Masterarbeit

Suche und Klassifikation von informationstechnologischen Spezifikationen

Matthias Hilpold

Technische Universität Graz
Institut für Wissensmanagement (IWM)
Institutsvorstand: Univ.-Prof. Dr. rer. nat. Klaus Tochtermann

Betreuer: Dipl.-Ing. Dr. techn. Michael Granitzer
Graz, Oktober 2010

Kurzfassung

In der heutigen Zeit steigt die Menge an digitalen Daten tagtäglich. Durch das Internet ist ein Teil davon der breiten Masse jederzeit zugänglich. Dabei unterstützen Suchmaschinen den Benutzer aus der scheinbar unerschöpflichen Menge an Daten die gewünschten Informationen herauszufiltern. Ebenso wird es im Intranet eines großen Unternehmens zunehmend schwieriger die Unmengen an Daten adäquat zu organisieren und zu strukturieren, um die gesuchten Informationen schnell zu finden.

Für den niederländischen Halbleiterhersteller namens NXP Semiconductors wurde in dieser Arbeit ein Wissensmanagementsystem entwickelt, um den Zugang zu den Spezifikationen des intern entwickelten JCOP-Betriebssystems zu optimieren. Dabei können Spezifikationen vom Benutzer verwaltet, gruppiert und durchsucht werden. Als Grundlage für die Volltextsuche wurde ein bewährtes Information-Retrieval-Verfahren namens Vektorraummodell verwendet. Bei der Indizierung der Spezifikationen wird der Text extrahiert, gefiltert und in einen Index eingebettet. Dadurch wird dem Benutzer ermöglicht den Volltext der Spezifikationen zu durchsuchen. Aufbauend auf die Volltextsuche der Spezifikationen wurde mittels eines maschinellen Lernverfahrens namens K-Nearest-Neighbour die vom Benutzer durchgeführte Gruppierung einzelner Spezifikationen mit den Ergebnissen der K-Nearest-Neighbour-Klassifikation verglichen. Nach mehreren Optimierungsschritten konnte der Recall der Klassifikation auf über 70% und die Accuracy auf über 90% verbessert werden.

Master´s Thesis

Search and Classification of Information Technology Specifications

Matthias Hilpold

Graz University of Technology
Institute for Knowledge Management (KMI)
Head of Institute: Univ.-Prof. Dr. rer. nat. Klaus Tochtermann

Supervisor: Dipl.-Ing. Dr. techn. Michael Granitzer
Graz, October 2010

Abstract

In our days the amount of digital data increases every day. A part of this data is available through the Internet for everyone and at any time. Different search engines support the user to find the desired information on the Internet. Even on an Intranet of a large company the effective management of data becomes progressively difficult. For this reason a knowledge management system was developed for the Dutch semiconductor manufacturer called NXP Semiconductors. It is used to optimize the access to the specifications of the JCOP operating system which is developed by NXP Semiconductors. The knowledge management system allows the user to manage, group and search specifications. As foundation for the system a reliable information retrieval model called vector space model was used. During the indexing process the text of the specifications is extracted, filtered and embedded in an index. This way the system offers a specification full text search to the user. Based on the full text search a machine learning method called K-Nearest-Neighbour was applied in order to compare the grouping of single specifications done by the user with the results of the K-Nearest-Neighbour classification. After several optimization steps the Recall of the classification was improved to over 70% and the Accuracy to over 10%.

Danksagung

Diese Arbeit wurde in Zusammenarbeit mit NXP Semiconductors in Gratkorn und dem Institut für Wissensmanagement an der Technischen Universität Graz durchgeführt.

Für die Betreuung und Bereitstellung eines Arbeitsplatzes in Gratkorn möchte ich mich bei Dipl.-Ing. Andreas Lessiak von NXP Semiconductors bedanken.

Technische Unterstützung erhielt ich seitens des Instituts für Wissensmanagement von Dipl.-Ing. Dr. techn. Michael Granitzer, dem ich an dieser Stelle auch herzlich danken möchte.

Besonderer Dank gilt natürlich meinen Eltern. Ohne ihre finanzielle Unterstützung wäre mir ein Studienabschluss in überschaubarer Zeit wohl nicht möglich gewesen.

Mein Dank geht auch an Melanie Glantschnig und Ralf Hillebrand, die mir bei der Überprüfung der Arbeit unterstützt haben.

Zum Schluss möchte ich mich noch bei meinen Freunden Hannes, Ivan und Markus für die schöne Studienzeit und für ihre Unterstützung im Laufe meines gesamten Studentenlebens bedanken.

Matthias Hilpold
Graz, Oktober 2010

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/ Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……

(Unterschrift)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In diesem Kapitel wird kurz das Unternehmen (siehe Abschnitt 1.1) vorgestellt, in dessen Zusammenarbeit diese Arbeit verfasst wurde. Ebenso werden die Motivation (siehe Abschnitt 1.3) und die Anforderungen (siehe Abschnitt 1.4) an das zu entwickelnde System vorgestellt. Anschließend werden einige Notationen (siehe Abschnitt 1.6) definiert, die für die formalen Beschreibungen in dieser Arbeit nötig sind, und ein Überblick (siehe Abschnitt 1.7) über die gesamte Arbeit gegeben.

1.1 NXP Semiconductors

NXP (Next eXPerience) Semiconductors ist ein niederländischer Halbleiterhersteller mit Firmensitz in Eindhoven. Ursprünglich war NXP Semiconductors ein Teil von Royal Philips Electronics. Im Jahr 2006 wurde es ausgegliedert und bildet seitdem ein eigenständiges Unternehmen. NXP Semiconductors zählt zu den größten Halbleiterherstellern in Europa. Halbleiterprodukte und Systemlösungen von NXP Semiconductors findet man im Automobilmarkt, in der Heimelektronik, in Mobiltelefonen sowie in unzähligen Chipkarten wie elektronischen Ausweisen, Reisepässen und Bankkarten wieder [26].

1.2 Java Card OpenPlatform

Java Cards[1] sind in der heutigen Zeit weit verbreitet. Sie reichen von einfachen Kundenkarten über Bankkarten bis hin zu Reisepässen. Die Sicherheit ist in den meisten Fällen oberstes Gebot und wird mittels komplexen kryptographischen Verfahren erreicht. Die Schnittstelle zwischen der Java Card Hardware und der entsprechenden Java-Card-Anwendung ist das sogenannte JC/ OP-Betriebssystem. JC/ OP steht für Java Card OpenPlatform. Es war das erste Smart-Card-Betriebssystem, das vom IBM Zürich Research Labor entwickelt wurde. Der erste Prototyp mit Atmel 8 Bit Mikroprozessor und FlashRAM wurde 1998 vorgestellt. Dieser hat jedoch das Prototyp-Stadium nie verlassen. Mit neuem Infineon Smart Card Chip und Unterstützung von Public-Key-Kryptographie seitens des Betriebssystems konnte im selben Jahr noch ein laufendes System basierend auf JC/ OP präsentiert werden. Im darauffolgenden Jahr wechselte IBM auf einen Philips MifarePro Chip. Damit wollte IBM beweisen, dass JC/ OP mit

[1] Java Cards: Chipkarten mit einer virtuellen Maschine, die Java Applikationen ausführen können [20].

Plattformunabhängigkeit, kontaktlosen Operationen und knappen Ressourcen umgehen kann. Das Resultat war der erste Chip, der gleichzeitig das kontaktlose ISO14443-4 Protocol und das kontaktbehaftete T=0 Protocol unterstützte. Daraufhin wurde der Name JC/OP zu JCOP umgetauft. Im Jahr 2007 kaufte NXP Semiconductors eine JCOP-Lizenz von IBM und darf seitdem ein eigenes JCOP-Betriebssystem entwickeln [20].

1.3 Motivation

Das JCOP-Betriebssystem basiert auf einer umfangreichen Anzahl von Spezifikationen. Mitarbeiter benötigen für ihre tägliche Arbeit jedoch nur einen Bruchteil dieser Spezifikationen und haben darum als Einzelne keinen Überblick über das gesamte Projekt. Aktuell werden die Spezifikationen lediglich auf einem Dateiserver hinterlegt. Das erschwert die Einarbeitung neuer Mitarbeiter, die Wartung und die Kontrolle des Projektes. Aus diesem Grund soll im Rahmen dieser Arbeit ein JCOP-Wissensmanagementsystem entwickelt werden, welches den Zugang zu diesen Spezifikationen optimiert.

1.4 Systemanforderungen

Die Systemanforderungen seitens NXP Semiconductors können in technologische und funktionale Anforderungen unterschieden werden.

1.4.1 Technologische Anforderungen

Aktuell betreibt NXP Semiconductors im internen Firmennetzwerk einen MySQL Datenbankserver und einen Apache Webserver. Die technologischen Anforderungen beschränken sich darauf, wenn möglich diese bestehenden Ressourcen zu nutzen. Des Weiteren sollen für die Entwicklung des JCOP-Wissensmanagementsystems neben den bereitgestellten Ressourcen ausschließlich Open-Source-Projekte verwendet werden.

Im Folgenden werden noch einmal alle technologischen Anforderungen zusammengefasst:

- MySQL Datenbankserver
- Apache Webserver
- Open-Source-Projekte

1.4.2 Funktionale Anforderungen

NXP Semiconductors ist ein internationales Unternehmen mit Standorten auf der ganzen Welt. Die Entwicklung des JCOP-Betriebssystems beschränkt sich auf die Standorte in Gratkorn bei Graz (A), Hamburg (D) und Caen (F). Bedingt durch die verteilte Entwicklung des Projektes soll das JCOP-Wissensmanagementsystem von verschiedenen Standorten erreichbar sein. Eine einfache Benutzerverwaltung soll im System integriert werden. Außerdem soll eine Benutzerschnittstelle im System die Wartung des Systems und die Integrierung neuer Spezifikationen erlauben.

Aktuell sind die Spezifikationen auf einem Dateiserver im internen Firmennetzwerk erreichbar. Dadurch ist es sehr schwierig abzuleiten, welche Spezifikationen untereinander in Beziehung stehen. Aus diesem Grund soll es im System möglich sein, Beziehungen zwischen Spezifikationen zu definieren und diese dem Benutzer visuell darzustellen. Ebenso soll es möglich sein, Spezifikationen in Form einer virtuellen Ordnerstruktur zu verwalten, wobei eine Spezifikation zu mehreren Ordnern gehören kann. Im Folgenden werden noch einmal alle grundlegenden funktionalen Anforderungen zusammengefasst:

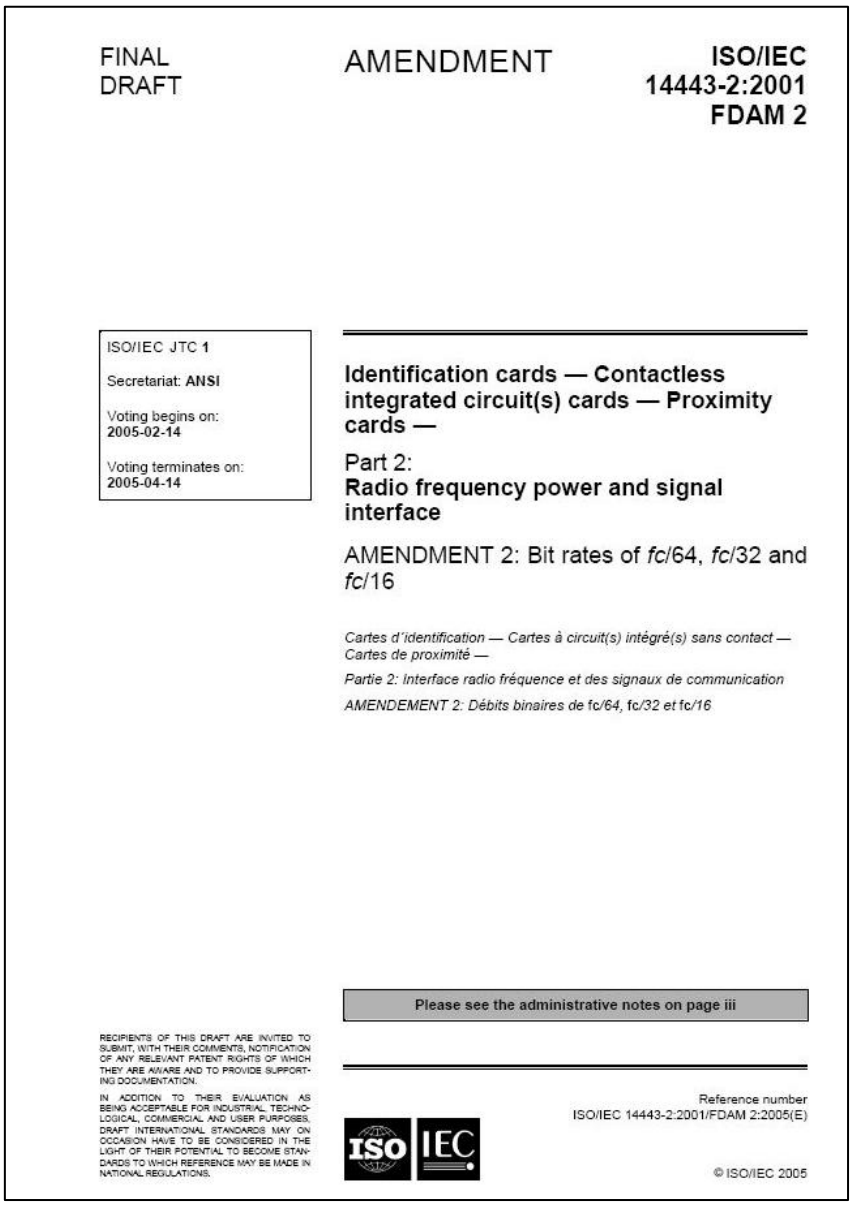
- Zugang von unterschiedlichen Standorten
- Einfache Benutzerverwaltung
- Benutzerschnittstelle für Wartung
- Beziehungen zwischen Spezifikationen
- Visualisierung der Beziehungen
- Virtuelle Ordnerstruktur für Spezifikationen

