie

- den Titel
- die Kurzbeschreibung
- den Dokumenttyp
- die Dokumentgröße
- die URL[21] des Ursprungsdokumentes
- sowie den Pfad zum temporären Verzeichnis

im Überblick. Zusätzlich wird die Klasse mit ihren Schlüsselworten abgebildet, in der man die Suchergebnisse gefunden hat, wenn man über die Websuche die Dokumente gefunden hat. Über den Link zum Dokument hat der Benutzer nun die Möglichkeit das Dokument sich in einem neuen Fenster anzeigen oder sich öffnen zu lassen, je nach Dokumenttyp. Vorraussetzung hier sind natürlich die erforderlichen, bzw. zugrunde liegenden Programme.

Unterhalb der Dokument-Details erscheint eine Klassifizierungsmaske für das Dokument. Hat der User das Dokument gelesen, hat er nun die Möglichkeit über diese Maske das Dokument in entsprechende Klassen zu klassifizieren. Hier gibt es folgende Klassifizierungsoptionen:

- Dieses Dokument für die angegebene Klasse in die Dokumentsammlung der Applikation zu übernehmen.
- Dieses Dokument in die Dokumentsammlung übernehmen, jedoch nicht für die Klasse über die gesucht wurde, sondern in eine beliebig andere zur Auswahl stehende Klasse. Dazu hat der Benutzer ein Drop-Down-Menü zur Auswahl, in der er alle zurzeit verfügbaren Klassen findet und auswählen

[21] URL – Uniform Resource Locator

kann. Das Dokument wird für die ausgewählte Klasse ins System übernommen.
- Dieses Dokument verwerfen und in die Black-Liste eintragen, damit es in den späteren Suchergebnissen nicht mehr auftaucht.

Wurden alle Angaben in der Maske gemacht, kann der User nun über den Button „*Dokument analysieren*“ eine Textanalyse starten.

6.6.4 Dokumentensammlung

Durch die oben erstellten und beschriebenen Verzeichnisstrukturen, in die die klassifizierten Dokumente abgelegt werden, hat man zusätzlich zu den folgenden Möglichkeiten, schon alleine eine komplette, klassifizierte Dokumentensammlung geschaffen. Ist man z.B. an keinem speziellem Dokument einer Klasse interessiert, sondern allgemein an dem Thema der jeweiligen Klasse, so kann man direkt in der Verzeichnisstruktur des Dateisystems rumstöbern und sich über das Themengebiet informieren.

Über die Weboberfläche der Applikation hat man über den Menüpunkt „Dokumentensammlung“ die folgenden Submenüpunkte-Optionen zur Verfügung:

- Übersicht nach Klassen
- Übersicht nach Schlüsselworten
- Übersicht nach alphabetischer Reihenfolge
- Dokumentdetails

6.6.4.1 Übersicht nach Klassen

Hier erhält man eine Übersicht aller im Projekt befindlichen Dokumente, die man sich anhand der Klassen anzeigen lassen kann. Man wählt eine Klasse aus und erhält die Dokumente, die dieser Klasse zugeteilt wurden. Hier hat man die Möglichkeiten über einen Link „*Details anzeigen*“, sich die Dokument-Details anzeigen zu lassen.

6.6.4.2 Übersicht nach Schlüsselworten

Hier erhält man eine Übersicht aller im Projekt befindlichen Dokumente, die man sich anhand der Schlüsselworte anzeigen lassen kann. Man wählt zunächst ein

Schlüsselwort und erhält die Dokumente, die dieses Schlüsselwort enthalten. Hier hat man die Möglichkeiten über einen Link „*Details anzeigen*“ Dokument-Details anzeigen zu lassen.

6.6.4.3 Übersicht nach alphabetischer Reihenfolge

Hier erhält man eine Übersicht aller im Projekt befindlichen Dokumente, die man sich anhand des alphabetischen Anfangsbuchstaben auflisten lassen kann. Hier hat man die Möglichkeiten über einen Link „*Details anzeigen*“ Dokument-Details anzeigen zu lassen.

6.6.4.4 Dokumentdetails

Hier erhält der Benutzer eine Übersicht folgender Details des ausgewählten Dokuments:

- Dokument-Name
- Den Pfad zum aktuellen Verzeichnis, indem das Dokument gespeichert ist
- Die Gesamtwortanzahl des Dokuments
- Den Hash-Wert (siehe Abschnitt 5.2.1) des Dokuments
- Das Datum, an dem es in das System übernommen wurde
- Wenn vorhanden die Such-Attribute des Dokuments
- Die Anzahl der Klassen, in die dieses Dokument klassifiziert wurde
- Zu jeder gefundenen Klasse die folgenden Klassendetails:
    - Den Klassennamen zuzüglich ihres Labels
    - Die Gesamtwortanzahl dieser Klasse
    - Die Gesamtanzahl aller Dokumente, die dieser Klasse zugeordnet wurden
    - Den Signifikanzvektor dieser Klasse zuzüglich des Schlüsselwortvorkommens in diesem Dokument und der Signifikanz des Schlüsselwortes in diesem Dokument
    - Einen Link zum Löschen des Dokuments aus dieser Klasse

Realisiert werden die Dokument-Details durch ein entsprechendes INNER JOIN SQL-Statement aus der Tabelle „*DOC*“, ggf. „*DOC_SA*“ und „*DOC_ACMCLASS*“. Die

Daten für die Klassendetails werden aus der Tabelle „*ACMCLASS*“, „*ACMCLASS_KEYWORD*“ und „*KEYWORD*“ gelesen.

6.6.5 Signifikanzvektoren

Über den Menüpunkt der Signifikanzvektoren kann man die folgenden Submenüpunkte aufrufen:

- Den Signifikanzvektor einer Klasse anzeigen
- Die Signifikanzvektoren aller Klasse

6.6.5.1 Den Signifikanzvektor einer Klasse anzeigen

Hier kann man aus allen im Projekt befindlichen Klassen über ein Drop-Down-Menü eine Klasse auswählen und mittels des Buttons „*Signifikanzvektor dieser Klasse anzeigen*“ den Signifikanzvektor dieser Klasse anzeigen lassen.

Der Signifikanzvektor besteht aus den einzelnen Schlüsselworten dieser Klasse und ihren jeweiligen Signifikanzen. (siehe Abschnitt 5.3) Diese stehen in dem Attribut „*signifikanz*“ der Tabelle „*ACMCLASS_KEYWORD*“ und werden mittels SQL-Statements ausgelesen.

6.6.5.2 Die Signifikanzvektoren aller Klassen anzeigen

Hier werden von allen sich im Projekt befindlichen Klassen die Signifikanzvektoren dargestellt.

Der einzelne Signifikanzvektor besteht wiederum aus den einzelnen Schlüsselworten der einzelnen Klasse und ihren jeweiligen Signifikanzen. (siehe Abschnitt 5.3) Diese stehen in dem Attribut „*signifikanz*“ der Tabelle „*ACMCLASS_KEYWORD*“ und werden mittels SQL-Statements ausgelesen.

7 Design

In den vorangegangenen Kapiteln wurden die notwendigen Anforderungen und die daraus resultierenden Funktionalitäten beschrieben. In diesem Kapitel soll das zur Umsetzung dieser Applikation notwenige Design entwickelt werden. Anschließend werden die einzelnen Komponenten anhand ihrer Funktionalitäten in der Webapplikation erörtert.

7.1 Kommunikationsdesign

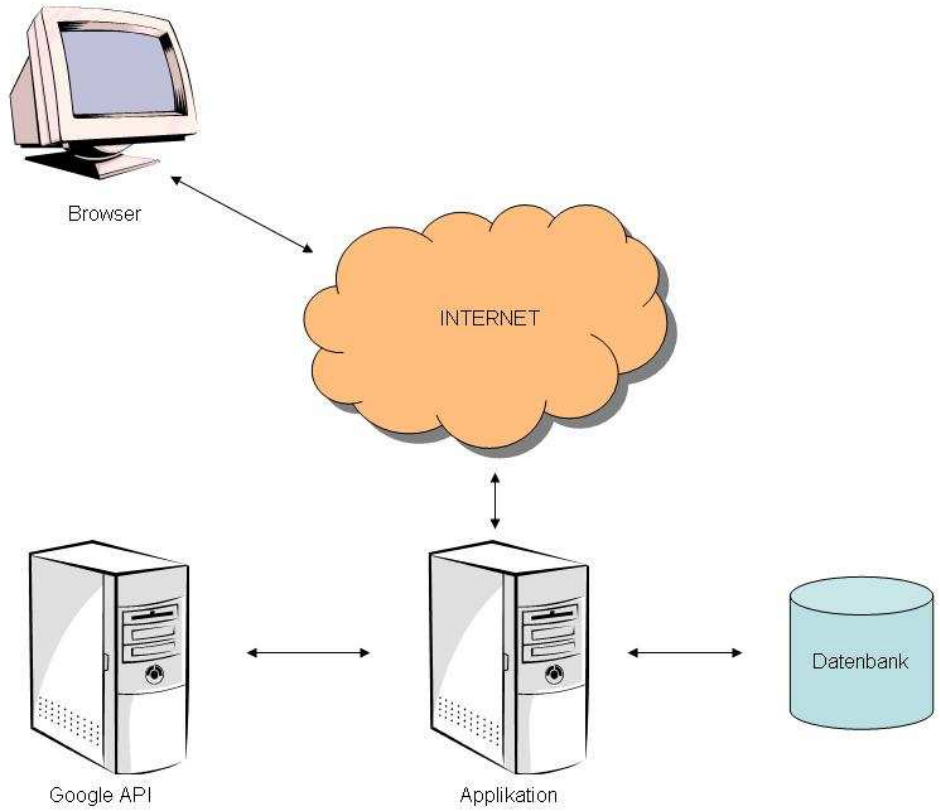

Abbildung 7-1 Kommunikationsmodell

Abb. 7.1 zeigt einen Überblick über die geplante Applikation. Die Kommunikation wird ausschließlich über ein Userinterface im Internet oder einem lokalen Server mit Hilfe eines Webclients und des http-Protokolls[22] laufen. Weiterhin wird für die Dokumentsuche im Web auf ein API von Google und eine relationale Datenbank

[22] Hyper Text Transfer Protocol

zugegriffen. Für die Dateisuche im Dateisystem greift die Applikation auf das Dateisystem des Users zu.

7.2 Systemdesign

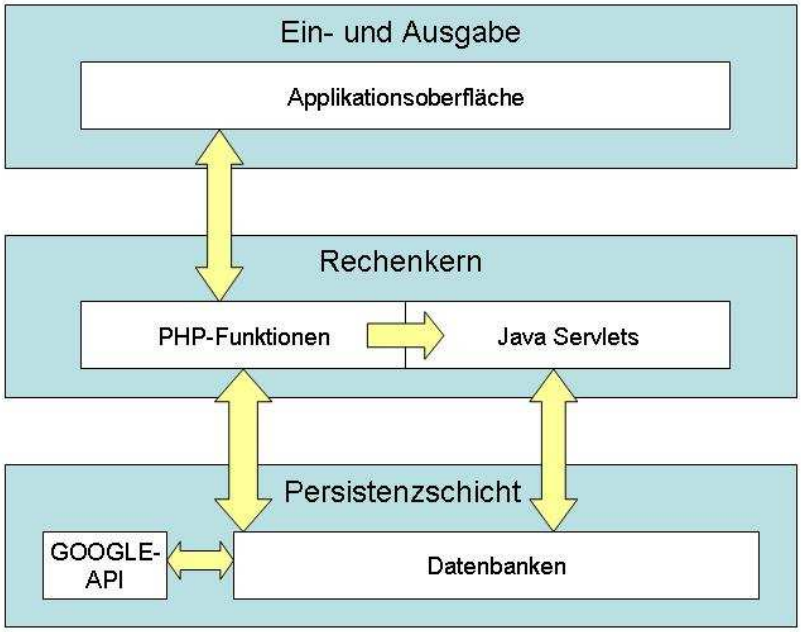

Abbildung 7-2 Übersicht über die Komponenten

Um die Wartbarkeit und Flexibilität der Applikation gewährleisten zu können, wird wie in Abbildung 7.2 dargestellt ist, auf eine klassische Dreischichten-Architektur zurückgegriffen. Neben dem Rechenkern gibt es eine Persistenzschicht sowie eine Ein- und Ausgabeschicht. Der Rechenkern ist für alle Funktionalitäten und deren Folgen verantwortlich. Das Google-API kann als Teil der Persistenzschicht betrachtet werden, da es lediglich grundlegende Informationen bzw. die zu analysierende Dokumente liefert.