Zusätzlich zu den grundlegenden funktionalen Anforderungen gibt es einige erweiterte Funktionen, die das JCOP-Wissensmanagementsystem besitzen soll. Dazu gehört eine Volltextsuche, die es erlaubt, den Inhalt von Spezifikationen zu durchsuchen. Des Weiteren soll das System es erlauben, ähnliche Spezifikationen aufzuspüren und die vom Benutzer durchgeführte Klassifikation der Spezifikationen zu überprüfen. Um Änderungen der Spezifikationen im System zu verfolgen, soll der Benutzer über RSS Feeds auf dem Laufenden gehalten werden. Im Folgenden werden noch einmal alle erweiterten funktionalen Anforderungen zusammengefasst:

- Volltextsuche über Spezifikationen
- Ähnlichkeitssuche von Spezifikationen
- Klassifikation von Spezifikationen
- RSS Feeds

1.5 JCOP-Spezifikationen

Das JCOP-Betriebssystem umfasst eine Vielzahl von Spezifikationen. Das Titelblatt einer solchen Spezifikation ist in Abbildung 1 dargestellt. Alle Spezifikationen sind in englischer Sprache verfasst und als PDF-Dateien verfügbar. Die PDF-Metadaten sind zum großen Teil nicht gefüllt und können darum nicht verwendet werden. Die Länge der Spezifikationen variiert sehr stark. Spezifikationen über 100 Seiten sind keine Seltenheit. Teilweise sind die PDF-Dateien durch Passwörter geschützt oder enthalten nur Bilder. Auch der strukturelle Aufbau der Spezifikationen ist nicht einheitlich gegliedert und kann deswegen nicht genutzt werden.



FINAL
DRAFT

AMENDMENT

ISO/IEC 14443-2:2001 FDAM 2

ISO/IEC JTC 1

Secretariat: **ANSI**

Voting begins on:
2005-02-14

Voting terminates on:
2005-04-14

Identification cards — Contactless integrated circuit(s) cards — Proximity cards —

Part 2:
Radio frequency power and signal interface

AMENDMENT 2: Bit rates of *fc*/64, *fc*/32 and *fc*/16

Cartes d'identification — Cartes à circuit(s) intégré(s) sans contact — Cartes de proximité —

Partie 2: Interface radio fréquence et des signaux de communication

AMENDEMENT 2: Débits binaires de fc/64, fc/32 et fc/16

Please see the administrative notes on page iii

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

Reference number
ISO/IEC 14443-2:2001/FDAM 2:2005(E)

© ISO/IEC 2005

Abbildung 1: Titelblatt einer JCOP-Spezifikation

1.6 Notationen

In Tabelle 1 werden einige mathematische Notationen vorgenommen, die für die formalen Beschreibungen in dieser Arbeit nötig sind.

Notation	**Bedeutung**
k_i	Ein beliebiger Indexterm aus K
$K = \{k_1, \dots, k_{\|K\|}\}$	Menge aller Indexterme
d_j oder d	Ein beliebiges Dokument aus D
$D = \{d_1, \dots, d_{\|D\|}\}$	Menge aller Dokumente
$tfreq_{i,j}$	Term-Frequenz für Indexterm k_i im Dokument d_j
$\mathrm{t}freqnorm_{i,j}$	Normalisierte Term-Frequenz von $tfreq_{i,j}$
$dfreq_i$	Dokumentfrequenz für Indexterm k_i
$\vec{d}_j = (w_{1,j}, \dots, w_{\|K\|,j})$	Indexterm-Vektor für Dokument d_j
$\vec{\mathrm{d}} = (w_1, \dots, w_{\|K\|})$	Indexterm-Vektor für Dokument d
$\vec{q} = (w_{1,q}, \dots, w_{\|K\|,q})$	Indexterm-Vektor für Suchanfrage q
$w_{i,j}$	Gewicht für Indexterm k_i im Dokument d_j
$w_{i,\mathrm{q}}$	Gewicht für Indexterm k_i in Suchanfrage q
$g_i(\vec{d}_j) = w_{i,j}$	Extrahiert aus einem Indexterm-Vektor $\vec{d}_j$ für einen Indexterm k_i das entsprechende Gewicht
$\mathrm{g_i}(\vec{\mathrm{q}}_{\mathrm{cc}}) = \mathrm{w_{i,q}}$	Extrahiert aus einer Konjunktiven Komponente von $\vec{q}_{dnf}$ für einen Indexterm k_i das entsprechende Gewicht
q_i oder q	Eine beliebige Suchanfrage des Benutzers
$\vec{q}_{dnf}$	Disjunktive Normalform einer booleschen Suchanfrage q
$\vec{q}_{cc}$	Konjunktive Komponente von $\vec{q}_{dnf}$
$tfreq_{\mathrm{l,j}}$	Term-Frequenz eines Indexterm k_l im Dokument d_j
$tfreq_{i,q}$	Term-Frequenz eines Indexterm k_i in Suchanfrage q
θ	Winkel zweier Vektoren
R	Menge relevanter Dokumente für ein Suchanfrage q
$\bar{R}$	Menge nicht relevanter Dokumente für ein Suchanfrage q
$sim(\vec{d}_j, q)$	Ähnlichkeitswert eines Dokuments $\vec{d}_j$ und einer Suchanfrage q
$sim(\vec{d}_j, \vec{q})$	Ähnlichkeitswert eines Dokuments $\vec{d}_j$ und einer Suchanfrage $\vec{q}$
V	Menge der gefundenen relevanten Dokumente
V_i	Menge der gefundenen relevanten Dokumente, die Indexterm k_i enthalten

c_i *oder* c	Eine beliebige Klasse aus C
$C = \{c_1, \dots, c_{\|C\|}\}$	Menge aller Klassen
$Q = \{q_1, \dots, q_{\|Q\|}\}$	Menge ausgewählter Suchanfragen
$P(R\|\vec{d}_j)$	Wahrscheinlichkeit, dass das Dokument d_j relevant ist
$P(\bar{R}\|\vec{d}_j)$	Wahrscheinlichkeit, dass das Dokument d_j nicht relevant ist
P(R)	Wahrscheinlichkeit, dass ein zufällig gewähltes Dokument relevant ist
P($\bar{R}$)	Wahrscheinlichkeit, dass ein zufällig gewähltes Dokument nicht relevant ist
$P(\vec{d}_j\|\mathrm{R})$	Wahrscheinlichkeit zufällig ein Dokument d_j aus R zu wählen
$P(\vec{d}_j\|\bar{R})$	Wahrscheinlichkeit zufällig ein Dokument d_j aus $\bar{R}$ zu wählen
$P(k_i\|\mathrm{R})$	Wahrscheinlichkeit, dass der Indexterm in einem zufällig gewählten Dokument aus R vorkommt
$P(\bar{k_i}\|R)$	Wahrscheinlichkeit, dass der Indexterm nicht in einem zufällig gewählten Dokument aus R vorkommt
$P(k_i\|\bar{\mathrm{R}})$	Wahrscheinlichkeit, dass der Indexterm in einem zufällig gewählten Dokument aus $\bar{R}$ vorkommt
$P(\bar{k_i}\|\bar{R})$	Wahrscheinlichkeit, dass der Indexterm nicht in einem zufällig gewählten Dokument aus $\bar{R}$ vorkommt
D_c	Menge aller Dokumente d_j einer Klasse c
D_d	Menge aller Nachbardokumente eines Dokuments d_j
$\tilde{\Phi}(\mathrm{j,i}): d_j \times c_i$	Näherungsfunktion, die bestimmt, ob ein Dokument d_j zu einer Klasse c_i gehört
$\tilde{\Phi}(\mathrm{j,q_i}): d_j \times q_i$	Näherungsfunktion, die bestimmt, ob ein Dokument d_j für eine Suchanfrage q_i relevant ist
$\vec{w}^T\vec{x} = b$	Definition einer Hyperebene im mehrdimensionalen Raum
$\vec{\mu}(c)$	Vektor, der den Schwerpunkt einer Klasse c beschreibt
TP_i	True Positive einer Klasse c_i oder Suchanfrage q_i
FP_i	False Positive einer Klasse c_i oder Suchanfrage q_i
FN_i	False Negative einer Klasse c_i oder Suchanfrage q_i
TN_i	True Negative einer Klasse c_i oder Suchanfrage q_i
$Recall_i$	Recall einer Klasse c_i oder Suchanfrage q_i
$Precision_i$	Precision einer Klasse c_i oder Suchanfrage q_i
$Error_i$	Error einer Klasse c_i oder Suchanfrage q_i
$Accuracy_i$	Accuracy einer Klasse c_i oder Suchanfrage q_i
$Fallout_i$	Fallout einer Klasse c_i oder Suchanfrage q_i

$F1_i$	F1 einer Klasse c_i oder Suchanfrage q_i
$Recall^M$	Makromittelwert Recall
$Recall^M$	Makromittelwert Precision
$Recall^\mu$	Mikromittelwert Recall
$Recall^\mu$	Mikromittelwert Precision
$F_\beta(i)$	F-Maß einer Klasse c_i oder Suchanfrage q_i mit Parameter β
$\alpha\epsilon[0,1]$	Konstante für das F-Maß
$\beta^2\epsilon[0,\infty]$	Konstante für das F-Maß
$E(i)$	E-Maß einer Klasse c_i oder Suchanfrage q_i
b	Konstante für das E-Maß

Tabelle 1: Mathematische Notationen

1.7 Überblick

In diesem Abschnitt wird der Aufbau der Arbeit erläutert. Um einen groben Überblick zu geben wird kurz auf den Inhalt der einzelnen Kapitel eingegangen.

Im Kapitel 1 wird kurz das Unternehmen vorgestellt, in dessen Zusammenarbeit diese Arbeit erstellt wurde. Die Motivation für das Projekt sowie dessen Anforderungen und Notationen für formale Beschreibungen werden ebenso in diesem Kapitel definiert.

Textoperationen und klassische Information-Retrieval-Modelle werden im Kapitel 2 vorgestellt. Den Schwerpunkt dabei bildet das sogenannte Vektorraummodell mit Gewichtungsfunktionen und Ähnlichkeitsfunktionen.

Textklassifikation ist das Thema des Kapitels 3. Dabei wird Textklassifikation im Allgemeinen behandelt und es werden verschiedene Algorithmen vorgestellt.

Im Kapitel 4 wird auf verschiedene Kennwerte eingegangen, die im Bereich Information Retrieval und Textklassifikation verwendet werden, um Systeme zu evaluieren und zu vergleichen. Recall und Precision spielen dabei eine zentrale Rolle.

Der gesamte praktische Teil dieser Arbeit wird im Kapitel 5 besprochen. Die Systemarchitektur sowie verwendete Datenstrukturen und Algorithmen werden in diesem Kapitel vorgestellt.

Das Kapitel 6 beschäftigt sich mit der Systemevaluierung. Dabei wird die Klassifikation der Spezifikationen jener der Benutzern gegenüber gestellt. Die Ergebnisse werden in diesem Kapitel ausgewertet und interpretiert. Außerdem werden die Systemanforderungen in diesem Kapitel evaluiert.

Eine Zusammenfassung der Arbeit und ein Ausblick, wie die Qualität des entwickelten Systems gesteigert werden kann wird im Kapitel 7 gegeben.

2 Information Retrieval

Zu Beginn dieses Kapitels wird eine Einführung (siehe Abschnitt 2.1) zum Thema Information Retrieval gegeben. Dann folgt ein Einblick in den Information-Retrieval-Prozess (siehe Abschnitt 2.2) und in die logische Abbildung von Dokumenten (siehe Abschnitt 2.4). Außerdem wird aufgezeigt, wie Information-Retrieval-Modelle (siehe Abschnitt 2.5) eingeteilt werden können. Die drei klassischen Information-Retrieval-Verfahren werden anschließend im Detail dargestellt. Am Ende dieses Kapitels wird auf Vektorähnlichkeitsfunktionen (siehe Abschnitt 2.5.4.2) im Vektorraummodell (siehe Abschnitt 2.5.4) eingegangen. Diese spielen eine zentrale Rolle für Ähnlichkeitsberechnung zwischen Dokumenten bzw. zwischen Suchanfrage und Dokumenten.

2.1 Einführung

Schon seit langer Zeit haben Menschen ihre Informationen in irgendeiner Weise verwaltet, um diese zu einem späteren Zeitpunkt wieder finden zu können. Als typisches Beispiel dafür kann das Inhaltsverzeichnis eines Buches gesehen werden. Falls ein Buchbestand aus mehr als nur ein paar wenigen Exemplaren besteht, ist es nötig, eine Datenstruktur anzulegen, um die Informationssuche zu beschleunigen. Ein altes populäres Konzept einer solchen Datenstruktur (siehe Abschnitt 2.3) ist der sogenannte Index. Ein Index ist eine Ansammlung von ausgewählten Termen bzw. Konzepten, welche in Relation zu den gesuchten Informationen oder Dokumenten stehen. Indizes treten in unterschiedlichster Weise auf. Sie bilden aber den Kern in nahezu jedem modernen Information-Retrieval-System. Für lange Zeit wurden solche Indizes in Form hierarchischer Kategorien manuell erstellt. In vielen Bibliotheken wird der Index immer noch auf diese Weise erstellt. Die Entwicklung moderner Computersysteme macht es möglich, solche Indizes nun voll automatisiert zu erstellen.

Das Information-Retrieval-Problem unterscheidet zwischen zwei Ansichten: computerorientiert und benutzerorientiert. In einem computerorientierten System geht es hauptsächlich darum, einen Index effektiv aufzubauen, Suchanfragen schnell zu verarbeiten und die Qualität der Suchantwort zu optimieren. In einem benutzerorientierten System werden die Bedürfnisse der Benutzer analysiert und darauf basierend charakteristische Terme abgeleitet.

Bibliotheken gehörten zu den ersten Institutionen, welche Information-Retrieval-Systeme zur Informationssuche nutzten. Diese Systeme wurden anfangs von

Universitäten und später von kommerziellen Institutionen entwickelt. Die erste Generation dieser Systeme erlaubte es nur nach Titel und Autor eines Dokumentes zu suchen. Erst in der zweiten Generation wurden Kapitelnamen und Terme aus dem Text in die Suche aufgenommen. In der heutigen Generation von Information Retrieval liegt der Schwerpunkt auf graphischen Benutzerschnittstellen, elektronischen Formularen und offenen Systemarchitekturen [2].

Häufig wird Information Retrieval mit Data Retrieval verwechselt, obwohl beide Modelle sich grundlegend unterscheiden. In einem Data-Retrieval-System wird die Suchanfrage des Benutzers direkt beantwortet. Die Antwort ist in diesem Fall entweder richtig oder falsch und vereinfacht die Evaluierung eines solchen Systems erheblich. Im Vergleich dazu wird in einem Information-Retrieval-System die Suchanfrage eines Benutzers indirekt beantwortet. Das bedeutet, es wird als Antwort eine Menge von Dokumenten zurück geliefert, in denen wahrscheinlich die gesuchte Information enthalten ist. Die Bewertung der Antwort ist in diesem Fall subjektiv und hängt vom Benutzer ab. Dies erschwert die Evaluierung (siehe Kapitel 4) von Information-Retrieval-Systemen [27].

2.2 Information-Retrieval-Prozess

Bevor in einem Information-Retrieval-System ein Suchprozess ausgeführt werden kann, müssen die Dokumente mittels Textoperationen in eine logische Abbildung (siehe Abschnitt 2.4) überführt werden. Diese logische Abbildung wird in einer speziellen Datenstruktur gespeichert. Häufig wird dafür ein Index verwendet. Ein Index ist eine spezielle Datenstruktur (siehe Abschnitt 2.3), die es erlaubt in großen Datenmengen schnell zu suchen [2].

Sobald der Index aufgebaut ist, kann der Suchprozess beginnen. Der Benutzer spezifiziert ein Informationsbedürfnis, welches mit denselben Textoperationen wie zuvor bei der Indizierung der Dokumente in eine logische Abbildung umgewandelt wird. Aus dieser logischen Abbildung wird nun die eigentliche Suchanfrage erstellt und mittels verschiedener mathematischer Modelle (siehe Abschnitt 2.5) und Gewichtungsfunktionen möglichst passende Dokumente zur Suchanfrage zu finden.

Die gefundenen Dokumente werden aufbereitet (z.B. nach Relevanz sortiert oder gruppiert) dem Benutzer zurück geliefert. Manche Systeme erlauben es dem Benutzer aus den aufbereiteten Ergebnissen jene zu markieren, die wirklich für ihn relevant sind. Auf Grund dieses Feedbacks wird die ursprüngliche Suchanfrage angepasst, um das Suchergebnis für den Benutzer zu optimieren [2].

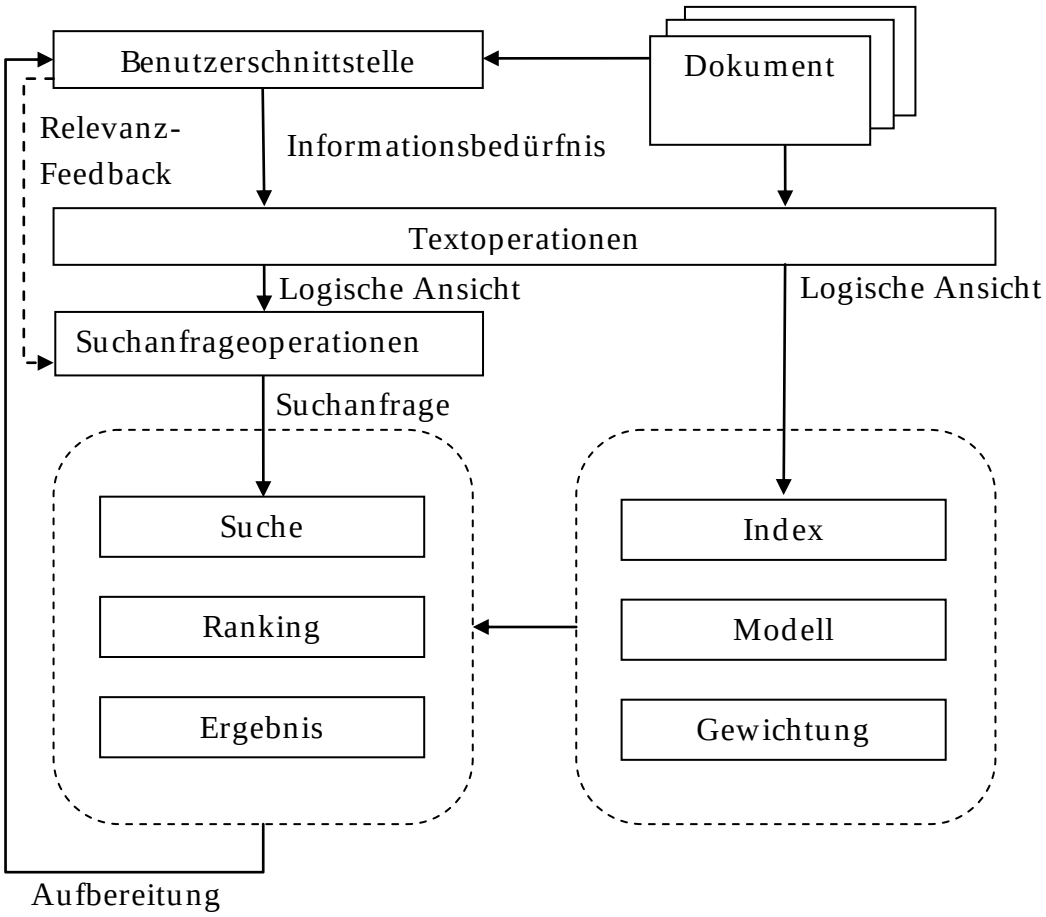


Abbildung 2: Information-Retrieval-Prozess [2]

Laut Salton [28] ist die grundlegende Idee von Relevance Feedback, dass relevante Dokumente bezüglich einer Suchanfrage ähnliche logische Abbildungen besitzen. Sind die relevanten Dokumente einer Suchanfrage bekannt, kann man die Suchanfrage verändern, um die Ähnlichkeit zu den identifizierten relevanten Dokumenten zu steigern. Die neu formulierte Suchanfrage sollte nun zusätzliche relevante Dokumente finden. Eine effektive Methode zur Anpassung der Suchanfrage zu finden ist das Hauptproblem von Relevance Feedback. In manchen Fällen liefert das Relevance Feedback keine Verbesserung des Suchergebnisses, da die relevanten Dokumente im Dokumentenraum nicht beisammen liegen oder von nicht relevanten Dokumenten umgeben sind. In Abbildung 2 ist der gesamte Information Retrieval Prozess dargestellt.

2.3 Datenstruktur

Die populärste Datenstruktur und laut Manning [3] ein zentrales Konzept im Bereich Information Retrieval ist der sogenannte Inverted Index. Der intuitive Ansatz ein Dokument zu speichern ist für jedes Dokument jeweils die gesamten Terme zu

speichern. Dies bedeutete einen hohen Zeitaufwand, um alle Dokumente zu finden, die einen bestimmten Term enthalten. Eine effizientere Methode bietet der Inverted Index. Dabei wird der intuitive Ansatz umgedreht, nämlich werden für jeden Term aus dem Dokumentenbestand jene Dokumente zugeordnet, die diesen Term beinhalten. Mit dieser Datenstruktur lässt sich schneller die Frage beantworten, welche Dokumente einen bestimmten Term enthalten. Manchmal wird der Inverted Index auch Inverted File oder Inverted List genannt [2].

2.4 Logische Abbildung

Zu Beginn des Information-Retrieval-Prozess (siehe Abschnitt 2.2) werden Dokumente in eine logische Abbildung überführt. Im Allgemeinen besitzen Dokumente einen Inhalt und eine Struktur. Beide Aspekte können in die logische Abbildung eines Dokumentes einfließen.

Wie man die Struktur eines Dokumentes für den Information-Retrieval-Prozess nutzen kann, haben Burkowski [29] und Baeza [30] gezeigt. Burkowski entwickelte ein Modell, das den Text eines Dokuments in nicht überlappende Textregionen zerlegt und in einer flachen Liste zusammenfasst. Baeza wählte für sein Modell hingegen einen hierarchischen Ansatz, um die Struktur eines Dokumentes abzubilden.

Mittels Textoperationen wird die Komplexität der Abbildung eines Dokumentes reduziert. Textoperationen werden häufig auch Texttransformationen genannt. Zu Beginn des Abbildungsprozesses werden durch eine lexikalische Analyse (siehe Abschnitt 2.4.1) die Wörter eines Dokumentes extrahiert.

Statistisch gesehen wird ein großer Teil der Sprache durch wenige häufige Wörter abgedeckt. Da häufig auftretende Wörter sich für die Differenzierung von Dokumenten nicht eignen, können diese schon während der logischen Abbildung eines Dokumentes mit sogenannten Stoppwort-Filtern (siehe Abschnitt 2.4.2) entfernt werden.

Besonders Hauptwörter eignen sich Dokumente zu differenzieren, da sie auch ohne Kontext aussagekräftig sind. Daher werden andere Wortarten wie Adjektive und Verben häufig mittels Hauptwort-Filter entfernt.

Durch Stemming-Algorithmen (siehe Abschnitt 2.4.3) können verwandte Wörter auf einen einheitlichen, gemeinsamen Wortstamm abgebildet werden. Damit wird die Komplexität der Abbildung weiter reduziert.

In manchen Fällen ist es nötig Dokumente auf ein einheitliches Vokabular abzubilden. Dafür werden sogenannte Thesauri (siehe Abschnitt 2.4.4) verwendet [2].

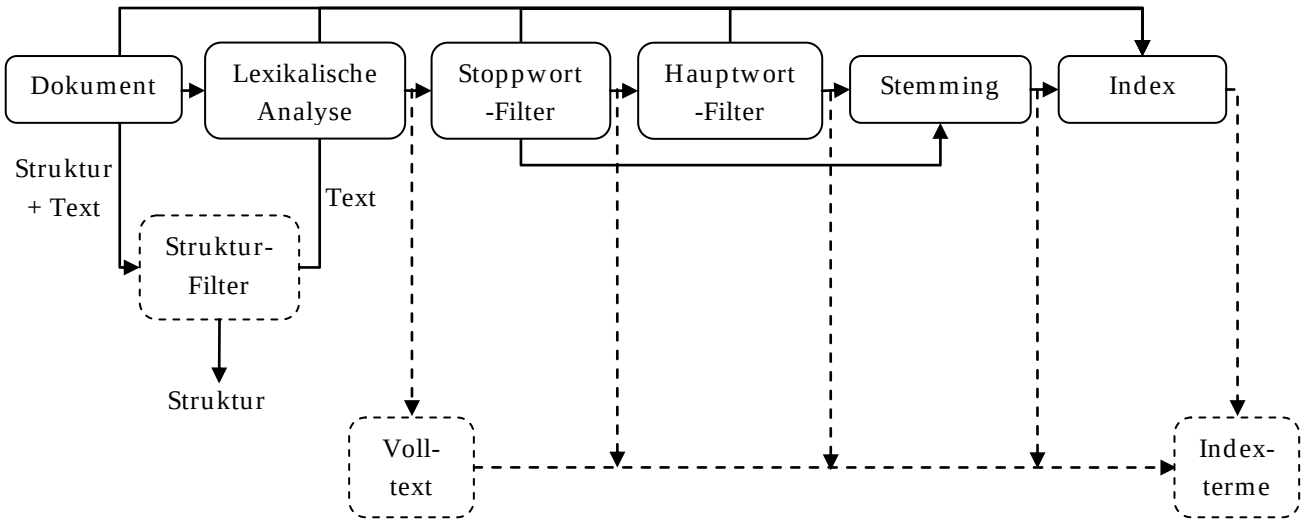


Abbildung 3: Logische Abbildung eines Dokuments [2]

2.4.1 Lexikalische Analyse

Die lexikalische Analyse beschreibt den Prozess aus einer Folge von Buchstaben und Zeichen eine Folge von Wörtern zu extrahieren. Diese extrahierten Wörter bilden die potentiellen Indexterme. Auf den ersten Blick scheint es ausreichend eine Zeichenfolge zwischen zwei Leerzeichen als Wort zu interpretieren. Bei weiterer Betrachtung wird ersichtlich, dass auch folgende Aspekte näher betrachtet werden müssen [21]:

- Zahlen
- Bindestriche
- Satzzeichen
- Groß-/ Kleinschreibung

Zahlen ohne Kontext sind im Allgemeinen keine guten Indexterme, da sie alleinstehend zu wenig aussagen. Ein gutes Beispiel dafür sind die Seitenzahlen eines Dokuments. Daher werden Zahlen häufig nicht im Index aufgenommen. Abhängig vom Kontext kann es jedoch nötig sein Ausnahmen zuzulassen. Jahreszahlen oder die Nummer einer Kreditkarte sind gute Beispiele dafür. In einer komplexeren lexikalischen Analyse können Datumsangaben und Zahlen auf ein einheitliches Format normalisiert werden.