Das Userinterface der Webapplikation wird aufgrund der Plattform- und Ortsunabhängigkeit in PHP programmiert. Der zugrunde liegende Rechenkern wird überwiegend, nämlich da wo es aufgrund der erfordernden Funktionalitäten und Ressourcen erforderlich ist, in Java realisiert. Ein weiter Grund für die Verwendung von Java ist der momentane Ist-Zustand auf Seiten der Textanalyse (siehe 6.2.2), da hier alle zur Verfügung stehenden Systeme fast ausschließlich in Java vorliegen. Man hätte theoretisch natürlich auch die komplette Applikation in Java realisieren

können, ich habe mich aber aufgrund der zeitlichen Begrenzung dieser Diplomarbeit für eine weitere Programmiersprache, nämlich PHP entschieden. Hier kommt man meiner Meinung nach sehr schnell und einfach zu einem sichtbaren Ergebnis. Vor allem auf Seiten der Visualisierung, welche mit PHP schell und einfach zu realisieren und zu ändern ist.

Die zu speichernden Daten der Persistenzschicht werden in einer *MySQL* Datenbank gespeichert. Hier hätte man auch andere Datenbank-Systeme wählen können wie beispielsweise *ORACLE*. Ich habe mich aber aufgrund des Kostenfaktors für eine *MySQL* Datenbank entschieden, da es sich hier um ein kostenloses Datenbanksystem handelt und deren Funktionalität und Performance für die Realisierung alles Notwendige bietet.

Die Schnittstelle für den Web-Client wird durch das Userinterface der Applikation realisiert. Diese bietet alle benötigten Eingabe- und Ausgabefunktionen. Die Schnittstelle zwischen der Eingabemaske und der Persistenzschicht wird durch die Funktionen der PHP-Funktionen und den Servlets in Java umgesetzt. Die Schnittstelle zwischen Eingabemaske und Rechenkern wird durch die Funktionen zum Erstellen und Entfernen von Projekten, zur Suche, Integration, Entfernung und Verwaltung von Dokumenten, sowie zu deren Klassifizierung realisiert. Zudem kommen Methoden zur Auswertung der erhobenen Daten aus den gewonnenen Dokumenten.

7.3 Der Rechenkern

Der Rechenkern besteht im Wesentlichen aus zwei Komponenten. Zum einen den Funktionen, die in PHP umgesetzt wurden und in dem Verzeichnis „*func*“ liegen, sowie die in Java programmierten Servlets.

7.3.1 PHP-Funktionen

Der PHP-Kern besteht wiederum hauptsächlich aus vier Dateien, die im Folgenden kurz erörtert werden sollen.

7.3.1.1 „*db_function.php*“

In dieser Datei stehen neben den Funktionen zum Aufbau einer Datenbankverbindung die Funktionen, mit denen die Daten in der Datenbank eingefügt, verändert und gelöscht werden können.

7.3.1.2 „*file_function.php*“

In dieser Datei stehen alle Funktionen, mit denen Verzeichnisse und Dokumente im Dateisystem erstellt, kopiert und gelöscht werden können.

7.3.1.3 „*create_db.php*“

In dieser Datei stehen alle Funktionen, mit denen die Projektdatenbanken bzw. –Tabellen erstellt werden. Diese Funktionen sind in der Programmiersprache PHP realisiert. Um welche genauen Tabellen es sich hier handelt, steht in Abschnitt 7.4.1 „*Neues Projekt anlegen*“.

7.3.1.4 „*XML-Parser.php*“

Der XML-Parser ist in der Programmiersprache PHP realisiert. Er hat die Aufgabe aus der zugrunde liegenden XML-Taxonomie-Datei die benötigen Attribute, bzw. XML-Tags herauszulesen und in entsprechende Datenbanktabellen zu speichern. Zur Erinnerung: die auszuwertende XML Datei besteht aus den Klassen bzw. der Hierarchie einer Taxonomie und ihren hinterlegten Schlüsselworten. Zu speichernde Werte sind somit die einzelnen Klassen mit ihren jeweiligen Schlüsselworten. Damit der Parser die Taxonomie verarbeiten kann, muss diese in der Struktur aus Abschnitt 6.5 vorliegen.

Die Schlüsselworte werden in der Tabelle „*KEYWORD*“, die Klassen in der Tabelle „*ACMCLASS*“ und ihre Beziehung in der Tabelle „*ACMCLASS_KEYWORD*“ gespeichert. Hier kann man dann ablesen, welches Schlüsselwort zu welcher Klasse gehört.

Dem Parser wird die XML Datei übergeben und hier mit Hilfe der Funktion SimpleXML aus der Standardbibliothek von PHP5 in Objekte konvertiert. So ist der gesamte Hierarchie-Baum mit seinen Klassen und Schlüsselworten als Objekt ansprechbar.

Der Parser kontrolliert jetzt, ob er eine Klasse findet, bzw. das Objekt „*taxon*“, welches aus dem XML-TAG *<TAXON>* entstanden ist. Ist dieses der Fall, und es existiert diese gefundene Klasse noch nicht in der Datenbank, wird die Klasse mittels der *Insert-Funktion* aus der „*db_function.php*“ in der Tabelle „*ACMCLASS*“ gespeichert. Bei der Klasse handelt es sich um die gefundenen Werte „*name*“ und „*label*“.

Nachdem der Parser das Objekt „*taxon*“ gefunden hat, sucht er ein Objekt namens „*keyword*“ innerhalb des Objektes „*taxon*“. Existiert nun zu einer Klasse ein Schlüsselwort, so ist es in der XML Datei als Tag *<KEYWORD>* eingetragen und von SimpleXML in das Objekt „*keyword*“ umgewandelt worden. Existiert jetzt ein Schlüsselwort in der gerade bearbeiteten Klasse, wird dieses wieder mittels der Insert-Funktion in die Tabelle „*KEYWORD*“ eingetragen. Als nächstes wird die Beziehung zwischen der Klasse und ihrem Schlüsselwort gespeichert, indem ein Eintrag in die Tabelle „*ACMCLASS_KEYWORD*“ vorgenommen wird. Dabei wird zunächst auf die Existenz dieser Verbindung geprüft. Fällt die Prüfung negativ aus, so wird die Verbindung nicht neu eingetragen. Dieses wäre beispielsweise beim Taxonomie Update der Fall. Hier würden alle existierenden Verbindungen (alle Klassen und Schlüsselwörter) schon in der Tabelle stehen. Daher werden jetzt nur diejenigen gespeichert, die noch nicht in der Datenbank stehen. Fällt die Prüfung somit positiv aus und der Eintrag existiert noch nicht, so wird er neu eingetragen.

Zusätzlich wird jetzt das Attribut „*newkeyword*“ der Tabelle „*ACMCLASS*“ auf „*1*“ gesetzt. Hierdurch weiß der Parser später, welche Klasse ein neues Schlüsselwort bekommen hat. Dieses ist notwendig, da am Ende des Taxonomie Updates alle in diese Klasse klassifizierten Dokumente neu analysiert, bzw. deren Signifikanz neu berechnet werden müssen, um bei dem neu hinzugefügten Schlüsselwort ein Ergebnis für die Signifikanz errechen zu können.

Wird nach einem gefundenen Schlüsselwort bzw. einem Objekt „*keyword*“ als nächstes ein weiteres Objekt „*keyword*“, also ein weiteres Schlüsselwort gefunden, wird der eben beschriebene Vorgang wiederholt.

Wird bei einer gefundenen Klasse kein oder kein weiteres Schlüsselwort gefunden, sucht der Parser erneut wieder nach dem nächsten Objekt „*taxon*“ und beginnt bei erfolgreicher Suche mit der oben beschriebenen Prozedur bis alle „*taxon*“ Objekte abgearbeitet wurden.

Wurde der XML Parser aus der Projektanlegung aufgerufen, so wird nach erfolgreichem Abschluss bei allen Klassen das Attribut „*newkeyword*“ auf „0“ gesetzt. Dieses kann gemacht werden, da alle Klassen und Schlüsselworte neu aufgenommen worden sind und somit noch kein Dokument diesen Klassen zugeordnet worden sein kann.

Wurde der XML Parser aber aus dem Taxonomie Update aufgerufen, so muss evtl. noch die Signifikanz der Schlüsselworte errechnet werden, die einer Klasse neu hinzugefügt wurden. Dieses geschieht beispielsweise durch einen nachträglichen Eintrag in der XML-Taxonomie-Datei der Tags <*taxon*> oder <*keyword*>. Dabei muss jedoch die Struktur aus Abschnitt 6.5 der Taxonomie Datei eingehalten werden.

Welche Klassen ein neues Schlüsselwort bekommen haben, wurde durch das Attribut „*newkeyword*“ in der Tabelle „*ACMCLASS*“ mit „*1*“ markiert. Dieser Eintrag ist jedoch nur notwendig, wenn wie beschrieben neue Schlüsselworte, bzw. neue Klassen der Taxonomie hinzugefügt wurden. Wurde eine neue Klasse hinzugefügt, interessiert diese nur, wenn sie auch ein Schlüsselwort besitzt. Ist dieses der Fall, so wurde in dieser neu hinzugefügten Klasse auch das „*newkeyword*“ Attribut auf „*1*“ gesetzt.

Ist bei einer Klasse nun das Attribut „*newkeyword*“ auf „*1*“ gesetzt, wird jetzt geprüft, ob der betroffenen Klasse auch schon Dokumente zugeordnet wurden. Ist dieses der Fall, muss die Signifikanz aller dieser Klasse angehörenden Schlüsselworte neu berechnet werden. Realisiert wird dieses durch den Aufruf des Servlet „*DocumentControllerServlet.java*“. Hat die Klasse noch keine zugeordneten Dokumente, so wird das Attribut „*newkeyword*“ ohne weitere Schritte auf „*0*“ gesetzt.

7.3.2 Servlets

Der Java-Servlets-Kern besteht aus zwei Servlets, die im Folgenden kurz beschrieben werden sollen, nämlich dem „*DocumentControllerServlet.java*“ und dem „*HashValueServlet.java*“.

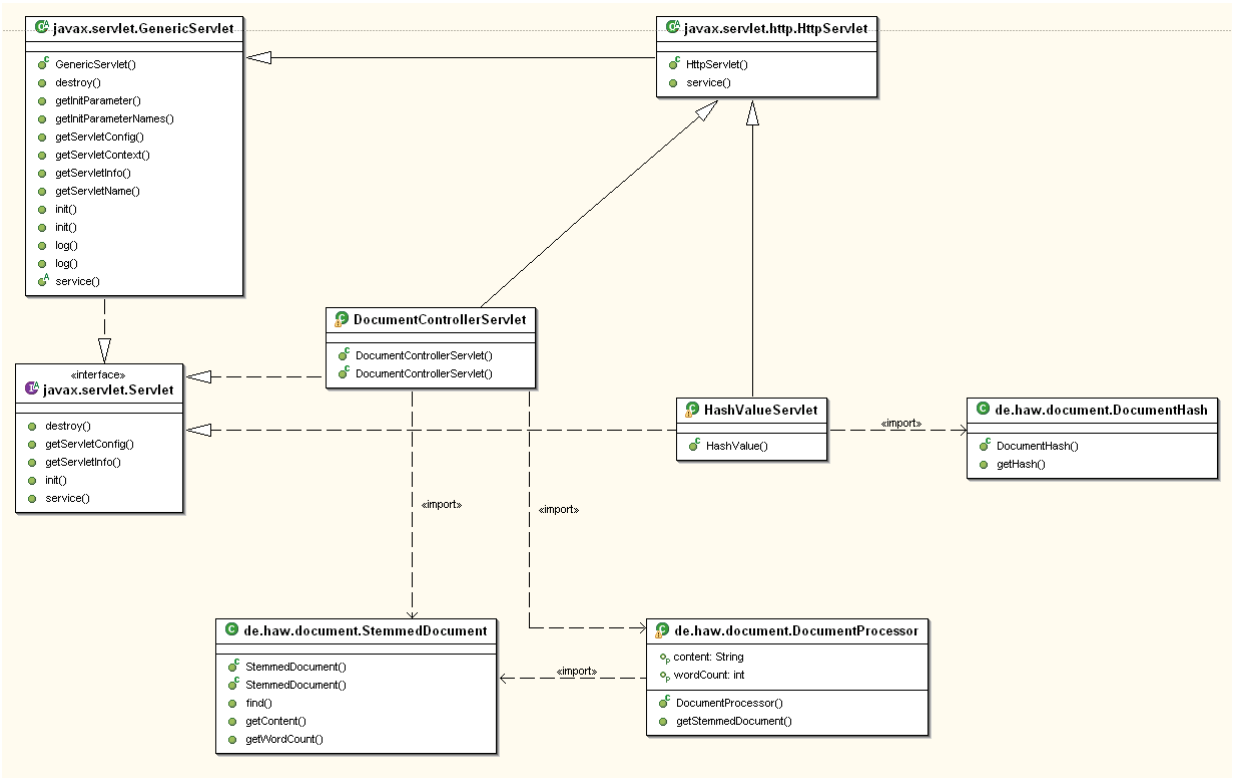

Abbildung 7-3 Klassendiagramm der verwendeten Servlets

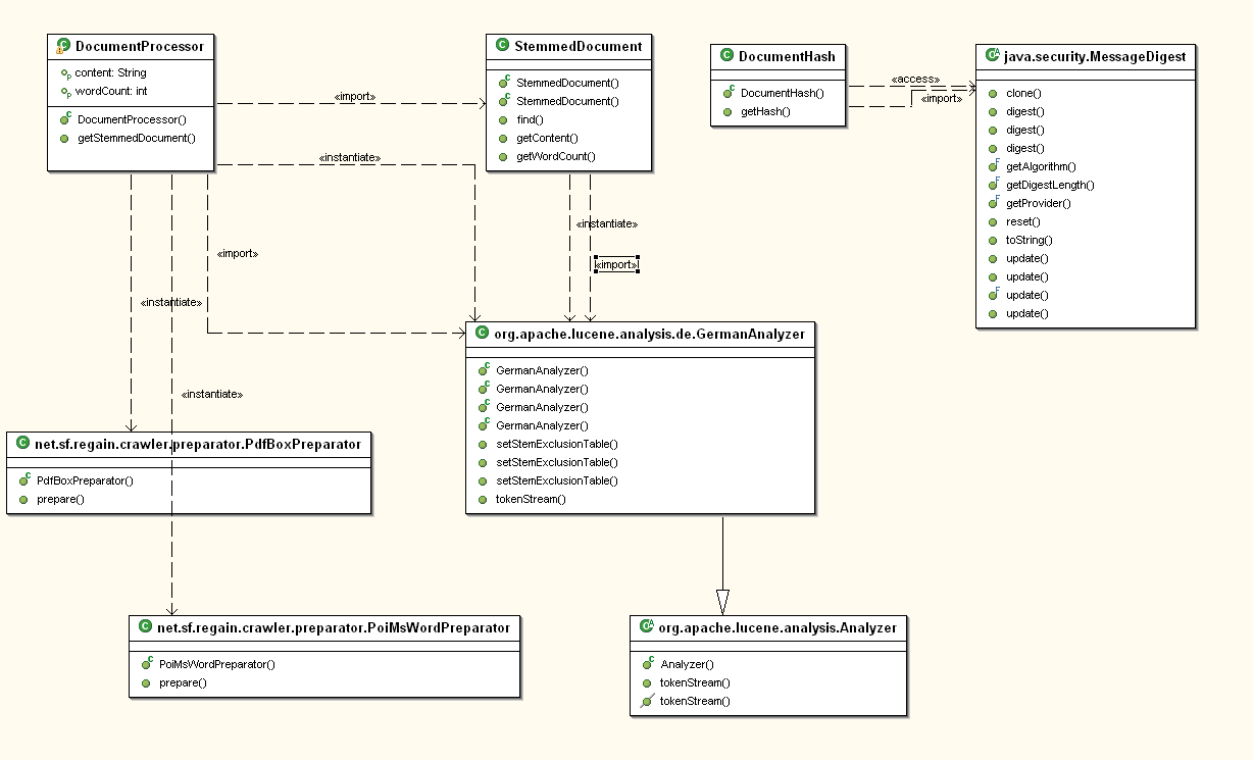

Abbildung 7-4 Klassendiagramm der Hilfsklassen für die verwendeten Servlets

7.3.2.1 „*DocumentControllerServlet.java*“

Dieses Servlet kann auf zwei Arten aufgerufen werden. Zum einen mit den Parametern „*doc_id*“, dem „*Projektnamen*“ und der „*acmclass_id*“. Die andere Möglichkeit ist der Aufruf ohne „*doc_id*“.

Der erste Fall ist für die Analyse eines Dokuments mit der ID „*doc_id*“. Bei der zweiten Variante werden alle Dokumente der Klasse mit der ID „*acmclass_id*“ neu ausgewertet. Auf diesem Weg wird die Signifikanz der einzelnen Schlüsselworte der Klasse „*acmclass_id*“ neu berechnet. Dieser Aufruf ist für die Funktionen des Taxonomie Updates (siehe Abschnitt 7.4.2) und dem Löschen eines Dokuments aus einer Klasse.

Dieses ist auch die erste Prüfung, die das Servlet vornimmt. Wird eine „*doc_id*“ übergeben, werden die Attribute dieses Dokuments ausgelesen und es erfolgt eine zweite Prüfung bezüglich des Analysestatus des Dokuments. Steht der Wert des Attributes „*analyse*“ auf „*1*“, so wurde dieses Dokument schon analysiert und alle erforderlichen Daten sind in den Attributen dieses Dokuments gespeichert.

Ist der Wert „*0*“, wurde das Dokument noch nicht analysiert. Dieses wird jetzt durch die folgenden Schritte realisiert. Der Pfad des Dokuments wird mittels des Attributes „*doc_pfad*“ an einen entsprechenden Präparator übergeben. (Siehe Abschnitt 6.2.3). Wird kein passender Präparator gefunden, so wird eine Exception ausgelöst.

Wird ein passender Präparator gefunden, so führt dieser eine Textnormalisierung durch, indem der Klartext aus dem übergebenden Dokument extrahiert wird. Hier werden unter anderem, wie in Abschnitt 4.2.2.1 beschrieben, die jeweiligen Formatierungs-Tags der entsprechenden Formate entfernt.

Der so gewonnene Klartext wird jetzt an die Textanalyseklassen von Jakarta Lucene übergeben (siehe auch 6.2.2).

Da der *Analyzer* von Lucene immer einen String oder einen Reader mit reinem Text erwartet, wird ihm der Klartext aus dem eben beschriebenen Vorgang übergeben. Das bedeutet, der Präparator wird als eine Art File-Konverter dem *Analyzer* vorgeschaltet, so dass immer der reine Klartext aus dem entsprechenden Dokument übergeben wird.

Der *Analyzer* besteht wiederum aus einem *Tokenizer* und weiteren Filtern. Der *Tokenizer* extrahiert nun die eigentlichen Wörter aus dem Klartext, indem er alle Zeichen, die nicht von Whitespaces unterbrochen sind, als ein Wort betrachtet. Die so extrahierten Worte bzw. „*Tokens*“ werden nun noch von dem *SimpleAnalyzer* (er wandelt alle Wörter in Kleinbuchstaben um), dem *StopAnalyzer* (er filtert alle Worte heraus, die er aus einer Stoppwortliste bekommt) und dem *StandardAnalyzer* (er entfernt Apostrophe und Punkte aus den Wörtern) bearbeitet. (siehe auch Abschnitt 6.2.2)

Als nächstes werden diese schon gefilterten Worte mit dem im Paket enthaltenen *GermanStemmer* bearbeitet. Seine Aufgabe ist es die Wörter in ihre Stammform zurück zu führen.

Die so gewonnenen isolierten Worte werden jetzt wieder zu einem String in der aufgefundenen Reihenfolge zusammengefügt und so als gestemmter Klartext im

Feld „*stemmedCont*“ der Tabelle „*DOC*“ gespeichert. Der Text wird gespeichert, da das Dokument so nicht immer erneut analysiert werden muss, wenn beispielsweise ein Taxonomie Update durchgeführt wird. Auf diese Weise können die entsprechenden Schlüsselworte, bei einer erneut erforderlichen Analyse, einfach in dem gestemmten Klartext, des in der Tabelle „*DOC*“ abgelegten Inhaltes, gesucht werden. Dieses führt zu einer enormen Zeitgewinnung, vor allem bei größeren Dokumenten.

Die gefundene Wortanzahl des gestemmten Klartextes wird im gleichen Schritt in dem Feld „*wortAnzahl*“ der Tabelle „*DOC*“ gespeichert. Diese Anzahl wird unter anderem später zur Signifikanzerrechnung benötigt (siehe Abschnitt 5.3). Auch das Analyse-Flag im Feld „*analyse*“ wird in diesem Schritt auf „*1*“ gesetzt. Dieser Wert sagt nun über das Dokument aus, dass es erfolgreich analysiert wurde. Dieser Wert ist, wie oben zu lesen, sehr wichtig für die Bearbeitungsweise des Servlets. Ist der Wert auf „*1*“ gesetzt, so muss bei erneutem Aufruf dieses Dokuments nicht erst erneut der Klartext extrahiert werden. Der gestemmte Klartext kann einfach aus dem Attribut „*stemmedCont*“ mittels der „*doc_id*“ aus der Tabelle „*DOC*“ ausgelesen werden.

Nach dieser Analyse wird die Verbindung des Dokuments, mit der aus der Klassifizierungsmaske übermittelten Klasse, gespeichert. Existiert diese Verbindung schon als Eintrag in der Tabelle „*DOC_ACMCLASS*“, so wird eine Exception ausgelöst. Existiert diese Verbindung noch nicht, wird sie mittels SQL-Insert Befehl in der Tabelle eingefügt.

Daraufhin wird zunächst die betroffene Klasse (Zur Erinnerung: das ist diejenige, welche über die Klassifizierungsmaske ausgewählt und übermittelt wurde) per SQL-Statement aus der Tabelle „*ACMCLASS*“ ausgelesen. Hier wird das Feld „*wortAnzInClass*“ angepasst. Darin steht die Anzahl aller Wörter der Dokumente, die dieser Klasse zugeordnet worden sind. Dieser Wert wird jetzt genommen und um die Gesamtwortanzahl des neu eingeordneten Dokumentes erhöht. Hierbei handelt es sich um den Wert, der eben in das Feld „*wortAnzahl*“ der Tabelle „*DOC*“ geschrieben wurde. Nach der Addition der beiden Werte wird der neue Wert wieder in das Feld „*wortAnzInClass*“ zurück geschrieben und gespeichert.

Um nun die betroffenen Schlüsselworte der Klasse zu finden, in die das Dokument hinzugefügt wurde, werden nun zunächst alle Schlüsselworte der übergebenen Klasse in der Tabelle „*ACMCLASS_KEYWORD*“ gesucht. So kann für jedes gefundene Schlüsselwort das Gesamtvorkommen in dem gestemmten Klartext ermittelt werden. Da der Name des Schlüsselwortes in dem Attribut „*name*“ der Tabelle „*ACMCLASS*“ steht, muss dieser zunächst über einen *INNER JOIN* mit der Tabelle „*ACMCLASS_KEYWORD*“ ermittelt werden. Der so gefundene Name wird als nächstes, wie in dem oben beschrieben Vorgang, „*gestemmt*“. Das ist notwendig, weil mit diesem Wort in dem gestemmten Gesamtinhalt des Dokumentes gesucht werden soll. Würde das Schlüsselwort nicht gestemmt werden, würde es nicht im dem gestemmten Inhalt gefunden werden können, da jetzt mittels Übereinstimmung der Stringkette (das Schlüsselwort) und dem Inhalt das Vorkommen dieses Schlüsselwortes ermittelt wird.

Wird eine Übereinstimmung zwischen gestemmtem Inhalt und eines der Klasse hinterlegtem Schlüsselwort gefunden, so wird dieses Schlüsselwort mit den folgenden Attributen in die Tabelle „*doc_acmclass*“ eingetragen. Neben dem Attribut „*doc_id*“ und „*acmclass_id*“ werden zwei weitere Attribute gespeichert. Die Anzahl der so gefundenen Übereinstimmungen zwischen gestemmtem Schlüsselwort und gestemmtem Inhalt des Dokuments wird in das Attribut „*anzahl*“ gespeichert. Die sich für dieses Schlüsselwort, auf dieses Dokument bezogene Signifikanz wird mittels Division der gefundenen Schlüsselwortanzahl „*anzahl*“ durch die im Dokument vorkommende Gesamtwortanzahl „*wortAnzahl*“ gebildet. (siehe 5.3) Dieser Wert wird nun in das Attribut „*signifikanz*“ der Tabelle „*doc_keyword*“ gespeichert.

Durch den Eintrag verändert sich auch die Signifikanz des Schlüsselwortes innerhalb seiner beschreibenden Klasse. Diese muss deswegen durch die folgenden Schritte angepasst werden. Sie steht in dem Attribut „*signifikanz*“ der Tabelle „*acmclass*“. Wie die Signifikanz errechnet wird, entnehmen sie bitte dem Abschnitt 5.3. Da sie zum einen aus der Summe aller einzelnen Signifikanzen des Schlüsselwortes dieser Klasse angehörigen Dokumente errechnet wird, müssen zunächst die einzelnen Signifikanzen aus der Tabelle „*DOC_ACMCLASS*“ ausgelesen werden. Realisiert wird dieses durch einen *INNER JOIN* über die Tabellen „*doc_acmclass*“ und „*DOC*“.

Das Attribut „*signifikanz*“ der gefundenen Ergebnisse wird in einem nächsten Schritt summiert. Diese Summe wird daraufhin durch die Anzahl der Klasse zugeteilten Dokumente dividiert. Diese neu errechnete Signifikanz wird nun im Attribut „*signifikanz*“ der Tabelle „*ACMCLASS_KEYWORD*“ gespeichert, indem der alte Wert überschrieben wird.

Die eingangs erwähnte zweite Möglichkeit dieses Servlet ohne den „*doc_id*“-Parameter aufzurufen, ist für den Fall eines Taxonomie Updates und dem Löschen eines Dokuments aus einer Klasse gedacht (siehe Abschnitt 7.4.2).