Ebenso umstritten ist die Handhabung der Bindestriche in einem Text. Die Auftrennung des Textes bei Bindestrichen kann in vielen Fällen die Semantik eines Terms zerstören. Es hat sich für die Wortextraktion bewährt eine allgemeine Regel für Bindestriche einzuführen und gegebenenfalls Ausnahmen von Fall zu Fall zuzulassen. Derselbe Ansatz wird für Satzzeichen verfolgt.

Die Groß-/ Kleinschreibung spielt in den meisten Fällen keine Rolle für die Identifikation eines Indexterms. Die Terme werden entweder in die eine oder in die andere Form transformiert.

Die Implementierung der Textoperationen stellt laut [21] keine große Schwierigkeit dar. Das Hauptproblem ist, dass es keine allgemeine Lösung für die lexikalische Analyse eines Textes gibt. Für jedes Information-Retrieval-System muss die lexikalische Analyse an den Kontext der Dokumente angepasst werden.

2.4.2 Stoppwörter

Wörter mit sehr hoher Häufigkeit im Dokumentenbestand eignen sich nicht für Information Retrieval bzw. Textklassifikation. Solche Wörter werden Stoppwörter genannt und während der Indizierung häufig aussortiert. Artikel, Präpositionen und Konjunktionen sind typische Kandidaten für solche Stoppwörter. Durch die Elimination der Stoppwörter ergibt sich noch ein weiterer Vorteil. Der Speicherbedarf der Indexstruktur wird erheblich reduziert. Die Liste der Stoppwörter variiert von System zu System und kann bei Bedarf auch beispielsweise durch Adjektive und Adverbien erweitert werden. In seltenen Fällen kann die Elimination von Stoppwörtern allerdings den Recall (siehe Abschnitt 4.1) verringern. Die Suche nach Phrasen, die nahezu nur aus Stoppwörtern bestehen, wird durch Stoppwort-Filterung nahezu unmöglich [2].

2.4.3 Stemming

Das ursprüngliche Ziel von Stemming-Algorithmen war es, die Anzahl der verschiedenen Terme eines Systems zu reduzieren. Dadurch wurden Ressourcen eingespart und die Systemleistung erhöht. Erstmals wurden Stemming-Algorithmen in den sechziger Jahren verwendet.

Mit steigender Leistungsfähigkeit der Prozessoren und Speichermedien war es nicht mehr nötig auf Ressourcen Rücksicht zu nehmen. In heutiger Zeit werden Stemming-Algorithmen verwendet, um die Effektivität eines Systems zu steigern und den Recall zu verbessern. Dabei werden verschiedene morphologische Varianten eines Terms auf den Wortstamm abgebildet. Voraussetzung dafür ist, dass der Wortstamm die Bedeutung des Wortes mit sich trägt und das Suffix des Wortes die Bedeutung nur syntaktisch verändert. Sprachen besitzen eine festgelegte Grammatik. Zusätzlich

zu den normalen Reduktionsregeln verlangen grammatikalische Ausnahmen immer wieder die Verwendung von sogenannten Look-Up-Tabellen.

Durch Stemming-Algorithmen sollten ursprünglich in erster Linie Ressourcen wie Speicherplatz eingespart werden. Die Arbeiten von Lennon im Jahre 1981 [17] und Harman im Jahre 1991 [18] zeigten allerdings, dass Stemming-Algorithmen für Speicherplatzersparnisse großer Datensätze nicht besonders geeignet sind. Lennon reduzierte den Speicherbedarf der kleinen Cranfield Collection mittels Stemming-Algorithmen um 32 %. Harman verwendete größere Datensätze und reduzierte diese um lediglich 13,5 %.

Stemming-Algorithmen werden heutzutage verwendet, um den Recall eines Systems zu verbessern. Solange ein semantisch konsistenter Wortstamm für verschiedene Wörter identifiziert werden kann, helfen Stemming-Algorithmen mögliche nicht-relevante, jedoch verwandte Wörter zu finden. Der Recall eines Systems kann mittels Stemming-Algorithmen zwar verbessert werden, dafür wird häufig die Precision (siehe Abschnitt 4.2) durch Stemming beeinträchtigt. Jede Generalisierung der Suchanfrage führt dazu, dass wahrscheinlich mehr nicht-relevante Dokumente als relevante gefunden werden. Da die Precision die nicht-relevanten der gefundenen Dokumente zu minimieren versucht, wird diese durch Stemming häufig verschlechtert.

Nicht jedes Wort ist für den Stemming-Prozess geeignet. Auf Abkürzungen oder Namen sollte kein Stemming angewandt werden, da ihre morphologische Basis nicht zum Stemming-Konzept passt. Systeme sollten daher Wörter kategorisieren können und auf Basis dessen entscheiden, ob Stemming angewandt werden soll oder nicht.

Die bekanntesten Stemming-Algorithmen entfernen die Suffixe eines Wortes, um den Wortstamm zu erhalten. Dies geschieht oft auch in rekursiver Form. Zu den bekanntesten Stemming-Algorithmen gehören folgende:

- Porter (siehe Abschnitt 2.4.3.1)
- Lovins
- Paice
- Dawson
- Krovetz

Stemming wird generell auf den Inhalt der Dokumente als auch auf Suchanfragen des Benutzers angewandt. Wird beim Stemming die semantische Bedeutung eines Terms der Suchanfrage geändert, werden Dokumente gefunden, die nichts mit der Suchanfrage zu tun haben. Dadurch könnte der Benutzer die Integrität des Systems

in Frage stellen. Alternative Methoden sind die Wörterbuch- und die Successor-Stemming-Methode [6].

2.4.3.1 Porter-Algorithmus

Der Porter-Algorithmus wurde 1980 von Martin Porter an der Cambridge Universität entwickelt. Er ist der am weitesten verbreitete Stemming-Algorithmus für englische Text und in den verschiedensten Programmiersprachen implementiert und verfügbar.

Der Algorithmus ist in fünf Schritte eingeteilt, die sequentiell abgearbeitet werden, um den Suffix zu ersetzen und so den Wortstamm zu erhalten. Für die nähere Betrachtung des Algorithmus sind einige Definitionen notwendig. Dazu gehört die Definition der Konsonanten bzw. der Vokale. Zu den Vokalen werden die Buchstaben A, E, I, O, U und ein Konsonant gefolgt von Y gezählt. Alle übrigen Buchstaben sind Konsonanten.

Eine Sequenz von Konsonanten bzw. Vokalen mit mindestens einem Zeichen wird mit C bzw. V bezeichnet. Jedes Wort kann somit durch folgende Form dargestellt werden, wobei m die Anzahl der VC-Wiederholungen kennzeichnet und die eckigen Klammern die optionalen Teile:

$$[C](VC)^m[V]$$

Jeder Schritt ist durch eine Regel definiert. Falls die Bedingung der Regel erfüllt ist, wird das aktuelle Suffix mit einem neuen Suffix ersetzt [25]:

$$(Bedingung): Suffix1 \rightarrow Suffix2$$

Einige Beispiele solcher Regeln sehen wie folgt aus:

$$(NULL)\text{:IES} \rightarrow \text{"}I\text{"}$$

$$(NULL)\text{:S} \rightarrow \text{""}$$

$$(m > 0)\text{:"}EED\text{"} \rightarrow \text{"}EE\text{"}$$

$$(*\ v\ *)\text{:ED} \rightarrow \text{""}$$

$$(*\ v\ *)\text{:ING} \rightarrow \text{""}$$

$$(*\ v\ *)\text{: Y} \rightarrow \text{" }I\text{"}$$

$$(m > 0)\text{:"}ATIONAL\text{"} \rightarrow \text{"}ATE\text{"}$$

Der detaillierte Algorithmus kann in der ursprünglichen Publikation von Porter [24] gefunden werden.

2.4.4 Thesaurus

Der Begriff Thesaurus hat griechische und lateinische Wurzeln. In der einfachsten Form versteht man unter dem Begriff Thesaurus eine Wortsammlung aus einem bestimmten Wissensbereich. Jedes dieser Wörter steht mit gewissen anderen Wörtern in Beziehung. Meistens werden diese syntaktisch anders geschrieben, besitzen aber dieselbe Bedeutung, sogenannte Synonyme. Ein Thesaurus besteht im Allgemeinen aus folgenden drei Hauptkomponenten:

- Indexterme
- Term-Beziehungen
- Datenstruktur der Term-Beziehungen

Ein komplexerer Thesaurus wurde 1988 von Roget [22] veröffentlicht. Darin berücksichtigte er nicht nur einfach Worte, sondern auch Phrasen. Dieser Thesaurus ist nicht nur auf einen Wissensbereich fokussiert, sondern ist von allgemeiner Natur. Wörter und Phrasen sind darin in Kategorien und Unterkategorien organisiert.

Die Motivation, die hinter der Verwendung eines Thesaurus steckt, ist das kontrollierte Vokabular für die Indizierung und Suche. Dadurch kann die Indizierung und die Suche normalisiert werden. Dies funktioniert gut für beschränkte Wissensbereiche. Im Web hingegen ist es noch umstritten, inwieweit Thesauri die Informationssuche unterstützen können, da die Informationsmenge unbeschränkt und sehr dynamisch ist. Foskett [23] fasst den Nutzen von Thesauri wie folgt zusammen:

- Ein Thesaurus bietet ein Standardvokabular für die Indizierung und die Suche.
- Der Benutzer wird mit Term-Vorschlägen bei der Formulierung der Suchanfrage unterstützt.
- Die Suchanfrage des Benutzers kann auf seine Bedürfnisse eingeschränkt bzw. ausgeweitet werden.

2.5 Information-Retrieval-Modelle

Im Bereich Information Retrieval wird zwischen drei klassischen Modellen unterschieden. Dazu gehört das Boolesche Modell, das Vektorraummodell und das probabilistische Modell [2].

Im Booleschen Model werden Suchanfragen und Dokumente als Mengen von Termen abgebildet; es wird in [5] zu den mengentheoretischen Modellen gezählt. Das Vektorraummodell bildet Suchanfragen und Dokumente als Vektoren im

mehrdimensionalen Raum ab und gehört dadurch zu den algebraischen Modellen. Für die Abbildung der Suchanfragen und Dokumente im klassischen probabilistischen Modell werden, wie der Name schon vermuten lässt, probabilistische Ansätze verwendet. Es zählt daher zu den probabilistischen Modellen.

Mit der Zeit haben sich parallel zu den drei klassischen Modellen alternative Paradigmen entwickelt (siehe Abbildung 4). Das Fuzzy-Modell basiert auf unscharfen Mengen und das Extended-Boolean-Modell ermöglicht partielle Übereinstimmung zwischen Suchanfragen und Dokument. Beide Modelle bilden Alternativen zum Booleschen Modell. Zu den alternativen algebraischen Modellen gehört das Generalized-Vector-Space-, das Latent-Semantic-Indexing- und das Neural-Network-Modell. Im Vergleich zum Vektorraummodell nimmt man im Generalized-Vector-Space-Modell an, dass zwei Indexterm-Vektoren zwar linear unabhängig jedoch nicht orthogonal zueinander sind. Das Latent-Semantic-Indexing-Modell verfolgt den Ansatz relevante Dokumente aufgrund von Konzepten anstatt von Indextermen zu finden. Dafür werden Indexterme in einem niederdimensionalen Raum mit Konzepten verknüpft. Das Neural-Network-Modell verwendet neuronale Netze, um für Terme einer Suchanfrage die passenden Dokumente zu finden. Das Inference Network und das Belief Network basieren auf das Bayesian Network und werden zu den alternativen probabilistischen Modellen gezählt [2].