Bei diesem Update müssen für alle Klassen, denen neue Schlüsselworte hinzugefügt wurden, die Signifikanzen neu berechnet werden. Dazu werden mit der übermittelten „*acmclass_id*“ alle betroffenen Dokumente der jeweiligen Klasse mittels eines INNER JOIN der Tabellen „*acmclass*“ und „*doc_acmclass*“ ausgelesen. Alle so gefundenen Dokumente werden nun einzeln auf ihre Schlüsselworte hin analysiert. Dieses geschieht durch den oben beschriebenen Prozess. Die bei diesem Aufruf fehlende „*doc_id*“ wird sozusagen, durch die aus dem *INNER JOIN* gefundenen „doc_id“’s, ergänzt. Mit jeder gefunden „*doc_id*“ wird der oben beschriebene Vorgang wiederholt.

7.3.2.2 „*HashValueServlet.java*“

Dieses Servlet ist für die Bildung des Hash-Wertes des gesamten Dokuments verantwortlich. Dazu wird dem Servlet der Pfad des betroffenen Dokuments übergeben; dann liefert das Servlet den Hash-Wert des übermittelten Dokuments wieder zurück, indem alle eingelesenen Bytes des Dokuments komplett dem md5-Algorithmus von Java übergeben werden und so der Hash-Wert errechnet werden kann. (siehe auch Abschnitt 5.2.1)

Da es sich aber bei dieser Webapplikation nicht um kritische Bereiche oder Daten handelt, wird das Dokument mit nur einer Hashfunktion berechnet und das Risiko einer möglichen Kollision eingegangen (siehe Abschnitt 5.2.1). Dieser Fall würde bei der Applikation dann eintreten, wenn ein schon analysiertes Dokument durch ein neues Dokument mit identischem Hash-Wert zum System hinzugefügt werden würde. In diesem Fall würden die Ergebnisse minimal bei den betroffenen Klassen

verfälscht werden, welches aber auf die Gesamtfunktionalität des Systems keine großen Auswirkungen hat.

7.4 Funktionen des Rechenkerns

In den folgenden Abschnitten sollen die einzelnen Funktionen, welche über das Userinterface aufrufbar oder zu starten sind, näher erklärt, sowie die beteiligten Komponenten aufgezählt werden.

7.4.1 Neues Projekt Anlegen

Für das Anlegen eines neuen Projektes sind die Komponenten „*create_db.php*“, „*XML-Parser.php*“, „*db_function.php*“ und „*file_function.php*“ verantwortlich.
Es werden mittels der Funktionen aus der „*file_function.php*“ zunächst die Projekt-Verzeichnisse mit dem Namen des Projektes und den Unterverzeichnissen „*Dokumente*“ und „*XML-Dateien*“ angelegt (siehe Abschnitt 6.4). Daraufhin wird das Projekt in der Datenbank „*admin_db*“ in der Tabelle „*Projekt*“ mittels der Insert-Funktion aus der „*db_function.php*“ und mit den aus den Parametern übermittelten Werten gespeichert. Datum und Taxonomie-Update werden automatisch gesetzt. War der Eintrag erfolgreich wird jetzt eine Projekt-Datenbank entsprechend seines Namens angelegt und es werden die folgenden Projekt-Datenbank-Tabellen erzeugt:

- KEYWORD (Schlüsselwörter)
- ACMCLASS (Taxonomie-Klassen)
- KEYWORD_ACMCLASS (Verbindung zwischen Schlüsselwort, Klasse und Lexikon)
- GOOGLE_SEARCH (temporäre Ergebnisse aus der Suchanfrage)
- DOC_SA (Dokument-Eigenschaften aus der Suchanfrage)
- DOC (Dokumente)
- DOC_ACMCLASS (Verbindung zwischen Dokument und Klasse)
- LEXIKON (Alle Worte aus der Dokumentanalyse)
- DOC_LEXIKON (Verbindung zwischen Dokument und jedem Wort aus dem Lexikon)

Den genauen Aufbau, sowie alle Attribute sind in der Konzeption der Datenbank zu finden. (siehe Abschnitt 6.3)

Wurden die Tabellen erfolgreich erstellt, wird im nächsten Schritt die angegebene XML-Datei mittels der Funktionen aus der „*file_function.php*“ in das angelegte „*XML-Dateien*“ Verzeichnis kopiert und gespeichert. War der Vorgang erfolgreich, wird nun der „*XML-Parser*“ aufgerufen und gestartet. Dieser analysiert nun die kopierte XML-Datei. Auf diese Weise werden die Tabellen „*KEYWORD*“, „*ACMCLASS*“ und „*KEYWORD_ACMCLASS*“ mit den Werten aus der XML-Datei gefüllt. Der genaue Ablauf, sowie die Funktionsweise sind im Abschnitt 7.3.1.4 des XML-Parsers zu finden.

7.4.2 Taxonomie-Update durchführen, bzw. eine Klasse oder ein Schlüsselwort der Taxonomie hinzufügen

Für das Taxonomie-Update sind die Komponenten „*XML-Parser.php*“, „*db_function.php*“, „*file_function.php*“ und „*DocumentControllerServlet.java*“ verantwortlich.
War die Überprüfung der XML-Datei positiv, startet der Update-Prozess mit dem Kopieren mittels der „*file_function.php*“ Funktion die neue, angegebene XML-Datei in das „*XML-Dateien*“ Verzeichnis. Nach erfolgreicher Durchführung wird nun erneut der „*XML-Parser.php*“ aufgerufen. Dieser überprüft daraufhin die kopierte Datei auf neue Klassen und Schlüsselworte. Der genaue Ablauf, sowie die Funktionsweise sind im Abschnitt 7.3.1.4 des XML-Parsers zu finden.

Wenn der XML-Parser seine Arbeit beendet hat und alle Klassen mit neu hinzugefügten Schlüsselworten im Attribut „*signifikanz*“ der Tabelle „*ACMCLASS_KEYWORD*“ markiert hat, werden für diese Klassen je einzeln der „*DocumentControllerServlet.java*“ aufgerufen. Das Servlet wird ohne eine „*doc_id*“ aufgerufen, was für das Servlet ein Indiz dafür ist, dass für diese Klasse alle Dokumente neu ausgewertet werden müssen. Die genaue Funktionalität ist im Abschnitt 7.3.2.1 dieses Servlets zu lesen.

7.4.3 Löschen eines Schlüsselwortes

Für das Taxonomie-Update ist die Komponente „*db_function.php*“ verantwortlich.
Beim Löschen des Schlüsselwortes wird zunächst nur der Eintrag der Beziehung zwischen dem Schlüsselwort und der gewählten Klasse in der Tabelle

„*ACMCLASS_KEYWORD*“ gelöscht. Nun kann es noch sein, dass das Schlüsselwort einer anderen Klasse zugeordnet ist. Um dieses herauszufinden wird ein SQL-Statement benutzt, welches die Anzahl des weiteren Vorkommens des Schlüsselwortes in der Tabelle „*ACMCLASS_KEYWORD*“ überprüft. Wird keine weitere Klassenzugehörigkeit gefunden dieses Schlüsselwortes gefunden, wird jetzt auch das Schlüsselwort selbst aus der Tabelle „*KEYWORD*“ gelöscht.

7.4.4 Dokument per manueller Klassifizierung einer Klasse hinzufügen

Beim Aktivieren des Buttons „*Dieses Dokument für die ausgewählte Klasse übernehmen*“ in der Klassifizierungsmaske werden mehrere Prozesse gestartet, die notwendig sind, um das Dokument erfolgreich in das System zu übernehmen.

7.4.4.1 Hash-Wert-Errechnung und Prüfung auf Existenz

Zunächst wird der Hash-Wert mittels des „*HashValueServlet.java*“-Servlets berechnet. Anhand des berechneten Hash-Wertes wird jetzt geprüft, ob sich das Dokument schon im System befindet. Dieses ist notwendig, um die späteren Ergebnisse, bzw. die Werte zur Errechnung der Signifikanz nicht unnötig zu verfälschen (siehe Abschnitt 5.1.1 und 5.1.2). Würde ein Dokument doppelt in das System aufgenommen werden, so würden die Werte dieses Dokuments, die Wortanzahl sowie die Anzahl der betroffenen Schlüsselworte doppelt in die Bewertung eingehen und so die statistische Analyse verfälschen.

Zur Prüfung werden nun schon alle im Dokument enthaltenen Dokumente aus der Tabelle „*DOC*“ mit Hilfe eines SQL-Statements herausgesucht, deren Hash-Wert mit dem neu errechneten Hash-Wert übereinstimmen. Liefert das SQL-Statement einen Treffer zurück, existiert dieses Dokument schon in dem System und muss nicht neu analysiert werden. Ist dieses der Fall muss jetzt überprüft werden, ob sich dieses Dokument schon in der zugeordneten Klasse befindet. Dieses wird mit einer SQL-Abfrage auf die Tabelle „*DOC_ACMCLASS*“ realisiert. Wird kein Treffer zurückgeliefert, existiert dieses Dokument noch nicht im System und es muss neu analysiert werden.

7.4.4.2 Verzeichnisse anlegen & Dokument kopieren

Um das Dokument richtig in der Dokumentensammlung bzw. in der Verzeichnisstruktur abzulegen (siehe 6.4), wird zunächst geprüft, ob für die aus der Klassifizierungsmaske übermittelte Klasse schon ein entsprechendes Verzeichnis im Unterverzeichnis „Dokumente“ des Projektverzeichnisses existiert. Ist dieses nicht der Fall, so wird ein Verzeichnis mittels der „*file_function.php*“ mit dem Namen und dem Label der Klasse erstellt. Existiert dieses Verzeichnis, wird das Dokument aus dem „*temp*“-Verzeichnis in dieses Verzeichnis kopiert und aus dem „*temp*“-Verzeichnis gelöscht.

7.4.4.3 Ggf. Search Attribute speichern

Wurde das Dokument über die Websuche gefunden, so besitzt dieses Dokument einen Eintrag in der Tabelle „*GOOGLE_SEARCH*“. Ist dieses der Fall, werden die schon oben erwähnten Such-Attribute dieses Dokuments aus der Tabelle gelesen und mittels der „*db_function.php*“ in die Tabelle „*DOC_SA*“ kopiert.

Wurde das Dokument über das Dateisystem gefunden, so hat es keine Such-Attribute und die Variable „*DOC_SA_ID*“ wird mit „*NULL*“ belegt.

7.4.4.4 Dokument speichern

Jetzt muss das Dokument selbst gespeichert werden. Dieses passiert durch einen Datenbank-Eintrag mittels der Insert-Funktion in der Tabelle „*DOC*“ mit den folgenden Attributen:

- der „*DOC_SA_ID*“ aus dem obigen Schritt
- dem Dokumentnamen – er wird aus dem Dokument-Pfad abgeleitet
- dem Dokument-Pfad – der Pfad zum Dokument im erstellten Dokumentverzeichnis
- der Wortanzahl des Dokuments – sie wird zunächst mit „*0*“ belegt, da diese Anzahl noch unbekannt ist
- dem Hash-Wert des Dokuments aus dem obigen Schritt
- dem Analyse-Flag – er wird zunächst auf „*0*“ gesetzt, da das Dokument noch nicht analysiert wurde
- dem Datum des Zeitpunktes, wo das Dokument eingetragen wird

7.4.4.5 Textanalyse

Wurde das Dokument erfolgreich in die Tabelle „*DOC*“ eingetragen, kann es nun analysiert werden. Für diese Aufgabe ist das Servlet „*DocumentControllerServlet.java*“ zuständig. Ihm wird die ID des Dokuments, bestehend aus der Variablen „*doc_id*“, der aktuelle Projektname zur Auswahl der richtigen Datenbank und die Klasse, bestehend aus der „*acmclass_id*“ der Klasse, in die das Dokument hinzugefügt werden soll, übermittelt.

Dieses Servlet ist dann für die Zählung der aus dem Dokument extrahierten betroffenen Schlüsselworte zuständig. Außerdem berechnet es die Signifikanz und speichert diese in dem Attribut „*signifikanz*“ der Tabelle „*ACMCLASS_KEYWORD*“. Die genaue Funktionalität des Servlets ist im Abschnitt 7.3.2.1 beschrieben.

7.4.4.6 Klassenzugehörigkeit des Dokumentes speichern

Nach der Analyse und wird auch die Verbindung des Dokuments mit der aus der Klassifizierungsmaske übermittelten Klasse gespeichert. Dieses geschieht über einen SQL-Insert Befehl in der Tabelle „*DOC_ACMCLASS*“. Auch dieser Schritt wird von dem Servlet „*DocumentControllerServlet.java*“ (siehe Abschnitt 7.3.2.1) übernommen.

7.4.4.7 Signifikanz anpassen

Auch dieser Schritt wird innerhalb des Aufrufes des Servlets „*DocumentControllerServlet.java*“ (siehe Abschnitt 7.3.2.1) realisiert. Wurde die Klassenzugehörigkeit des Dokuments erfolgreich gespeichert, müssen nun alle Werte der betroffenen Klasse und ihren Schlüsselworten angepasst und ihre Signifikanz neu berechnet werden.