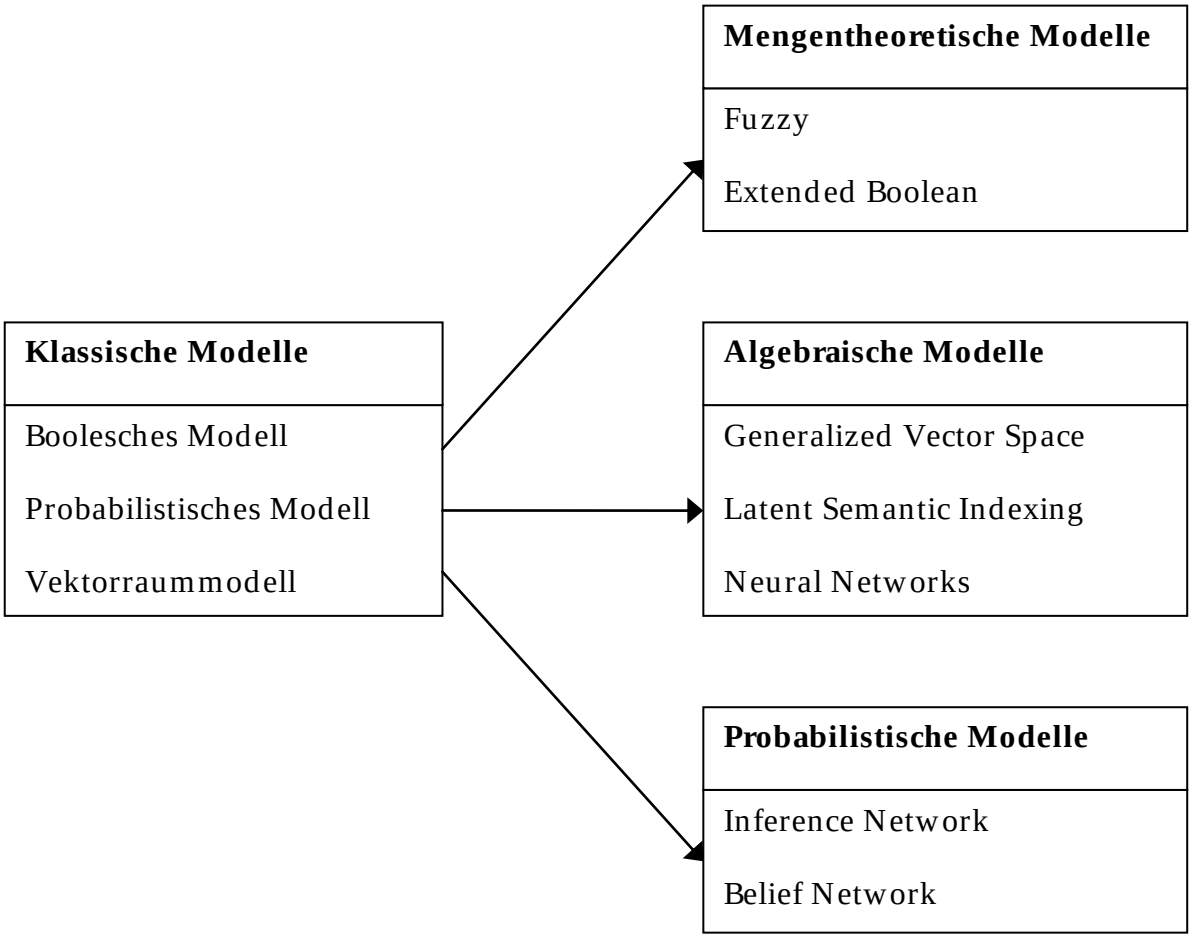


Abbildung 4: Information-Retrieval-Modelle [2]

2.5.1 Grundkonzept

In klassischen Information-Retrieval-Modellen wird ein Dokument d_j als Menge repräsentativer Terme abgebildet. Diese Terme werden Indexterme k_i genannt. Ein Indexterm ist ein im Dokumentenbestand $D = \{d_1, \dots, d_{|D|}\}$ vorkommendes Wort. Im Allgemeinen eignen sich besonders Hauptwörter als Indexterme, da ihre Bedeutung im Vergleich zu Adjektiven, Verben oder Adverbien selbsterklärend ist.

Bei Betrachtung aller Indexterme $K = \{k_1, \dots, k_{|K|}\}$ eines Systems ist erkennbar, dass nicht jeder Indexterm geeignet ist, um den Inhalt eines bestimmten Dokuments abzubilden. In Wirklichkeit ist die Auswahl, welche Indexterme ein Dokument am besten beschreiben, äußerst schwierig. Trotz dieser Schwierigkeit können gewisse Eigenschaften eines Indexterms mittels mathematischer Modelle gemessen und für die Berechnung seiner Brauchbarkeit herangezogen werden. Zum Beispiel ist ein in verschiedenen Dokumenten sehr häufig auftretendes Wort als Indexterm nicht geeignet, da es nicht zur Unterscheidung von Dokumenten genutzt werden kann.

Kaum auftretende Wörter sind hingegen für die Beschreibung eines Dokuments sehr viel besser geeignet.

Wie gut ein Indexterm k_i ein Dokument d_j beschreibt, wird durch ein numerisches Gewicht $w_{i,j} \geq 0$ beschrieben. Alle Gewichte zusammengefasst bilden den gewichteten Indexterm-Vektor $\vec{d}_j = (w_{1,j}, \dots, w_{|K|,j})$. Mit der Funktion $g_i(\vec{d}_j) = w_{i,j}$ wird ein bestimmtes Gewicht aus einem gewichteten Indexterm-Vektor extrahiert [2].

Zur Vereinfachung werden üblicherweise einzelne Indexterme als voneinander unabhängig angenommen. In Wirklichkeit sind Indexterme jedoch nicht voneinander unabhängig, da das Auftreten eines Indexterms innerhalb eines Dokuments das Auftreten eines anderen mit sich ziehen kann. Durch diese Annahme wird die Berechnung der Indexterm-Gewichtung stark vereinfacht [7].

2.5.2 Boolesches Modell

Das Boolesche Modell ist ein einfaches Information-Retrieval-Modell basierend auf Mengentheorie und Boolescher Algebra. Der Benutzer formuliert im Booleschen Modell eine Suchanfrage q mittels Boolescher Ausdrücke. Dafür stehen die logischen Operatoren „AND“ ($\wedge$), „OR“ ($\vee$) und „NOT“ ($\neg$) zur Verfügung. Mittels Klammersetzung können auch komplexere Suchanfragen formuliert werden. Boolesche Ausdrücke besitzen zwar eine klare Semantik, sind aber für den Benutzer nicht immer leicht zu formulieren.

Eine Boolesche Suchanfrage q kann in eine sogenannte Disjunktive Normalform $\vec{q}_{dnf}$ überführt werden (siehe Abbildung 5). Die einzelnen Komponenten dieser Normalform werden konjunktive Komponenten $\vec{q}_{cc}$ genannt. Jeder dieser konjunktiven Komponenten ist ein binär gewichteter Vektor, dessen Einträge jeweils für einen Indexterm stehen. In Abbildung 5 steht jede konjunktive Komponente für ein Tupel (k_1, k_2, k_3).

Der gewichtete Dokument-Indexterm-Vektor $\vec{d}_j$ im Booleschen Modell ist ebenso ein binär Vektor mit $w_{i,j} \in \{0,1\}$. Ist ein Indexterm in einem Dokument enthalten, erhält dieser im gewichteten Indexterm-Vektor das Gewicht 1, ansonsten das Gewicht 0.

Das Boolesche Modell kann formal wie folgt definiert werden:

$$\boldsymbol{sim}(\vec{d}_j, q) = \begin{cases} 1, falls\ \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall_{k_i},\ g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0, sonst \end{cases}$$

$$q = k_1 \wedge (k_2 \vee \neg k_3) \Rightarrow \vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$$

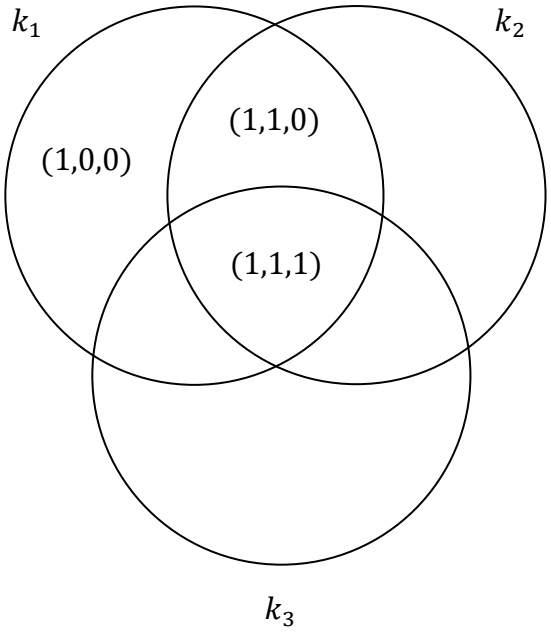


Abbildung 5: Disjunktive Normalform einer Booleschen Suchanfrage [2]

Im Booleschen Modell ist ein Dokument d_j für eine Suchanfrage q_t entweder relevant ($sim(\overrightarrow{d_j}, q) = 1$) oder nicht relevant ($sim(\overrightarrow{d_j}, q) = 0$). Damit ein Dokument für eine Suchanfrage relevant ist, muss die exakte Übereinstimmung (Exact Matching) zwischen Suchanfrage und Dokument erfüllt sein. Diese Tatsache stellt den Schwachpunkt des Booleschen Modells dar, da Dokumente mit partieller Suchanfragenübereinstimmung nicht als relevant bewertet werden. Ein Dokument mit dem Indexterm-Vektor $\vec{d}_j = (0,1,0)$ ist demnach für die Suchanfrage in Abbildung 5 nicht relevant, obwohl der Indexterm k_2 im Dokument enthalten ist [2].

Die Einfachheit und die klare formale Beschreibung werden als Hauptvorteile des Booleschen Modells angesehen. Die Nachteile dieses Modells überwiegen jedoch klar. Die Hauptnachteile des Booleschen Modells und aller anderen Modelle, die mit der Methode der exakten Übereinstimmung arbeiten, können wie folgt zusammengefasst werden [8]:

- Durch exakte Übereinstimmung zwischen Suchanfrage und Dokument werden viele relevante Dokumente nicht beachtet.
- Alle Ergebnisse einer Suchanfrage sind gleich relevant, da kein Ranking der Suchergebnisse stattfindet.
- Alle Wörter innerhalb einer Suchanfrage oder eines Dokuments werden gleich relevant behandelt.

- Die Formulierung der Suchanfragen ist zwar klar definiert, wird vom Benutzer allerdings als kompliziert empfunden.
- Die Suchanfrage und die Abbildung des Dokuments müssen dasselbe Vokabular verwenden.

Laut Ferber [4] bieten viele Boolesche Information-Retrieval-Systeme neben den grundlegenden Operationen weitere Operationen zur Suche an. Dazu gehört, dass für Felder mit Zahlen Vergleichsoperationen zur Verfügung stehen und nach mehreren Termen zusammengefasst als Phrase gesucht werden kann. Häufig können mittels Wildcards und regulärer Ausdrücke komplexere Suchanfragen formuliert werden.

2.5.3 Probabilistisches Modell

Das probabilistische Modell wurde 1976 von Roberston und Jones [16] vorgestellt. Später wurde es auch unter dem Binary Independence Retrieval Modell bekannt. Dieses Modell bedient sich, wie der Name schon vermuten lässt, probabilistischer Ansätze.

Die fundamentale Idee des probabilistischen Modells besagt, dass es für eine Suchanfrage eine ideale Antwortmenge an Dokumenten gibt. Der Prozess der Suchanfrage kann auch als Eigenschaftssuche der idealen Antwortmenge gesehen werden. Da die Eigenschaften der idealen Antwortmenge zum Zeitpunkt der Suchanfrage unbekannt sind, werden sie anfangs geschätzt. Diese Schätzung erlaubt es, vorab eine probabilistische Beschreibung der idealen Antwortmenge zu generieren. Diese wird verwendet, um die erste Antwortmenge mit relevanten Dokumenten für die Suchanfrage zu finden. Das System verfeinert dann die anfängliche Schätzung. Dabei kann auch das Feedback des Benutzers mit einbezogen werden. Durch Wiederholen dieses Vorgangs und ständige Verfeinerung der Schätzung nähert man sich der realen Beschreibung der idealen Antwortmenge.

Wie beim Booleschen Modell sind beim probabilistischen Modell alle Term-Gewichte binär ($w_{i,j} \in \{0,1\}$ und $w_{i,q} \in \{0,1\}$). Die Menge der relevanten Dokumente für eine Suchanfrage q wird durch R gekennzeichnet und die der nicht relevanten Dokumente durch $\overline{R}$. Die Wahrscheinlichkeit, mit der ein Dokument d_j für eine Suchanfrage q relevant ist, wird durch $P(R|\vec{d}_j)$ definiert. Das entsprechende gilt für $P(\overline{R}|\vec{d}_j)$. Damit ist die Ähnlichkeit eines Dokuments zu einer Suchanfrage wie folgt definiert:

$$sim(\vec{d}_j, \vec{q}) = \frac{P(R|\vec{d}_j)}{P(\bar{R}|\vec{d}_j)}$$

Durch Anwendung der Bayesschen Regel gelangt man zu folgender Formulierung:

$$sim(\vec{d}_j, \vec{q}) = \frac{P(\vec{d}_j|\mathrm{R}) \times \mathrm{P}(\mathrm{R})}{P(\vec{d}_j|\overline{\mathrm{R}}) \times \mathrm{P}(\overline{\mathrm{R}})}$$

P(R) steht für die Wahrscheinlichkeit, mit der ein Dokument aus dem gesamten Dokumentenbestand relevant ist. $P(\vec{d}_j|\mathrm{R})$ hingegen beschreibt, wie wahrscheinlich es ist, zufällig ein Dokument aus der Menge R der relevanten Dokumente zu finden. Das entsprechende gilt für $\mathrm{P}(\overline{\mathrm{R}})$ und $\mathrm{P}(\vec{d}_j|\overline{\mathrm{R}})$.