Dazu wird zunächst die betroffene Klasse (das ist diejenige, welche über die Klassifizierungsmaske ausgewählt und übermittelt wurde) per SQL-Statement aus der Tabelle „*ACMCLASS*“ herausgesucht. Hier muss das Feld „*wortAnzInClass*“ angepasst werden. In im stehen die Anzahl aller Wörter, die dieser Klasse zugeordnet worden sind. Hierbei handelt es sich nicht um die Anzahl der Schlüsselworte dieser Klasse, sondern der Gesamtsumme der Gesamtanzahl der Worte aller Dokumente, die dieser Klasse angehören. Dieser Wert wird jetzt

genommen und um die Gesamtwortanzahl des neu eingeordneten Dokumentes erhöht. Hierbei handelt es sich um den Wert, der eben in das Feld „*wortAnzahl*“ der Tabelle „*DOC*“ geschrieben wurde. Nach der Addition der beiden Werte wird der neue Wert wieder in das Feld „*wortAnzInClass*“ zurück geschrieben und gespeichert.
Um nun die betroffenen Schlüsselworte der Klasse zu finden, in die das Dokument hinzugefügt wurde, werden nun zunächst alle Schlüsselworte der übergebenen Klasse in der Tabelle „*ACMCLASS_KEYWORD*“ gesucht, um dann für jedes gefundene Schlüsselwort das Gesamtvorkommen dieses Schlüsselwortes in dem gestemmten Klartext zu ermitteln. Die so gefundene Anzahl des Vorkommens des Schlüsselwortes in dem Dokument wird dann bei der Anpassung des Attributes „*keyAnzInClass*“ und „*signifikanz*“ verwendet. In dem Attribut „*keyAnzInClass*“ handelt es sich um die Anzahl des Vorkommens dieses Schlüsselwortes in den bislang dieser Klasse zugeordneten Dokumenten. Bei dem Attribut „*signifikanz*“ steht die Signifikanz des Schlüsselwortes innerhalb der betroffenen Klasse. Um das Attribut „*keyAnzInClass*“ nun zu ändern, wird der alte Wert des Feldes „*keyAnzInClass*“ mit dem eben ermittelten Wert des Schlüsselwortvorkommens addiert und wieder in das Feld zurück geschrieben. Die Signifikanz im Feld „*signifikanz*“ wird durch den neuen, eben gespeicherten Wert „*keyAnzInClass*“ und den oben erwähnten neuen Wert „*wortAnzInClass*“ der Klasse neu berechnet und wieder in das Feld „*signifikanz*“ geschrieben.

7.4.5 Signifikanzvektor einer Klasse

Um den Signifikanzvektor auszugeben wird mittels einem „*INNER JOIN*“- SQL-Statement über die Tabellen „*ACMCLASS_KEYWORD*“ und „*KEYWORD*“ alle dieser Klasse zugehörigen Schlüsselworte gesucht und ausgegeben. Die Namen der Schlüsselworte stehen im Attribut „*name*“ der Tabelle „*KEYWORD*“. Die Signifikanz dieses Schlüsselwortes steht in der Tabelle „*ACMCLASS_KEYWORD*“. Das gemeinsame Attribut der beiden Tabellen, welches für den *INNER JOIN* gebraucht wird ist „*keyword_id*“. Dieses sind nach Abschnitt 5.3 alle notwendigen Daten, um den Signifikanzvektor der ausgewählten Klasse abbilden zu können.

8 Kapitel Implementierung & Testdurchlauf

Im Folgenden soll die bisher realisierte Webapplikation in der Praxis angewendet werden. Dazu wird eine Klasse des Zweiges D.4. „*Betriebssysteme*“ der ACM CSS Taxonomie mit Dokumenten befüllt werden. Um passende Dokumente für die jeweiligen Klassen zu finden, werden die benötigten, manuell zu klassifizierende Dokumente über das Google-API gesucht. Daraufhin werden die entsprechenden Schlüsselworte jeder betroffenen Klasse aus den Klartexten der Dokumentinhalte extrahiert, gezählt und somit für diese ACM CSS Klasse ein Signifikanzvektor gebildet.

Neben der Geschwindigkeit des Systems sind die zu erwartenden Ergebnisse der Signifikanz jedes einzelnen Schlüsselwortes bzw. die Entwicklung des Ergebnisses mit steigender Anzahl von dieser Klasse hinein klassifizierten Dokumenten von Interesse. Nach dem starken Gesetz der großen Zahlen (siehe Anschnitt 5.3) ist mit zunehmender Dokumentanzahl eine Konvergenz der einzelnen Signifikanzen zu erwarten.

8.1 Voraussetzungen für den Test

Bevor auf die Durchführung der Tests eingegangen wird, sollen seine technischen Grundlagen erläutert werden, um die erzielten Ergebnisse besser einschätzen zu können. Zudem werden die zur Verfügung gestellten Daten beschrieben, die für die Bewertung genutzt werden.

8.1.1 Eingesetzte Hard- und Software

Die Testumgebung wird unter *Microsoft Windows XP Home Edition Version 2002* mit dem *Service Pack 2* auf einem PC mit *Intel Pentium IV* Prozessor mit *3,2 GHz* und *1 GB* Hauptspeicher simuliert. Um die PHP-Skripte und die Java-Servlets erfolgreich ausführen zu können, sind ein *Apache-Webserver*, ein *Apache-Tomcat-Webserver*, sowie eine *MySQL-Datenbank* erforderlich. Hierfür wurde die Distribution *XAMPP 1.4.13* von apachefriends.org[23] installiert, in der unter anderem die Version von *MySQL 4.1.13* und *Apache 2.0.53* enthalten sind. Für den *Tomcat-Server* wurde die aktuelle Version *5.5* von *Apache* und das *Sun J2SDK 1.5* für *Windows* benutzt.

[23] http://www.apachefriends.org

Weiterhin ist eine funktionierende Internetverbindung erforderlich, wenn man die Suchfunktion mittels dem Google-API der Webapplikation benutzen möchte.

8.1.2 Datenmaterial zur Klassifikation

Für die zur Erstellung des Taxonomiebaumes mit seinen Klassen und diesen beschreibenden Schlüsselworten benötigte XML-Taxonomie-Datei wurde die im Rahmen des HyLOS Projektes erstellte ACM CSS Taxonomie verwendet. Diese Datei enthielt zum Zeitpunkt des Testes 1767 Klassen mit insgesamt 3514 hinterlegten Schlüsselworten, von denen 1603 unterschiedliche Schlüsselworte sind und die restlichen doppelt oder mehrfach verwendet wurden. Die über die Klassifizierungsmaske manuell zu klassifizierenden Dokumente wurden mit Hilfe der Taxonomieklassen hinterlegten Schlüsselworte gesucht. Bei den Dokumenten handelte es sich die Formate Word und PDF. Die Größe der Dokumente lag dabei zwischen 110 KB und 2 730 KB.

8.2 Ergebnisse des Tests

Nachdem das Testsystem sowie die Daten für die Auswertung vorgestellt wurden, sollen in den nächsten Abschnitten die mit ihnen erzielten Ergebnisse erläutert werden.

Um die Korrektheit der Funktionalität des verwendeten Algorithmus zur Berechnung der Signifikanz über die relative Häufigkeit zu testen, wurden zunächst zwei kleine Word Testdokumente erstellt. Daraufhin wurden diese einer Testklasse der Taxonomie zugeteilt und analysiert. Gleichzeitig ist dieses ein Funktionstest für die verwendete Textanalyse und der darauf aufbauenden Schlüsselwortsuche in dem Inhalt des analysierten Dokuments.

8.2.1 Funktionalitätstest des verwendeten Algorithmus und der Schlüsselwortsuche

Die verwendete Testklasse besteht aus den folgenden Schlüsselworten aus Tabelle 8.1:

Schlüsselworte der Testklasse
Data Mining
Test
Diplom
Taxonomie
Schlüsselwort
Klassifikation

Tabelle 8-1 Schlüsselworte einer Testklasse

Nun wurden zwei Dokumente A und B jeweils auf die Häufigkeit der der Testklasse hinterlegten Schlüsselworte untersucht. Auf der einen Seite wurden manuell „per Hand“ die Wörter in den Dokumenten gesucht und gezählt. Auf der anderen Seite wurde dieser Vorgang mittels der Analysefunktion der Webbapplikation gemacht. Die Ergebnisse sind in den folgenden Tabellen zu sehen.

In der Tabelle 8.3 werden die auch auf diese beiden Wege berechneten Signifikanzen innerhalb des jeweiligen Dokuments gezeigt. Um schließlich noch die Gesamtsignifikanz eines Schlüsselwortes in seiner Klasse zu errechen, müssen die einzelnen Signifikanzen aufsummiert und durch die Anzahl der Klasse zugeordneten Dokumente geteilt werden (siehe 5.3). Diese Ergebnisse sind in der Tabelle 8.4 zu sehen.

Schlüsselwort	Dokument A		Dokument B	
	„per Hand“ gezählt	Analyse per Webapplikation	„per Hand“ gezählt	Analyse per Webapplikation
Data Mining	4	4	4	4
Test	2	2	2	2
Diplom	7	7	3	3
Taxonomie	3	3	0	0
Schlüsselwort	0	0	0	0
Klassifikation	2	2	1	1
Gesamtwortanzahl	117		107	

Tabelle 8-2 Vergleich von manuellen und per Webappliaktion errechneten Häufigkeiten

Schlüsselwort	Dokument A		Dokument B	
	per Hand ausge-rechnete Signifikanz	Signifikanz der Webapp.	per Hand ausge-rechnete Signifikanz	Signifikanz der Webapp.
Data Mining	0.034188	0.034188	0.0373832	0.0373832
Test	0.017094	0.017094	0.0186916	0.0186916
Diplom	0.0598291	0.0598291	0.0280374	0.0280374
Taxonomie	0.025641	0.025641	0	0
Schlüsselwort	0	0	0	0
Klassifikation	0.017094	0.017094	0.00934579	0.00934579

Tabelle 8-3 Vergleich von manuellen und per Webappliaktion errechneten Signifikanzen

Schlüsselwort	∑ der einzelnen Signifikanzen aus Dokument A und B / Anzahl der Dokumente (nach Formel 5.4); „per Hand“ gerechnet	∑ der einzelnen Signifikanzen aus Dokument A und B / Anzahl der Dokumente (nach Formel 5.4); „per Webapplikation“ gerechnet
Data Mining	0.0357856	0.0357856
Test	0.0178928	0.0178928
Diplom	0.0439332	0.0439332
Taxonomie	0.0128205	0.0128205
Schlüsselwort	0	0
Klassifikation	0.0132199	0.0132199

Tabelle 8-4 Vergleich von manuellen und per Webappliaktion errechneten Gesamtsignifikanzen

8.2.2 Funktionalitätstest bezüglich des verwendeten Algorithmus

Schlüsselworte der Klasse D.4. - „Betriebssysteme - Allgemein“
BSD
Kern
Kerndienst
Kerndienstaufruf
Kernprogramm
Kommunikationsprimitive
Linux
Prozess
Prozeß
Cron
Betriebsmittelanforderung
Betriebsmittel Verwaltung
Betriebssystemkern
Betriebssystem
Endbenutzersystem

Tabelle 8-5 Schlüsselworte der Klasse D.4.0

Dokument	A	B	C	D	E	F
Format	PDF	PDF	PDF	PDF	PDF	PDF
Gesamtwortanzahl	76199	23997	192294	19878	711	5251
Dauer der Analyse in ms	23734	16984	28672	1812	1375	3250
BSD	0	13	0	0	0	8
Kern	1	48	0	0	0	21
Kerndienst	0	1	0	0	0	0
Kerndienstaufruf	0	0	0	0	0	0
Kernprogramm	0	0	0	0	0	0
Kommunikationsprimitive	0	1	0	0	0	0
Linux	0	0	0	0	0	6
Prozess	7	328	0	0	0	18
Prozeß	0	0	0	0	0	0
Cron	6	12	0	0	0	26
Betriebsmittelanforderung	5	2	0	0	0	0
Betriebsmittel Verwaltung	0	0	0	0	0	0
Betriebssystemkern	0	10	0	0	0	0
Betriebssystem	0	268	0	0	0	56
Endbenutzersystem	5	0	0	0	2	0

Tabelle 8-6 Details der Klasse D.4.0 zugeteilten Dokumente

Dokument	G	H	I	J	K
Format	PDF	PDF	PDF	PDF	PDF
Gesamtwortanzahl	0	33975	338	30612	4074
Dauer der Analyse in ms	0	18672	297	1469	2578
BSD	0	3	0	0	0
Kern	0	88	1	0	0
Kerndienst	0	0	0	0	0
Kerndienstaufruf	0	0	0	0	0
Kernprogramm	0	1	0	0	0
Kommunikationsprimitive	0	3	0	0	0
Linux	0	1	0	0	0
Prozess	0	655	8	0	65
Prozeß	0	0	0	0	0
Cron	0	1	0	0	0
Betriebsmittelanforderung	0	1	0	0	2
Betriebsmittel Verwaltung	0	0	0	0	0
Betriebssystemkern	0	1	0	0	0
Betriebssystem	0	168	7	0	2

Endbenutzersystem	0	0	0	0	0

Tabelle 8-7 Details der Klasse D.4.0 zugeteilten Dokumente

Dokument	A	B	C	D	E	F
BSD	0	0.000270867	0.000180578	0.000135434	0.000108347	0.000344209
Kern	1.31235e-005	0.00100669	0.000671124	0.000503343	0.000402675	0.0010021
Kerndienst	0	2.08359e-005	1.38906e-005	1.0418e-005	8.33437e-006	6.94531e-006
Kerndienst-aufruf	0	0	0	0	0	0
Kernprogramm	0	0	0	0	0	0
Kommunikations-primitive	0	2.08359e-005	1.38906e-005	1.0418e-005	8.33437e-006	6.94531e-006
Linux	0	0	0	0	0	0.00019044
Prozess	9.18647e-005	0.00688012	0.00458675	0.00344006	0.00275205	0.00286469
Prozeß	0	0	0	0	0	0
Cron	6.37412e-005	0.000289402	0.000192935	0.000144701	0.000115761	0.000921707
Betriebsmittel-anforderung	7.87412e-005	4.16719e-005	2.77813e-005	2.08359e-005	1.66687e-005	1.38906e-005
Betriebsmittel Verwaltung	0	0	0	0	0	0
Betriebs-systemkern	0	0.000208359	0.000138906	0.00010418	8.33438e-005	6.94531e-005
Betriebs-system	0	0.00558403	0.00372269	0.00279202	0.00223361	0.00363878
Endbenutzer-system	6.56177e-005	3.28088e-005	2.18726e-005	1.64044e-005	0.000575711	0.00047976

Tabelle 8-8 Veränderungen der Signifikanz der Klasse D.4.0 mit steigender Dokumentanzahl

Dokument	G	H	I	J	K
BSD	0.000344209	0.000269194	0.000239284	0.000215355	0.000195778
Kern	0.0010021	0.00107534	0.00128459	0.00115613	0.00105103
Kerndienst	6.94531e-006	5.20898e-006	4.63021e-006	4.16719e-006	3.78835e-006
Kerndienstaufruf	0	0	0	0	0
Kernprogramm	0	3.67918e-006	3.27038e-006	2.94334e-006	2.67576e-006
Kommunikationsprimitive	6.94531e-006	1.62465e-005	1.44413e-005	1.29972e-005	1.18156e-005
Linux	0.00019044	0.000146509	0.00013023	0.000117207	0.000106552
Prozess	0.00286469	0.00455838	0.00668174	0.00601357	0.00691732
Prozeß	0	0	0	0	0
Cron	0.000921707	0.000694959	0.000617742	0.000555968	0.000505425
Betriebsmittelanforderung	1.38906e-005	1.40971e-005	1.25308e-005	1.12777e-005	5.48814e-005
Betriebsmittel Verwaltung	0	0	0	0	0
Betriebssystemkern	6.94531e-005	5.5769e-005	4.95725e-005	4.46152e-005	4.05593e-005
Betriebssystem	0.00363878	0.00334719	0.0052764	0.00474876	0.00436168
Endbenutzersystem	0.000479776	0.00035982	0.00031984	0.000287856	0.000261687

Tabelle 8-9 Veränderungen der Signifikanz der Klasse D.4.0 mit steigender Dokumentanzahl

8.3 Zusammenfassung

Anhand des Testes aus Abschnitt 8.2.1 ist die korrekte Funktionsweise des programmierten Algorithmus gezeigt worden. Anhand der übereinstimmenden Ergebnisse der Webapplikation und der auf manuell nachgerechneten Kontrolle ist

bewiesen, dass die Umsetzung des Algorithmus nach Formel 5.4 korrekt implementiert wurde.

Die ermittelten Ergebnisse aus Tabelle 8.2 lassen dazu darauf schließen, dass auch die Schlüsselwortsuche korrekt funktioniert. Hierbei muss aber unbedingt darauf hingewiesen werden, dass diese sehr stark von der Textanalyse und den hiermit verbundenen Präparatoren abhängt. Die Schlüsselworte werden innerhalb des gestemmten Klartextes gesucht, welches durch einen Vergleich an Übereinstimmungen von Strings realisiert wird. Liefert der Präparator hierbei schon falsche Grundwerte bzw. einen inkorrekten Klartext, so können diese Schlüsselworte folglich auch nicht gefunden werden. Dieser Test bezüglich der Suchfunktion zeigt nur, dass diese bei korrekt gelieferten Grunddaten bzw. des Klartextes korrekt funktioniert. Werden hingegen schon falsche Grunddaten von den Präparatoren geliefert, so wird die Suchfunktion zwar richtig funktionieren, aber anhand der falschen Grunddaten evtl. nicht die gewünschten Ergebnisse liefern. Dieses wird unter anderem auch durch die Ergebnisse des Testes aus Abschnitt 8.2.2 belegt, die im Folgenden ausgewertet werden.

Betrachtet man die gewonnen Testdaten aus Tabelle 8.6 und 8.7 bezüglich der Performance, so weisen die gemessenen Daten auf eine gute Geschwindigkeit des Systems bzw. der Analyse hin. So braucht die Webapplikation beispielsweise für Dokument A mit einer Größe von 1009 KB eine Zeit von 23 734 ms und für ein kleineres Dokument wie E mit einer Größe von 261 KB 1375 ms. Dieses sind für den Benutzer akzeptable Wartezeiten. Hierbei muss natürlich auch erwähnt werden, dass die Analysezeit nicht nur von der reinen Dokumentgröße abhängt, sondern primär von der Anzahl der zu analysierender Worte.

Laut dem starken Gesetz der großen Zahlen (siehe 5.3) ist bei der Betrachtung bezüglich der Signifikanzentwicklung eine Konvergenz bei den Signifikanzen der einzelnen Schlüsselworte einer Klasse zu erwarten, desto mehr Dokumente dieser Klasse zugeordnet werden.

Betrachtet man nun die erhaltenen Werte aus Tabelle 8.6 und 8.7, so ist keine Konvergenz der Signifikanz bei den einzelnen Schlüsselworten zu erkennen. Dieses

Ergebnis lässt mehrere Schlussfolgerungen zu. Werden beispielsweise die Dokumente C, D, G und J betrachtet, so zeigen sich bei diesen Dokumenten keine Treffer bezüglich der in diesen Dokumenten vorkommenden Schlüsselworte. Diese Ergebnisse lassen wiederum auch auf mehrere Ursachen ihrer Entstehung schließen.

Die erste Möglichkeit wäre, dass der Analyse- bzw. die Suchfunktion nicht richtig funktioniert. Dieses wurde aber zum Teil durch den Test aus Abschnitt 8.2.1 widerlegt. Dagegen spricht auch die Tatsache, dass bei den anderen Dokumenten Schlüsselworte gefunden wurden. Es kann aber durchaus sein, dass das errechnete Ergebnis aufgrund des falsch gefilterten Klartextes ermittelt worden ist. Ein Indiz hierfür wäre eine relativ hohe oder niedrige Gesamtwortanzahl. Werden diese vier Dokumente genauer betrachtet, so stehen hier teilweise tatsächlich sehr hohe Werte. Bei Dokument C wurden insgesamt 192 294 Worte bei einer Dateigröße von 751 KB gefunden. In Dokument D wurden 19878 Worte bei einer Größe von 394 KB gefunden. In Dokument G wurde hingegen kein Wort gefunden und in Dokument J wiederum 30612 Worte bei einer Größe von 238 KB. Hieraus lässt sich schlussfolgern, dass die gefundene Gesamtwortanzahl im Verhältnis zur Größe tatsächlich ein Indiz für falsch gelieferte Grunddaten ist. Der Beweis hierfür liegt in der Datenbank. Da hier der gestemmte Klartext des analysierten Dokuments gespeichert wird, zeigt eine Analyse dieses Attributes in den betroffenen Dokumenten die Ursache dieses Fehlers. Hier stehen nämlich nur einzelne Buchstaben und nicht die gestemmten Worte, wie bei den anderen Dokumenten. Da die Suchfunktion zur Ermittlung der entsprechenden Schlüsselworte nun mittels Vergleich aus diesem gestemmten Klartext arbeitet, ist klar, dass in den gestemmten Klartexten der betroffenen Dokumente nichts gefunden werden kann.

Um herauszubekommen, wieso dieses nur bei diesen drei Dokumenten so ist, und nicht bei allen getesteten Dokumenten (es sind immerhin alles PDF Dokumente), bedarf es einer weiteren Analyse dieser Dokumente. Dazu wurden die einzelnen Dokumente geöffnet. Bei Dokument C waren bezüglich des Formates keine Besonderheiten zu erkennen. Bei Dokument D handelte es sich hingegen um eine in PDF konvertierte Power Point Präsentation, gleiches gilt für Dokument J. Dieses

deutet darauf hin, dass es anscheinend Probleme mit PDF Dokumenten gibt, die aus Power Point Präsentationen konvertiert wurden.

Aufgrund dieser Vermutung, wurden weitere Tests bezüglich dieser speziellen PDF Dateien gemacht. Dabei stellte sich heraus, dass es in der Tat überwiegend mit diesem speziellen PDF Probleme gibt. Hier kann nicht der korrekte Klartext gefiltert werden. Es stellte sich aber auch heraus, dass es nicht immer zu diesem Fehler kommt. Die Erklärung hierfür, sowie für die falsche Extraktion bei Dokument C ließ sich nicht klären, da es sich anscheinend nicht um eine Regelmäßigkeit handelt. Die Vermutung ist nur, dass es an einigen speziellen PDF Spezifikationen liegen muss, die aufgrund von irgendwelchen Konvertierungen einiger Programme entstehen, die das Konvertieren in ein PDF Format zur Verfügung stellen.

Wie schon erwähnt wurde bei dem Dokument D keine Wortanzahl gefunden. Dieses ist eigentlich ein Indiz dafür, dass dieses Dokumentformat nicht von der Applikation unterstützt wird. Da es sich aber auch um ein PDF Format handelt, kann dieses also ausgeschlossen werden. Aus diesem Grund wurde auch diese Datei näher untersucht und herausgefunden, dass es sich bei diesem Dokument um ein PDF Dokument mit Schreibschutz handelte.

Die zweite Möglichkeit in der Ursache der nicht eingetretenen Konvergenz bei der Signifikanzentwicklung, wäre eine falsche Klassifizierung eines Dokuments. Es könnte sein, dass der Inhalt des Dokuments thematisch nicht mit dem Thema dieser Klasse übereinstimmt, so dass folglich auch nicht alle bzw. keine Schlüsselworte dieser Klasse gefunden werden können. In Frage kommende Beispiele wären somit auch wieder die vier Dokumente C, D, G und J. Diese können aber aufgrund der oben beschriebenen Erklärung als Indiz für diese Ursache ausgeschlossen werden. Hierfür in Frage kommende Dokumente sind aber aufgrund ihres Schlüsselwortvorkommens alle anderen Dokumente. Denn bei allen anderen, außer eventuell noch der Dokumente B und G, werden doch verhältnismäßig sehr wenig Schlüsselworte gefunden. Bei den meisten Dokumenten kommen die meisten dieser Klasse hinterlegten Schlüsselworte überhaupt nicht vor. Dieses könnte bedeuten, dass diese Dokumente in eine falsche Klasse eingeteilt wurden.

Die dritte Möglichkeit für die Ursache der nicht eingetretenen Konvergenz kann aber auch darin bestehen, dass die die Klassen identifizierenden Schlüsselworte zu allgemein oder zu speziell gewählt worden sind. Wären sie beispielsweise zu allgemein, so würden diese Schlüsselworte wahrscheinlich sehr oft und in vielen dieser Klasse zugehörigen Dokumenten zu finden sein. Diese Möglichkeit kann anhand der vorliegenden Testdaten jedoch ausgeschlossen werden, da eher genau das Gegenteil der Fall ist. Wurden die zugrunde liegenden Schlüsselworte auf der anderen Seite zu speziell gewählt, besteht die Gefahr, dass diese Wörter in nur wenigen oder keinen Dokumenten gefunden werden können. Ein Indiz hierfür wäre, unter der Vorraussetzung einer korrekten Klassifikation, dass bei allen Dokumenten dieser Klasse keine des betroffenen Schlüsselwortes gefunden wurde. Dieses würde so sein, wenn die Signifikanz des Schlüsselwortes gleich *0* oder sehr klein sein würde.

Eine Aussage bezüglich dieser Möglichkeit lässt sich leider nicht machen, da die Vorraussetzung einer korrekten Klassifikation nicht gegeben ist. Aufgrund fehlenden Fachwissens kann eine falsche Klassifikation bei diesem Test nicht ausgeschlossen werden.