Da P(R) und $\mathrm{P}(\overline{\mathrm{R}})$ für alle Dokumente gleich sind, werden diese beiden Werte zur Vereinfachung weggelassen. Dadurch verändert sich der Ausdruck wie folgt:

$$sim(\vec{d}_j, \vec{q}) \sim \frac{P(\vec{d}_j|\mathrm{R})}{P(\vec{d}_j|\overline{\mathrm{R}})}$$

Unter Verwendung der Logarithmus-Funktion, Ignorierung der konstanten Terme für dieselbe Suchanfrage und der Annahme dass alle Indexterme voneinander unabhängig sind, gelangt man zu folgender grundlegenden Ähnlichkeitsfunktion des probabilistischen Modells:

$$sim(\vec{d}_j, \vec{q}) \sim \sum_{i=1}^{|K|} w_{i,q} \times w_{i,j} \times (log \frac{P(k_i|R)}{1 - P(k_i|R)} + log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})})$$

$$mit\ P(k_i|R) + P(\overline{k_i}|R) = 1$$

Da anfangs nur die Suchanfrage existiert und noch keine Dokumente gefunden wurden, werden zur Vereinfachung zwei Annahmen getroffen. $P(k_i|R)$ wird für alle k_i konstant angenommen und die Verteilung der Indexterme unter den nicht-relevanten Dokumenten wird durch die Verteilung der Indexterme unter allen Dokumenten approximiert.

$$P(k_i|R) = 0.5$$

$$P(k_i|\overline{R}) = \frac{dfreq_i}{|D|}$$

Mit dieser Schätzung können die ersten Dokumente, die Suchanfrageterme enthalten, gefunden und sortiert werden.

Die anfängliche Schätzung kann nun verbessert werden. Dabei wird $P(k_i|R)$ durch die Verteilung des Indexterms k_i unter den bisherigen gefundenen Dokumenten

ersetzt. Für $P(k_i|\overline{R})$ wird angenommen, dass alle bisher nicht gefundenen Dokumente nicht relevant sind. Daraus ergeben sich folgende Formulierungen:

$$P(k_i|R) = \frac{|V_i|}{|V|}$$

$$P(k_i|\overline{R}) = \frac{dfreq_i - |V_i|}{|D| - |V|}$$

Durch Wiederholung dieses Prozesses werden die Schätzungen immer wieder verfeinert und nähern sich den optimalen Werten an. Häufig wird ein Anpassungsfaktor addiert, um Problemen für kleine Werte von $|V|$ und $|V_i|$ vorzubeugen:

$$P(k_i|R) = \frac{|V_i| + 0.5}{|V| + 1}$$

$$P(k_i|\overline{R}) = \frac{dfreq_i - |V_i| + 0.5}{|D| - |V| + 1}$$

Manchmal ist ein konstanter Anpassungsfaktor nicht wirklich geeignet. Als Alternative kann $\frac{dfreq_i}{|D|}$ als Anpassungsfaktor verwendet werden.

$$P(k_i|R) = \frac{|V_i| + \frac{dfreq_i}{|D|}}{|V| + 1}$$

$$P(k_i|\overline{R}) = \frac{dfreq_i - |V_i| + \frac{dfreq_i}{|D|}}{|D| - |V| + 1}$$

Der Hauptvorteil des probabilistischen Modells besteht darin, dass die Dokumente nach ihrer Relevanz-Wahrscheinlichkeit für eine Suchanfrage sortiert werden [2].

Jedoch überwiegen die Nachteile. Diese könne wie folgt zusammengefasst werden [2]:

- Anfangs muss die Verteilung zwischen relevanten und nicht-relevanten Dokumenten geschätzt werden.
- Die Frequenz, mit der ein Indexterm auftritt wird nicht berücksichtigt.
- Indexterme werden voneinander unabhängig angesehen (in der Praxis nicht unbedingt ein Nachteil).

2.5.4 Vektorraummodell

Die Schwächen des Booleschen Modells und der exakten Übereinstimmung (Exact Matching) hat Salton erkannt und infolgedessen ein Modell vorgestellt, das die partielle Übereinstimmung (Partial Matching) zwischen einer Suchanfrage und einem Dokument berücksichtigt [9][13]. Das Vektorraummodell war geboren. Es bildet Suchanfragen und Dokumente als Vektoren in einem mehrdimensionalen Raum ab. In Abbildung 6 ist als Beispiel ein dreidimensionaler Vektorraum dargestellt. Dabei werden den Termen der Suchanfrage ebenfalls Gewichte zugeordnet. Dieses Modell wird zu den algebraischen Modellen gezählt.

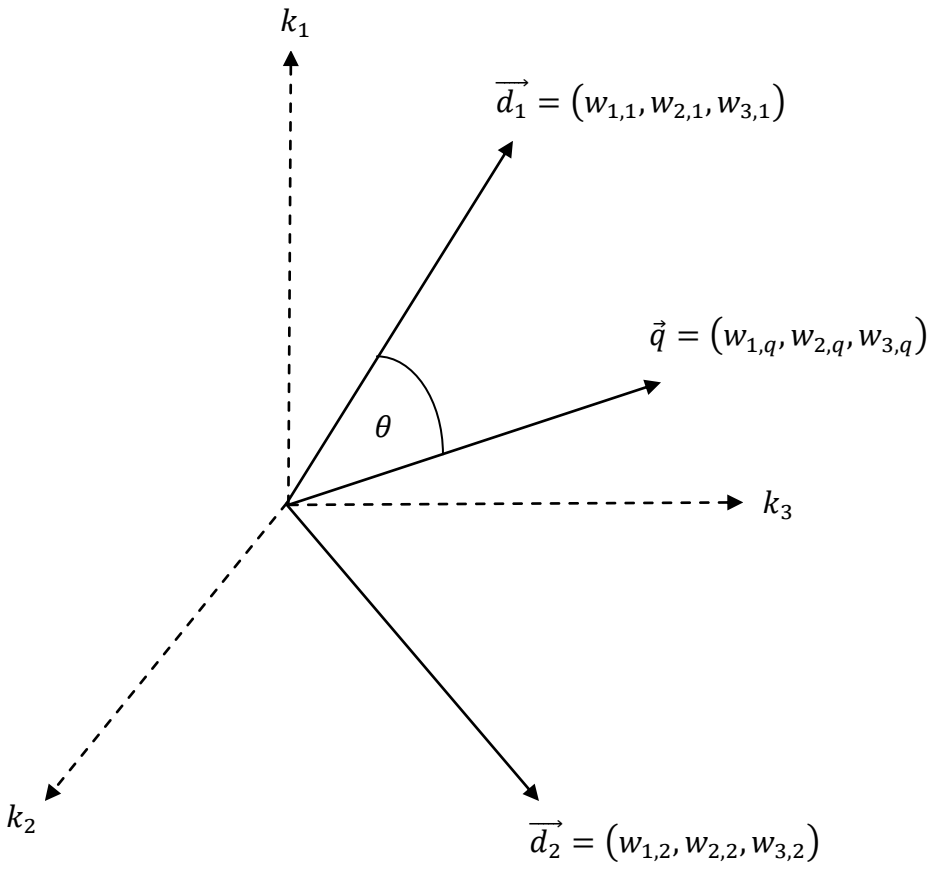


Abbildung 6: Dreidimensionaler Vektorraum [31]

Im Vergleich zum Booleschen Modell werden den Index- und Suchanfragetermen im Vektorraummodell nicht-binäre Gewichte ($w_{i,j} \geq 0, w_{i,\mathrm{q}} \geq 0$) zugewiesen. Wie diese Term-Gewichte ermittelt werden können, wird im Abschnitt 2.5.4.1 vorgestellt.

Diese Gewichte werden in Form der gewichteten Term-Vektoren dazu verwendet, die Ähnlichkeit zwischen Dokumenten bzw. zwischen Dokumenten und Suchanfragen zu ermitteln. Durch die nicht-binären Gewichte werden nun für eine Suchanfrage auch Dokumente gefunden, die nur teilweise mit der Suchanfrage

übereinstimmen. Die Ergebnisse der Suchanfrage sind infolgedessen nun unterschiedlich relevant. Durch einen festgelegten Schwellwert können die Dokumente im Suchergebnis begrenzt werden. Durch Sortierung der Suchergebnisse erfolgt ein Ranking der gefundenen Dokumente. Der Benutzer erhält auf eine Suchanfrage im Vektorraummodell viel präzisere Ergebnisse als im Booleschen Modell.

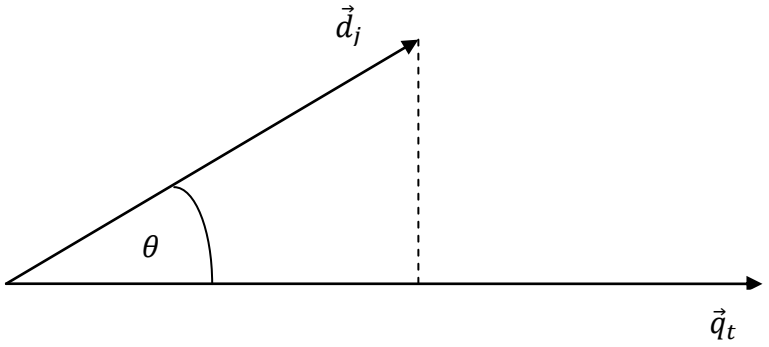


Abbildung 7: Cosinus-Maß zweier Vektoren [2]

Um die Ähnlichkeit zwischen einer Suchanfrage und den einzelnen Dokumenten zu ermitteln wird im Vektorraummodell die Korrelation der Term-Vektoren $\vec{d}_j$ und $\vec{q} = (w_{1,q}, \ldots, w_{|K|,q})$ berechnet. Dafür stehen mehrere Methoden zur Verfügung auf die in Abschnitt 2.5.4.2 näher eingegangen wird. Als Beispiel einer solchen Ähnlichkeitsfunktion kann das Cosinus-Maß verwendet werden:

$$sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{|K|} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{|K|} w_{i,q}^2}}$$

Dabei wird der Winkel zweier Vektoren (siehe Abbildung 7) als Maß ihrer Ähnlichkeit zueinander verwendet. Je kleiner dieser Winkel, desto größer ihre Ähnlichkeit und umgekehrt [2].

2.5.4.1 Gewichtungsfunktionen

Die Indexterm-Gewichtung kann auf unterschiedlichste Weise berechnet werden. Salton und McGill [14] haben in den frühen 60er-Jahren verschiedene dieser Gewichtungsmethoden untersucht.

Die grundlegende Idee der Indexterm-Gewichtung im Bereich Information Retrieval kann als Klassifikationsproblem gesehen werden. Bei einer Klassifikation müssen

zwei wichtige Aspekte berücksichtigt werden: Klassenähnlichkeiten und Klassenunähnlichkeiten.

Das bedeutet, es müssen für eine Dokumentklassifikation Eigenschaften gefunden werden, die ein Dokument besonders gut beschreiben, und Eigenschaften, die es erlauben, das Dokument besonders gut von anderen zu unterscheiden. Die effektivsten Klassifikationsalgorithmen versuchen diese beiden Aspekte möglichst gleich stark einfließen zu lassen. Ferber [4] nennt diese zwei Aspekte globale und lokale Einflussfaktoren.

Das Vektorraummodell bedient sich für die Indexterm-Gewichtung der sogenannten Term-Frequenz $tfreq_{i,j}$. Dieser Faktor beschreibt wie oft ein Term k_i in einem Dokument d_j vorkommt und somit wie gut dieser Term das Dokument beschreibt. Die Term-Frequenz zählt laut Ferber [4] zu den lokalen Einflussfaktoren. Die Struktur eines Dokuments bzw. die Position, in der der Term im Dokument vorkommt, sind weitere lokale Einflussfaktoren. Damit sehr lange Dokumente gegenüber kurzen nicht bevorteilt werden, wird die Term-Frequenz auf den am häufigsten auftretenden Term im Dokument normalisiert:

$$freqnorm_{i,j} = \frac{tfreq_{i,j}}{max(tfreq_{l,j})}$$

Um Dokumente möglichst gut zu unterscheiden, nutzt das Vektorraummodell für die Indexterm-Gewichtung die sogenannte Inverse Term-Frequenz. Diese zählt laut Ferber [4] zu den globalen Einflussfaktoren. Sie kann wie folgt berechnet werden:

$$idf_i = log\frac{|D|}{dfreq_i}$$

Die bekanntesten Indexterm-Gewichtungsfunktionen kombinieren lokale und globale Einflussfaktoren. Die grundlegende Gewichtungsfunktion sieht wie folgt aus:

$$w_{i,j} = freqnorm_{i,j} \times log\frac{|D|}{dfreq_i}$$

Salton und Buckley haben verschiedene Gewichtsfunktionen analysiert [15]. Im Allgemeinen bietet die hier vorgestellte Gewichtsfunktion sehr gute Ergebnisse. Für die Suchanfrage Term-Gewichtung haben Salton und Burckley eine andere Gewichtungsfunktion empfohlen:

$$w_{i,q} = (0.5 + \frac{0.5 \times tfreq_{i,q}}{max(tfreq_{l,q})}) \times log\frac{|D|}{dfreq_i}$$

Die Hauptvorteile des Vektorraummodells können wie folgt zusammengefasst werden:

- Die Term-Gewichtung führt zu höherer Qualität der Suchergebnisse.
- Durch die partielle Übereinstimmung zwischen Suchanfrage und Dokumenten werden auch Dokumente gefunden, die nicht exakt mit der Suchanfrage übereinstimmen.
- Die entsprechende Ähnlichkeitsfunktion sortiert die Dokumente nach Ähnlichkeit zur Suchanfrage.

Nicht nur wegen der gerade aufgezählten Vorteile, sondern auch wegen seiner Einfachheit und Schnelligkeit ist das Vektorraummodell das bekannteste Information-Retrieval-Modell in der heutigen Zeit. Im Vergleich zu anderen Modellen ist es entweder besser oder nahezu gleich gut als andere [2]. Die Steigerung der Qualität des Suchergebnisses im Vektorraummodell ist ohne sogenannter Query Expansion oder Relevance Feedback nur sehr schwierig zu erreichen.