Auffällig wären unter der Gegebenheit einer korrekten Klassifikation die Schlüsselworte „*Kerndienstaufruf*“, „*Prozeß*“ und „Betriebsmittelverwaltung“. Die Begründung für das Nichtvorkommen des Wortes „*Prozeß*“ ist dabei leicht erklärt. Da dieses Wort ein „*ß*“ enthält wird es somit von der Textanalyse in „*ss*“ umgewandelt und stimmt somit mit dem Schlüsselwort „Prozess“ überein. Und dieses Wort wurde, wie aus Tabelle 8.9 ersichtlich, gefunden. Die anderen beiden Wörter kämen jedoch für eine zu spezielle Beschreibung in Frage. Vor allem das Wort „Kerndienstaufruf“. Gibt man es nämlich bei Google als Suchbegriff ein, erhält man lediglich ein Ergebnis. Vergleicht man dieses mit der Anzahl, die man mit den anderen Schlüsselworten finden kann, so ist dieses doch ein sehr erheblicher Unterschied.

Sowohl in dem Fall der falschen Klassifikation, als auch bei dem Eintreten der einer falschen Testanalyse wie bei den Dokumenten C, D und J. würde die errechnete Signifikanz bezüglich seiner mathematischen Korrektheit vollkommen richtig sein,

aber bei der praktischen Anwendung, beispielsweise einer Klassifizierung, wahrscheinlich zu falschen Ergebnissen führen.

Die Einteilung der Schlüsselworte in eine Klasse ist ausschlaggebend für die Analyse des Dokuments. Somit folglich auch, welche Signifikanzen errechnet bzw. neu errechnet werden müssen. Ist die Zuordnung in die ausgewählte Klasse falsch, so stimmt die inhaltliche Thematik des Dokuments nicht mit der inhaltlichen Thematik der ausgewählten Klasse überein. Auf diese Weise wird letztendlich die Signifikanz mathematisch wieder richtig berechnet. Kommt es aber zu einer praktischen Anwendung der errechneten Signifikanzen in anderen Anwendungen, so würden hier wahrscheinlich falsche Ergebnisse geliefert bzw. Schlussfolgerungen gezogen werden. Ein wesentlicher Faktor spielt hier natürlich die Gesamtmenge der dieser Klasse zugeteilten Dokumente. Da von allen Dokumenten die einzelne Signifikanz aufsummiert wird, und dann durch die Gesamtanzahl der Dokumente geteilt wird, verfälscht ein einziges falsches Dokument, bei einer größeren Menge an richtig klassifizierten Dokumenten, dieses Ergebnis nicht viel. Ist die Anzahl der falsch klassifizierten Dokumente jedoch größer, oder die Gesamtanzahl aller dieser Klassen zugeordnete Dokumente relativ klein, so ist diese Verfälschung des Ergebnisses für die Aussage bezüglich der Signifikanz eines Schlüsselwortes erheblich. Der einzige Lösungsweg diese Verfälschung der Signifikanz zu verhindern, ist das ausschließliche Klassifizieren mittels eines Experten. Experte soll in diesem Fall eine Person sein, die sich sehr gut mit der behandelnden Thematik auskennt. Da der Mensch aber Fehler machen kann und auch der Umfang des Wissensgebietes, der in diesem Falle benutzten ACM CSS sehr groß ist, ist eine falsche Klassifikation niemals vollkommen auszuschließen.

Der Fall, in dem das Dokument G aufgrund seines Schreibschutzes nicht analysiert werden konnte, ist für das Gesamtergebnis der Signifikanz nicht von Bedeutung. Die Dokumente werden immer von dem „*DocumentControllerServlet.java*“ (siehe 7.3.2.1) erst dann in einer Klasse gespeichert, nachdem sie analysiert wurden. Dieses bedeutet, tritt bei der Analyse ein Fehler auf, so wird dieses Dokument dieser Klasse nicht zugeordnet und kann somit auch nicht im weiteren Verlauf mit in die Berechnung der Signifikanz einfließen. Dieser Fall würde auch eintreten, wenn ein Dokumentformat nicht von der Textanalyse akzeptiert wird.

Für die Berechnung der Signifikanz sind hingegen die Fälle problematisch, bei denen die Textanalyse algorithmisch funktioniert, aber auf falschen Grunddaten durchgeführt wird. Diese Dokumente mit ihren „falschen“ Ergebnissen fließen schließlich gleichgewichtet mit einem korrekt analysierten Dokument in die Berechnung der Signifikanz mit ein. Um diese Fehlerquelle auszuschließen, bedarf es einer genauen manuellen Kontrolle des analysierten Dokuments. Theoretisch müsste nach jeder durchgeführten Analyse der Schlüsselwortvektor dieses Dokuments überprüft werden. Wurden keine Schlüsselworte in diesem Dokument gefunden, so gab es wahrscheinlich bei dem analysierten Format ein Problem mit der Extrahierung des Klartextes. Ist dieses der Fall, so muss dieses Dokument wieder aus der zugeteilten Klasse entfernt werden. Hieraufhin wird die Signifikanz des Ausgangszustandes wieder errechnet und die Signifikanz würde wieder unverfälscht sein.

Die unterschiedlichen, ermittelten Faktoren zeigen, dass eine genaue Interpretation bezüglich eines richtig verwendeten Algorithmus für die Berechnung der Signifikanz anhand der zugrunde liegenden Testdaten nicht möglich ist.

Zur Gesamtfunktionalität bzw. ob es ein richtiger Weg ist, die Errechnung der Signifikanzen über die Summe der relativen Häufigkeiten zu machen, kann anhand der zugrunde liegenden Testdaten nicht geschlussfolgert werden. Dafür sind die ermittelten Faktoren zu unterschiedlich. Wenn man ein Kriterium, wie beispielsweise die falsche Klassifikation, ausschließen könnte, wäre eine solche Aussage eventuell möglich. Da dieses aber aufgrund der für so einen Test erforderlichen Fachkenntnisse nicht realisierbar ist, kann ein solcher Test nicht durchgeführt werden.

9 Zusammenfassung

In der vorliegenden Diplomarbeit wurde eine Konzeption und Realisierung einer Webapplikation zur automatischen Erstellung von Signifikanzvektoren von Schlüsselworten in klassifizierten Texten hergeleitet und entwickelt. Die Datengrundlage war zum einen eine XML Datei mit den Klassen der ACM CSS Taxonomie und ihren identifizierenden Schlüsselworten. Zum anderen waren bzw. sind Daten erforderlich, auf deren Basis die Berechnung der Signifikanzen erfolgen kann.

Die dieser Webapplikation zugrunde liegende Thematik wurde durch die Möglichkeiten von Datenstrukturierung und ihrer Auffindung vorgestellt. Dabei wurden die Themen Metadaten, Klassifikationsschemata, sowie die Möglichkeiten und Verfahren einer automatischen Klassifikation näher erläutert.

Eine logische Herleitung ergab, dass die Signifikanzen durch die Summe der einzelnen relativen Häufigkeiten der jeweiligen Schlüsselworte in klassifizierten Dokumenten zu errechnen sind. Die über das Dateisystem oder das Internet gefundenen Dokumente müssen hierzu zunächst manuell über eine Klassifikationsmaske entsprechenden Klassen zugeordnet werden.

Mit der Zuordnung zu einer Klasse wurden daraufhin mittels Textanalyse, die dieser Klasse hinterlegten Schlüsselworte im Klartext des Dokuments gesucht und gezählt. Die Basis der Textanalyse bilden die Analyseklassen von *Jakarta Lucene*. *Open Source Präparatoren* filtern hierbei den benötigten Klartext aus unterschiedlichen Dateiformaten. Die durch diese Analyse gefundenen Werte werden in einer *MySQ*L Datenbank gespeichert und bilden so eine Grundlage bei der Erstellung der Signifikanzvektoren.

Alle zur Verfügung stehenden Funktionalitäten, sowie die Visualisierung der Ergebnisse, werden dabei über die Oberfläche der Webapplikation realisiert.

In einem abschließenden Test wurde die Webapplikation anhand eines praktischen Beispiels getestet. Die Ergebnisse zeigten die korrekte Funktionalität der

Webapplikation. Ob es ein richtiger Weg ist, die Errechnung der Signifikanzen über die Summe der relativen Häufigkeiten zu machen, konnte anhand der zugrunde liegenden Testdaten nicht geschlussfolgert werden.

9.1 Ausblick

Die im Durchführungstest teilweise aufgetretenen Probleme mit wenigen speziellen Dokumenten einiger Dateiformate wären ein Anlass für die Weiterentwicklung. Mit der Zeit werden neue Präparatoren und neuere Versionen der dieser Webapplikation zugrunde liegenden Präparatoren entwickelt werden. Hier könnte daher in regelmäßigen Abständen die Webapplikation weiterentwickelt werden, indem neuere Versionen bzw. eventuell neu entwickelte Präparatoren implementiert werden. Auch die Unterstützung weiterer Dateiformate kann als mögliche Weiterentwicklung an dieser Stelle genannt werden.

Weiterführende Entwicklungen könnten auch in Richtung automatische Klassifikation gehen. So wäre es denkbar, ein Modul zu implementieren, welches dem Benutzer automatisch Vorschläge für passende Klassen macht, die für das zu klassifizierende Dokument in Frage kommen. Basis wäre jedoch eine Schlüsselwortliste mit schon existierenden, hinterlegten Signifikanzen, die aussagekräftig genug sind, um einen solchen Vorschlag machen zu können. Das heißt eine große Menge an richtig klassifizierten Dokumenten in jeder Klasse.

Des Weiteren könnten noch Funktionalitäten, wie eine Analyse zur Entwicklung der Signifikanz implementiert werden. Hier könnte man z.B. visuell die Entwicklungen der einzelnen Signifikanzen darstellen.

Anhang I. Literaturverzeichnis

[ACM98] The ACM Computing Classifikation System (1998); Hrsg..: Association for Computing Machinery; elektr. Res.; URL: http://www.acm.org/class/1998/ccs98.html; Stand: 1998; letzter Zugriff: 10.07.2005

[BAL00] Balzert, Helmut: Lehrbuch der Softwaretechnik. Heidelberg, Berlin: Spektrum Akademischer Verlag, 2000; ISBN3-8274-0301-4

[BS95] Bearman, David; Sochats, Ken: Metadata Requirements for Evidence. Automating 21st Century Science – The Legal, Regulatory, Technical and Social Aspects of Electronic Laboratory Notebooks and Collaborative Computing; R&D, TeamScience Publishing, 1996; elektr. Res.; URL: http://www.archimuse.com/papers/nhprc/BACartic.html; letzter Zugriff: 20.07.2005

[BOH00] Bohner Michael: Instrumente zur Metadatenerschließung im WWW am Beispiel der Onlineschlagwortnormdatei; Masterarbeit: Universität Konstanz, Fachbereich Informatik und Informationswissenschaft, Studienfach Information Engineering; elektr. Res.; URL:http://www.bsz-bw.de/swop/volltexte/2003/22/pdf/metawww.pdf; letzter Zugriff: 20.07.2005

[BUC89] Buchanan, Brian: Bibliothekarische Klassifikationstheorie; München: Saur, 1989; ISBN 3-598-10788-9

[BUR04] Burkard, Johann: Dokumentation Projekt Digitale Medien, Beschleunigung von Suchanfragen durch Hashing; 2004; elektr. Res.; URL: http://johannburkard.de/documents/projekt_digitale_medien.pdf; letzter Zugriff: 20.07.2005

[CHA94] Chan, Lois Mai; Comaromi, John P.; Satija, Mohinder P.: Dewey Decimal Classification. A Practical Guide; Albany (New York): Forest Press, 1994; ISBN 0-910608-49-0

[DAV94] Davidson, K. T.: Four Reasons why we'll never be paperless; Mailer's Review, 1994; elektr. Res.; URL: http://www.raycomm.com/techwhirl/archives/9501/techwhirl-9501-01127.html; letzter Zugriff: 20.07.2005

[DCMI99] Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1; elektr. Res.; URL: http://dublincore.org/documents/dces/; letzter Zugriff: 20.07.2005

[DCMI02] Dublin Core Metadata Initiative: History of the Dublin Core Metadata Initiative; elektr. Res.; URL: http://dublincore.org/about/history/; letzter Zugriff: 20.07.2005

[DDC02] DDC: About the DDC; OCLC Dewey Decimal Classification; elektr. Res; URL http://www.oclc.org/dewey/about/default.htm; letzter Zugriff: 20.07.2005

[DIN32705] DIN 32 705: Klassifikationssysteme. Erstellung und Weiterentwicklung von Klassifikationssystemen; in: DIN-Taschenbuch 154; 4. Aufl.; hrsg. von DIN Deutsches Institut für Normung e. V.; Berlin, Wien, Zürich: Beuth 1996

[DIEPO] Diepold, P.: Ordnung im Multimediachaos, Bildungsserver und Meta-Daten; 16. Kongress der deutschen Gesellschaft für Erziehungswissenschaft; elektr. Res.; URL: http://www.educat.hu-berlin.de/publikation/paed_rs.pdf; letzter Zugriff: 20.07.2005

[EHS05] Engelhardt, M.; Hildebrand, A.; Schmidt, T. C.: Automatisierte Augmentierung von Lernobjekten in einer semantischen Interpretationsschicht der HyLOS Plattform; Protokoll 3. e-Learning Fachtagung Informatik – Delfi 2005; Bonn;