Die Annahme, dass die Indexterme voneinander unabhängig sind, ist theoretisch ein Nachteil des Vektorraummodells. In der Praxis jedoch kann die Berücksichtigung der Indexterm-Abhängigkeiten die Qualität der Suchergebnisse sogar beeinträchtigen [2].

2.5.4.2 Ähnlichkeitsfunktionen

Im Vektorraummodell werden Suchanfragen und Dokumente als Vektoren im mehrdimensionalen Raum abgebildet. Um passende Dokumente für eine Suchanfrage oder ähnliche Dokumente zu finden, werden deren Vektoren verglichen. Dafür werden sogenannte Ähnlichkeitsfunktionen verwendet. Im Laufe der Zeit sind verschiedene Ähnlichkeitsmaße entwickelt und untersucht worden. Je nach verwendeter Methode beeinflussen andere Eigenschaften der Indexterm-Vektoren die Ähnlichkeit. Die Richtung eines Indexterm-Vektors kann möglicherweise auf das Thema eines Dokuments hinweisen und die Länge auf die Intensität [10]. Damit hängt die Wahl der Ähnlichkeitsfunktion stark mit der verwendeten Gewichtungsmethode, die für die Einträge im Indexterm-Vektor verantwortlich ist, zusammen [4].

Jones und Furnas [10] haben die Eigenschaften unterschiedlicher Ähnlichkeitsmaße beschrieben und für den zweidimensionalen Fall dargestellt. Einige ihrer Ergebnisse werden in folgenden Abschnitten erläutert.

2.5.4.2.1 Euklidischer Abstand

Der Euklidische Abstand dient dazu, den kürzesten Abstand zweier Vektoren im Raum zu bestimmen, er wird auch L_2-Abstand genannt. Umso kleiner der kürzeste Abstand zweier Vektoren ist, desto ähnlicher sind sie sich. Sind $\vec{d}_j$ und $\vec{q}$ zwei normierte Vektoren, liefert der Euklidische Abstand denselben Wert wie das Cosinus-Maß. Für $|K| = 2$ entspricht der Euklidische Abstand dem Satz des Pythagoras [3].

$$sim(\vec{d}_j, \vec{q}) = \sqrt{\sum_{i=1}^{|K|} (w_{i,j} - w_{i,q})^2}$$

2.5.4.2.2 Skalarprodukt

Das Skalarprodukt ist eine lineare Funktion. Wird einer der beiden zu vergleichenden Vektoren um einen Faktor vergrößert, vergrößert sich auch sein Ähnlichkeitsmaß. Laut Definition des Skalarprodukts, definiert das Quadrat der euklidischen Länge eines Vektors die Ähnlichkeit, die ein Vektor zu sich selbst hat.

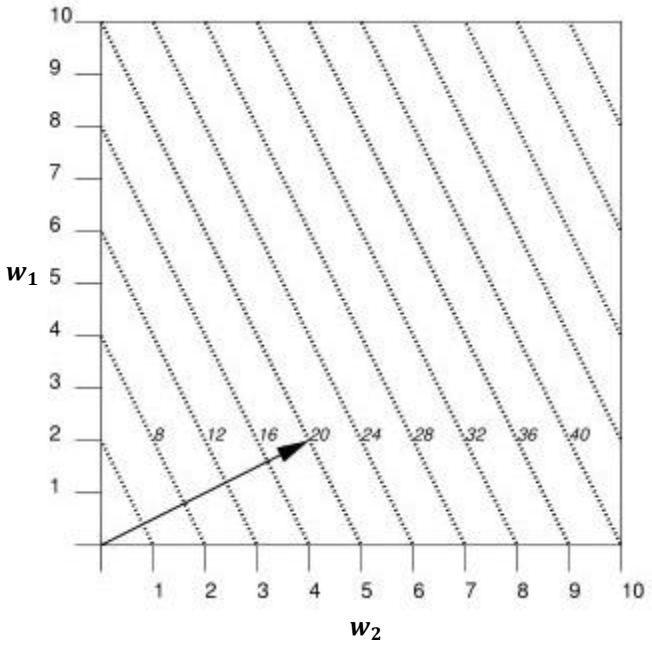


Abbildung 8: Skalarprodukt im zweidimensionalen Raum [4]

$$sim(\vec{d}_j, \vec{q}) = \sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}$$

Beim Skalarprodukt liegen Vektoren mit gleicher Ähnlichkeit zum Anfragevektor im zweidimensionalen Raum auf einer Gerade senkrecht zum Anfragevektor (siehe Abbildung 8). Beliebig große Ähnlichkeiten können mit dem Skalarprodukt erreicht werden [4].

2.5.4.2.3 Cosinus-Maß

Im Vergleich zum Skalarprodukt, hat die Länge der Vektoren beim Cosinus-Maß keinen Einfluss. Die Ähnlichkeit zweier Vektoren liegt im Allgemeinen im Intervall $[-1,1]$. Da es im Vektorraummodell keine negativen Gewichte gibt, liegt die Ähnlichkeit zweier Vektoren im Vektorraummodell immer im Intervall [0,1]

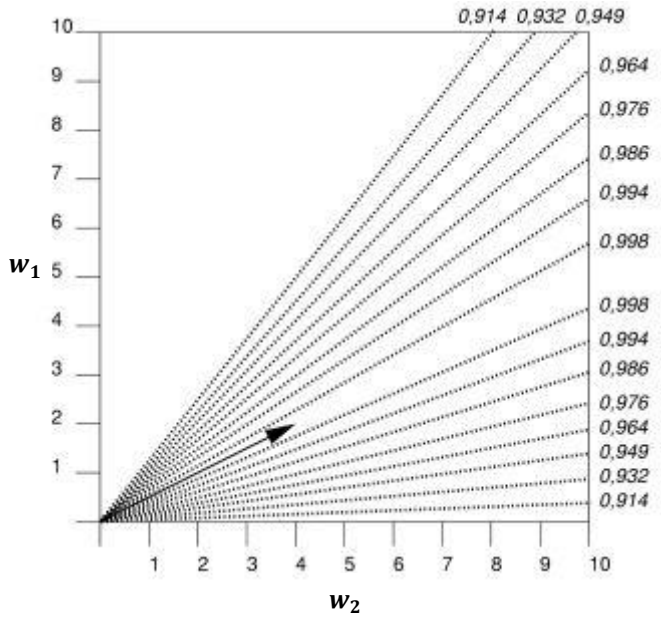


Abbildung 9: Cosinus-Maß im zweidimensionalen Raum [4]

$$\boldsymbol{k_1} \quad sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{|K|} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{|K|} w_{i,q}^2}}$$

Die Wurzeln im Nenner sind die euklidischen Längen der Vektoren. Aus diesem Grund verhält sich das Cosinus-Maß wie das Skalarprodukt mit normalisierten Vektoren. Damit hängt das Cosinus-Maß lediglich von der Richtung der Vektoren ab und nicht mehr von deren Länge. Zeigen zwei Vektoren also in dieselbe Richtung ist ihre Ähnlichkeit am größten. Bilden zwei Vektoren denselben Winkel zu einem Anfragevektor sind sie dem Anfragevektor gleich ähnlich (siehe Abbildung 9) [4].

2.5.4.2.4 Pseudo-Cosinus-Maß

Der Name Pseudo-Cosinus-Maß lässt eine gewisse Verwandtschaft zum Cosinus-Maß vermuten. In der Tat werden beim Pseudo-Cosinus-Maß lediglich die euklidischen Längen durch die Summe der Vektoreinträge (L_1-Länge) ersetzt. Dadurch kann auch das Pseudo-Cosinus-Maß als Skalarprodukt L_1-normierter Vektoren gesehen werden.

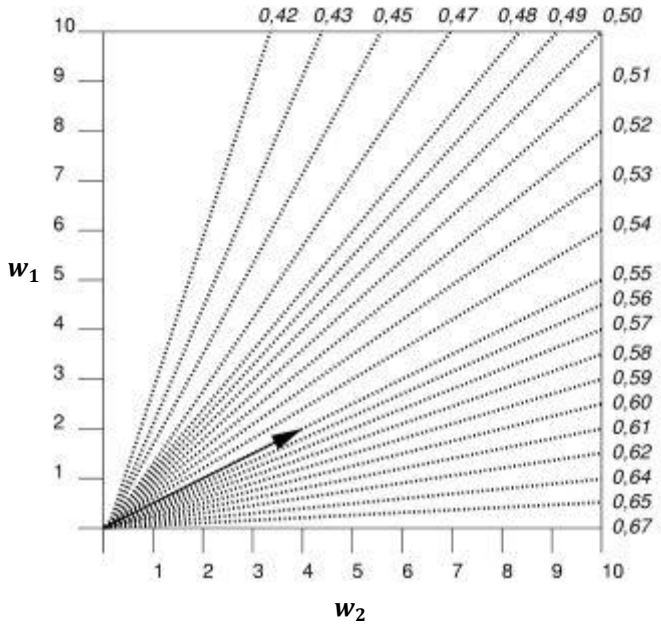


Abbildung 10: Pseudo-Cosinus-Maß im zweidimensionalen Raum [4]

$$sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{|K|} w_{i,j}} \times \sqrt{\sum_{i=1}^{|K|} w_{i,q}}}$$

Die Ähnlichkeit eines Vektors zum Anfragevektor hängt vom Winkel zum Anfragevektor ab. Jedoch haben zwei Vektoren, die mit dem Anfragevektor zwar denselben Winkel einschließen, unterschiedliche Ähnlichkeiten. Daher ist das Pseudo-Cosinus-Maß in Vergleich zum Cosinus-Maß (siehe Abbildung 9) nicht mehr symmetrisch. Die Ähnlichkeit eines Vektors zu einem Anfragevektor nimmt zu, wenn sich der Vektor auf die Achse des größten Eintrages des Anfragevektors zubewegt (siehe Abbildung 10). Große Einträge in den Vektoren spielen also beim Pseudo-Cosinus-Maß eine besondere Rolle [4].

2.5.4.2.5 Dice-Maß

Wie beim Cosinus-Maß und dem Pseudo-Cosinus-Maß entspringen die Ähnlichkeitsgeraden des Dice-Maßes in einem gemeinsamen Punkt. Dieser Punkt liegt aber nicht mehr im Ursprung sondern ist vom Anfragevektor abgängig (siehe Abbildung 11) [4].

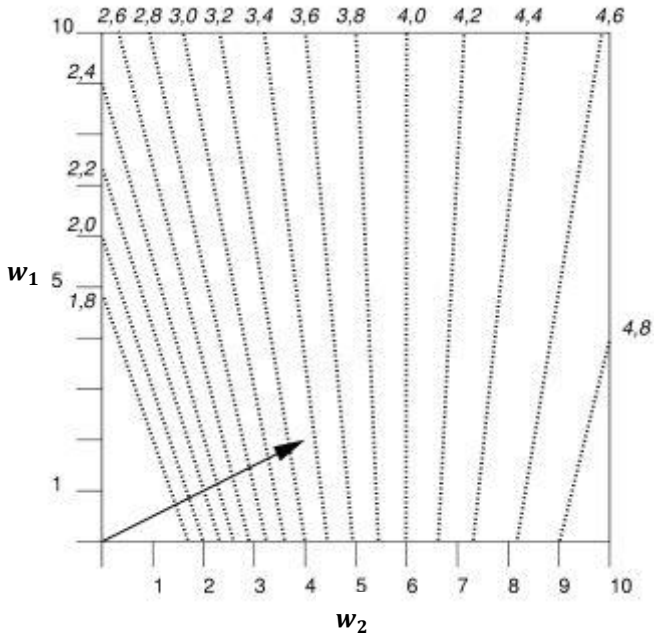


Abbildung 11: Dice-Maß im zweidimensionalen Raum [4]

$$sim(\vec{d}_j, \vec{q}) = \frac{2 \times \sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{|K|} w_{i,j}} + \sqrt{\sum_{i=1}^{|K|} w_{i,q}}}$$

2.5.4.2.6 Overlap-Maß

Beim Overlap-Maß liegen die Ähnlichkeitswerte im Intervall [0,1]. Berechnet wird es ähnlich zum Pseudo-Cosinus-Maß. Dabei wird die Multiplikation im Nenner des Pseudo-Cosinus-Maßes durch das Minimum der Summanden ersetzt.

Anstelle von simplen Geraden werden beim Overlap-Maß Vektoren gleicher Ähnlichkeit auf komplexere Flächen abgebildet (siehe Abbildung 12). Sind alle Einträge eines Vektors entweder kleiner oder größer als die des Anfragevektors, ist der Vektor beim Overlap-Maß dem Anfragevektor maximal ähnlich [4].