[EF00] Endres, Albert; Fellner, Dieter W.: Digitale Bibliotheken; Informatik-Lösungen für globale Wissensmärkte; Heidelberg: dpunkt, 2000; ISBN: 3-932588-77-0

[FEB00] Ferber R.: Data Mining und Information Retrival; Skript zur Vorlesung Information Retrival, Universität Darmstadt, elektr. Res.; URL. http://information-retrieval.de/dm-ir/; letzter Zugriff: 20.07.2005

[FUH02] Fuhr N.: Information Retrieval; Skript zur Vorlesung, Universität Dortmund; 2002; elektr. Res.; URL: http://www.is.informatik.uni-duisburg.de/courses/ir_ss05/folien/irskall.pdf; letzter Zugriff: 20.07.2005

[HAC00] Hacker, Rupert: Bibliotekarisches Grundwissen; 7., neu bearbeitete Aufl.; München: Saur, 2000; ISBN 3-598-11394-3

[HEB00] Heber J.: Knowlegde Discovery – Broker Entwicklungsarbeiten für das xFIND-Suchsystem, Diplomarbeit am Institut für Informationsverarbeitung und Computergestützte neue Medien, Technische Universität Graz; Österreich; elektr. Res.; URL: http://www.iicm.edu/thesis/jheber; letzter Zugriff: 20.07.2005

[HOF02] Hoffmann R.: Entwicklung einer benutzerunterstützten automatisierten Klassifikation von Web-Dokumenten, Diplomarbeit am Institut für Informationsverarbeitung und Computergestützte neue Medien, Technische Universität Graz 2002; Österreich; elektr. Res.; URL: http://www.iicm.edu/thesis/rhoff/Hoffmann_DA.htm; letzter Zugriff: 20.07.2005

[HOL01] Holzinger, Andreas: Interoperabilität und Metadaten, Workshop am 2. Business Meeting „Forum Neue Medien“, Wien 2001; Österreich; elektr. Res.; URL: http://serverprojekt.fh-joanneum.at/sp/thema/meta/metadaten.pdf; letzter Zugriff: 20.07.2005

[HOE02] Hörth C.: Klassifikation für digitale Bibliotheken; Diplomarbeit an der Fachhochschule Stuttgart; Stuttgart 2002; elektr. Res.; URL: http://www.fkdigbib.de/downloads/diplomarbeit_C_Hoerth.pdf; letzter Zugriff: 20.07.2005

[HOD01] Hodgins, W.: Draft Standard for Learning Object Metadata IEEE P1484.12/D6.1; IEEE Standards 2001; elektr. Res.; URL:

http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf; letzter Zugriff: 20.07.2005

[IEEE] Working Group 12: IEEE LOM; elektr. Res.; URL: http://ltsc.ieee.org/wg12/; letzter Zugriff: 20.07.2005

[KOE00] Koehler, K.; Web Document Morbidity and Change; elektr. Res.; URL: elektr. Res.; URL: http://ltsc.ieee.org/wg12/; letzter Zugriff: 20.07.2005; letzter Zugriff: 20.07.2005

[KLI98] Klinkenberg, R.: Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich ändernden Konzepten; Diplomarbeit Universität Dortmund 1998; elektr. Res.; URL: http://www-ai.cs.uni-dortmund.de/DOKUMENTE/klinkenberg_98a.pdf; letzter Zugriff: 20.07.2005

[KNO02] Knorz G.: Vorlesung Informationsmethodik, FH Darmstadt; elektr. Res.; URL: http://spock.iuw.fh-darmstadt.de/methodik/publ/lvlit2.htm; letzter Zugriff: 20.07.2005

[KNU99] Knudsen H.: Brauchen wir die Dewey-Dezimalklassifikation?; Bibliotheksdienst Heft 3, 1999; elektr. Res.; URL: http://bibliotheksdienst.zlb.de/1999/1999_03_Erschlie01.pdf; letzter Zugriff: 20.07.2005

[LUH58] Luhn H. P.: The automatic creation of literature abstracts, Journal of Research and Development; 1958; elektr. Res.; URL: http://www.research.ibm.com/journal/rd/022/luhn.pdf; letzter Zugriff 20.07.2005

[LV00] Lyman, P.; Varian, H. R.: How Much Information; Study of the University of California; 2000; elektr. Res.; URL: http://www.sims.berkeley.edu/how-much-info; letzter Zugriff 20.07.2005

[MG02] Marx, W.; Gramm, G.: Literaturflut – Informationslawine – Wissensexplosion – Wächst der Wissenschaft das Wissen über den Kopf?; Zentrale

Informationsvermittlung am Max-Planck-Institut, Stuttgart: 2002; elektr. Res.; URL: http://www.mpi-stuttgart.mpg.de/ivs/literaturflut.html; letzter Zugriff 20.07.2005

[MST94] Mitchie D., Spiegelhalter D., Taylor C.: Machine Learning, Neural and Statistical Classification; Ellis Horwood series in artifical intelligence; Ellis Horwood: New York; 1994; elektr. Res.; URL: http://www.amsta.leeds.ac.uk/~charles/statlog/whole.pdf; letzter Zugriff: 20.07.2005

[MZA01] Maly, K.; Zubair M.; Hesham A.; An Automated Classification System and Associated Digital Library Services; NDDL; 2001; elektr. Res.; URL: www.cs.odu.edu/~anan/publications/nddl.doc; letzter Zugriff: 20.07.2005

[NOH] Nohr, H.: Maschinelle Indexierung, Skript der Hochschule für Bibliotheks- und Informationswesen; Stuttgart; elektr. Res.; URL: http://www.iuk.hdm-stuttgart.de/nohr/IWS/Mi/MI.html; letzter Zugriff: 20.07.2005

[OCLC] OCLC Newsletter; No. 254; November/December 2002; Hrsg. OCLC Online Computer Library Center, Incorporated; Dublin (Ohio); ISSN 0163-898X

[PAP97] Papula, Lothar: Mathematik für Ingenieure und Naturwissenschaftler Band 2; 8. Auflage; Braunschweig/ Wiesbaden: Vieweg, 1997; ISBN 3-528-74237-2

[PIE01] Pierre, J. M.: On the automated classification of web sites; Linköping Electronic Articles in Computer and Information Science, Vol. 6(2001); ISSN 1401-9841; elektr. Res.; URL: http://citeseer.ist.psu.edu/pierre01automated.html; letzter Zugriff: 24.07.2005

[PKK] Priebe, Torsten; Kolter, Jan; Kiss, Christine: Semiautomatische Annotation von Textdokumenten mit semantischen Metadaten; Regensburg: Universität Regensburg; elektr. Res.; URL: http://www-ifs.uni-regensburg.de/PDF_Publikationen/PrKK05.pdf; letzter Zugriff: 24.07.2005

[RIS79] Van Rijsbergen C. J.: Information Retrival; London: Butterworth; elektr. Res.; URL: http://www.dcs.gla.ac.uk/Keith/Preface.html; letzter Zugriff: 20.07.2005

[RH98] Rosenstiel, W.; Heuser, U; Internetsuche und neuronale Netze: Stand der Technik; Universität Tübingen; ISSN 0946-3852

[WEI97] Weibel, Stuart: The Evolving Metadata Architecture fort he World Wide Web: Bringing Together the Semantics, Structure and Syntax of Resource Description; In: Proceedings on the International Symposium on Research, Development and Practice in Digital Libraries: ISDL'97, Tsukuba, Japan; elektr. Res.; URL: http://www.dl.ulis.ac.jp/ISDL97/proceedings/weibe.html; letzter Zugriff: 20.07.2005

[WEI01] Weiss M.; Automatische Indexierung mit besonderer Berücksichtigung deutschsprachiger Texte, Seminararbeit an der WU Wien; Institut für Informationswissenschaft; elektr. Res.; URL: http://www.ai.wu-wien.ac.at/~koch/lehre/inf-sem-ws-00/weiss/indexierung.pdf; letzter Zugriff: 20.07.2005

Anhang II. Beschreibung der beiliegenden CD

Die beiliegende CD ist in die folgenden Bestandteile aufgeteilt:

- Webapplikation in ausführbarer Form und Quellcode
- Die vorliegende schriftliche Diplomarbeit in digitaler Form
- Literatur der Internetquellen in digitaler Form
- Weitere Software

Verzeichnis	**Beschreibung**
webapplikation	Ausführbare Dateien der Webapplikation
quellcode	Quellcode der Webapplikation
diplomarbeit	Digitale Form dieser Diplomarbeit
literatur	Internetquellen in digitaler Form
software	Für die Webbapplikation benötigte Software
software → xampp	*XAMPP 1.4.13* Distribution, unter anderen mit: • *MySQL 4.1.13* • *Apache 2.0.53*
software → java	*Sun J2SDK 1.5* für Windows
software → tomcat	*Jakarta Tomcat 5.5*

Anhang III. Installation des Webapplikation

In diesem Teil soll kurz die Installation der Webapplikation, sowie der Programmteile beschrieben werden, die für die Ausführung der Webapplikation notwendig sind. Die nachfolgenden Ausführungen beziehen sich auf die Installation auf einem Windows XP System.

Für die Webapplikation sind Server notwendig, die die Dateiformate **.php* und **.war* ausführen können. Ist bzw. sind solche Server auf dem Zielsystem vorhanden, kann dieser Abschnitt übersprungen werden. Andernfalls sind im Verzeichnis „*xampp*“ und „*tomcat*“ des „*software*“-Verzeichnisses diese Server zu finden. Der *Tomcat Server* benötigt die Version *1.4* des *JDK*. Diese ist ggf. vorher zu installieren und befindet sich in dem Verzeichnis „*java*“.

Als Datenbanksystem wurde *MySQL* verwendet. Eine Version für Windows wurde durch die Installation der *XAMPP* Software bereits installiert. Alle benötigten Datenbanken, sowie Tabellen werden bei dem ersten Aufruf der Webapplikation automatisch erstellt. Hierfür ist ein User notwendig, der alle Rechte auf dieser Datenbank besitzt.

Um die Webapplikation aufrufen zu können, müssen im nächsten Schritt die Datei „*webapplikation.zip*“ aus dem „*webapplikation*“-Verzeichnis in eine gewünschtes Verzeichnis innerhalb des „*htdocs*“-Verzeichnisses des zuvor installierten Apache-Webservers (aus der *XAMPP* Distribution) entpackt werden.

Als nächstes muss die Datei „*DocAnalyzer.war*“ in das Verzeichnis „*webapps*“ des *Tomcat* Servers kopiert werden.

Bevor die Webapplikation nun aufgerufen werden kann, müssen zunächst diese beiden Server gestartet werden und die Konfigurationsdaten angepasst werden. Dieser Schritt ist in Anhang IV zu lesen.

Jetzt ist die Weppapplikation über die „*index.php*“ Seite des gewählten Verzeichnisses der entpackten „*webapplikation.zip*“ –Datei des Apache Webservers über einen Browser aufrufbar und funktionsbereit.

Anhang IV. Konfiguration der Webapplikation

Folgende Konfigurationsdaten müssen in der Datei „*config.inc.php*“ im Verzeichnis „*func*“ des Verzeichnisses „*content*“ angepasst werden. Das „*content*“-Verzeichnis befindet sich in dem ausgewählten Verzeichnis der entpackten „*webapplikation.php*“-Datei.

Zugangsdaten zur Datenbank	
$GLOBALS[db_user]	Benutzername für die Datenbank
$GLOBALS[db_password]	Passwort für die Datenbank
$GLOBALS[db_host]	Pfad zur Datenbank

Pfad zum „htdoc“ - Verzeichnis der „*index.php*“-Datei	
$GLOBALS[home]	Hier muss der Pfad von dem „*htdocs*“ - Verzeichnis des (Apache) Webservers zum Verzeichnis der „*index.php*“ - Datei angegeben werden.

Pfad zu dem „*webapps*“ - Verzeichnis des Tomcat Servers	
$GLOBALS[servletpfad]	Hier muss der Pfad zu dem „*webapps*“ - Verzeichnis des Tomcat Servers bzw. zu dem Verzeichnis der „DocAnalyzer.war“ - Datei angegeben werden.

Key für die Benutzung des Google API	
$GLOBALS[googlekey]	Hier muss der Key aus der Registrierung für die Benutzung des Google – API eingegeben werden.

Die folgenden Konfigurationsdaten müssen in der Datei „*db.properties*“ angepasst werden. Diese Datei liegt nach dem Start ersten Start des Tomcat Servers bzw. nach dem „*Deployen*“ im Verzeichnis „*servlets*“. Zu finden ist dieses Verzeichnis durch den Pfad „*DocAnalyzer/WEB-INF/classes/de/haw/*“ beginnend im „*webapps*“ - Verzeichnis des Tomcat Servers.

Zugangsdaten zur Datenbank	
user	Benutzername für die Datenbank
password	Passwort für die Datenbank
host	Den Pfad zur Datenbank

Anhang V. Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 23. August 2005 _____

Torben Hoffmann