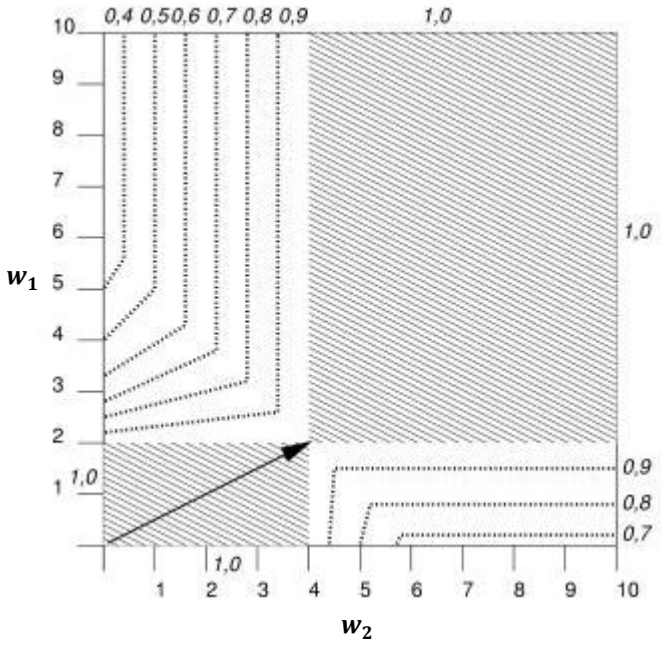


Abbildung 12: Overlap-Maß im zweidimensionalen Raum [4]

$$sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{|K|} min\,(w_{i,j}, w_{i,q})}{min\,(\sqrt{\sum_{i=1}^{|K|} w_{i,j}}, \sqrt{\sum_{i=1}^{|K|} w_{i,q}})}$$

2.5.4.2.7 Jaccard-Maß

Das Jaccard-Maß eignet sich gut für die Ähnlichkeitsberechnung von binären Vektoren. Dabei bildet der Zähler die größte Schnittmenge der Indexterme von Anfrage und Dokument und der Nenner die größte Vereinigungsmenge. Ähnlichkeitswerte liegen immer im Intervall [0,1].

Verwendet man jedoch allgemeine Vektoren, kann der Nenner des Jaccard-Maßes 0 werden. Das hat Unstetigkeiten in der Ähnlichkeit der Vektoren zur Folge. Somit können kleine Änderungen eines Vektors eine große Änderung der Ähnlichkeit bewirken. In Abbildung 13 sind die Ähnlichkeitsgeraden des Jaccard-Maßes dargestellt, welche dem Dice-Maß sehr ähnlich sehen. Im Vergleich zum Dice-Maß erlaubt das Jaccard-Maß jedoch auch negative Ähnlichkeiten. Die durchgezogene Gerade in Abbildung 13 kennzeichnet jene Vektoren, bei denen der Nenner des Jaccard-Maßes 0 wird. Das bedeutet, die Ähnlichkeiten von Vektoren auf dieser Gerade zum Referenzvektor sind undefiniert. Vektoren, die sich der durchgezogenen Linie von oben nähern, können eine beliebig negative Ähnlichkeit zum Referenzvektor erreichen. Das Gegenteil gilt für Vektoren, die sich der durchgezogenen Linie von unten nähern. Neben den Unstetigkeiten des Jaccard-Maßes für allgemeine Vektoren zeigt auch der Ähnlichkeitswert von -2,5 des

Referenzvektors aus Abbildung 13 zu sich selbst, dass dieses Maß für allgemeine Vektoren nahezu ungeeignet ist [4].

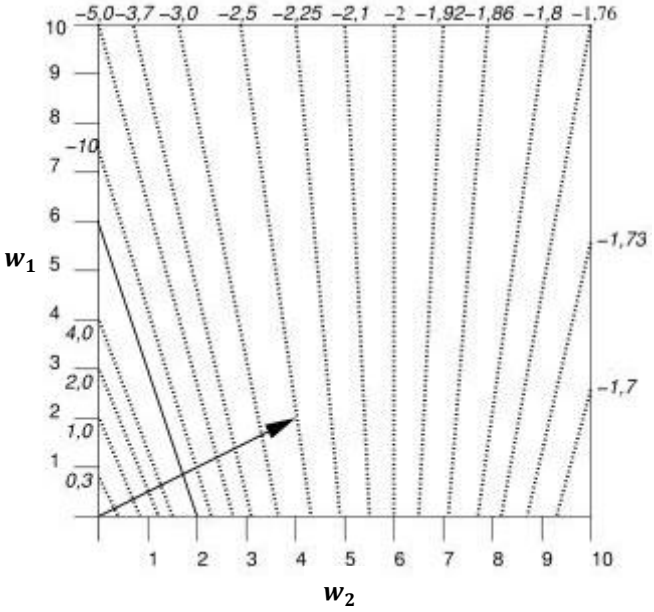


Abbildung 13: Jaccard-Maß im zweidimensionalen Raum [4]

$$sim(\vec{d}_j, \vec{q}) = \frac{\sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{|K|} w_{i,j}} + \sqrt{\sum_{i=1}^{|K|} w_{i,q}} - \sum_{i=1}^{|K|} w_{i,j} \times w_{i,q}}$$

3 Textklassifikation

Auf den ersten Blick wird häufig Textklassifikation mit Text-Clustering verwechselt, da in beiden Verfahren Dokumente gruppiert werden. Doch bei genauerer Betrachtung wird klar, dass zwischen beiden Verfahren ein fundamentaler Unterschied besteht. In diesem Kapitel wird kurz Klassifikation (siehe Abschnitt 3.2) und Clustering (siehe Abschnitt 3.1) gegenübergestellt und anschließend genauer auf Verfahren der Textklassifikation im Vektorraummodell eingegangen.

3.1 Clustering

Klassifikation und Clustering gehören beide in den Bereich maschinellen Lernens. Ziel beider Verfahren ist eine Menge von Dokumenten zu gruppieren. Im Bereich maschineller Lernverfahren werden Algorithmen grob in drei Kategorien eingeteilt:

- Überwachtes Lernen
- Unüberwachtes Lernen
- Reinforcement-Lernen

Beim überwachten Lernen stehen Referenzdaten zur Verfügung, die trainiert werden können. Das unüberwachte Lernen muss ohne Referenzdaten auskommen. Es erstellt aus den gegebenen Daten ein Modell, das möglichst gut beschreibt und vorhersagt. Einem etwas anderen Ansatz folgt das Reinforcement-Lernen. Dabei soll durch Bestrafung und Belohnung gelernt werden, wie in bestimmten Situationen zu entscheiden ist [4].

Clustering-Algorithmen zählen zu den unüberwachten Lernverfahren. Dabei sollen Dokumente so gruppiert werden, dass diese innerhalb einer Gruppe so ähnlich wie möglich, jedoch so unähnlich wie möglich zu Dokumenten anderer Gruppen sind. In Abbildung 14 wird ein einfaches Beispiel gezeigt, bei dem ein Mensch sofort die Gruppierung erkennt. Clustering-Algorithmen haben das Ziel, solche Gruppen ohne jegliches Vorwissen selbst zu finden. Werden die Suchergebnisse eines Information-Retrieval-Systems gruppiert, kann der Recall (siehe Abschnitt 4.1) des Systems verbessert werden.

Während beim Flat Clustering unabhängige Gruppen gesucht werden, wird beim Hierarchical Clustering versucht, die Gruppen untereinander hierarchisch zu strukturieren. Wird ein Dokument genau einer Gruppe zugewiesen, spricht man von Hard Clustering, ansonsten vom Soft Clustering [3].

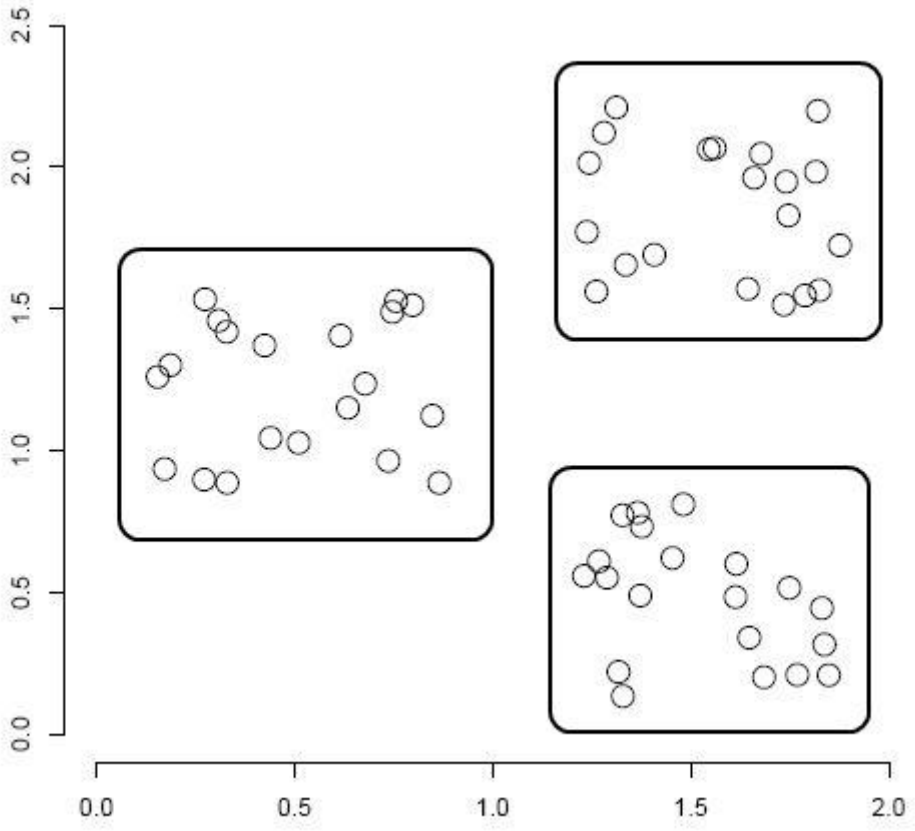


Abbildung 14: Zweidimensionaler Vektorraum mit klarer Cluster-Struktur [3]

3.2 Klassifikation

Eine Klassifikation von Dokumenten muss nicht zwangsweise von einem automatischen Algorithmus durchgeführt werden. Traditionell wurden Dokumente von Menschen manuell klassifiziert. Ein klassisches Beispiel ist die Einteilung von Büchern in einer Bibliothek zu bestimmten Fachgebieten. Diese Methode der Klassifikation ist mit wachsendem Dokumentenbestand sehr aufwendig und schwierig zu warten.

In diesem Abschnitt wird eine komfortablere Methode der Textklassifikation vorgestellt. Es handelt sich um die automatische Textklassifikation aus dem Bereich maschineller Lernverfahren. Basierend auf möglichst aussagekräftigen Trainingsdaten (überwachtes Lernen) erlernt der jeweilige Algorithmus Entscheidungskriterien, nach denen die Dokumente klassifiziert werden sollen. Die Trainingsdaten werden dabei weiterhin manuell von einem Menschen klassifiziert.

In folgenden Abschnitten werden Klassifikationsverfahren vorgestellt, die auf das Vektorraummodell aufbauen. Abhängig vom Term-Vektor $\vec{d}_j = (w_{1,j}, \ldots, w_{|K|,j})$ eines

Dokumentes werden diese im Vektorraum in eine bestimmte Region abgebildet. Eine solche Region entspricht einer Klasse. Die Menge aller Klassen $C = (c_1, \ldots, c_{|C|})$ im System ist durch die Trainingsdaten definiert. Die grundlegende Hypothese dabei lautet, dass Dokumente derselben Klasse im Vektorraum eine zusammenhängende Region bilden, die sich jedoch mit den Regionen anderer Dokumente nicht überschneidet. Ziel der Klassifikation ist es, die Grenzen dieser Regionen zu erlernen und durch eine Näherungsfunktion $\widetilde{\Phi}(j, i): d_j \times c_i \rightarrow \{True, False\}$ Dokumente der richtigen Klasse zuzuordnen [3].

3.2.1 Rocchio-Klassifikation

Die Rocchio-Klassifikation teilt den Vektorraum aufgrund der Trainingsdaten in unterschiedliche Regionen ein. Eine solche Region entspricht einer Klasse aus den Trainingsdaten. Die Grenzen der unterschiedlichen Regionen werden mittels Schwerpunktberechnung ermittelt. Dieser wird für jede Klasse aus den Trainingsdaten wie folgt berechnet:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d_j \in D_c} \vec{d}_j$$

Dokumente werden nun jener Klasse zugeordnet, dessen Schwerpunkt sie am ähnlichsten sind. In Abbildung 15 sind die schwarz ausgemalten Kreise die Schwerpunkte der jeweiligen Klasse. Der schwarze Stern markiert ein zu klassifizierendes Dokument und wird mit der Rocchio-Klassifikation der Klasse „China“ zugeordnet. Das schwarz gefärbte Quadrat markiert ein weiteres nicht klassifiziertes Dokument. Obwohl es aufgrund der Verteilung besser zur Klasse „UK“ gehören würde, wird es von der Rocchio-Klassifikation der Klasse „Kenya“ zugeordnet, da nur der Schwerpunkt der Klassen für die Klassifikation ausschlaggebend ist und nicht deren Verteilung. Welche Vektor-Ähnlichkeitsfunktion (siehe Abschnitt 2.5.4.2) dafür verwendet wird, ist frei wählbar.

Die Grenze zwischen zwei Klassen ist definiert durch eine Menge von Punkten, die von den beiden Schwerpunkten der Klassen denselben Abstand haben. In Abbildung 15 gilt $|a_1| = |a_2|$, $|b_1| = |b_2|$ und $|c_1| = |c_2|$. Dadurch sind drei Punkte auf einer Linie definiert, die die Klassen „UK“ und „Kenya“ abgrenzen. Allgemein kann man sagen, dass im zweidimensionalen Raum eine Linie die Klassen trennt, im dreidimensionalen Raum eine Ebene und im mehrdimensionalen Raum eine Hyperebene. Daraus ergibt sich für den mehrdimensionalen Raum allgemein folgender Ausdruck:

$$\vec{w}^T \vec{x} = b$$

Dabei ist $\vec{w}^T$ der Normalvektor der Hyperebene und b eine Konstante. Mit diesem Ausdruck können Linien, Ebenen sowie Hyperebenen beschrieben werden [3].

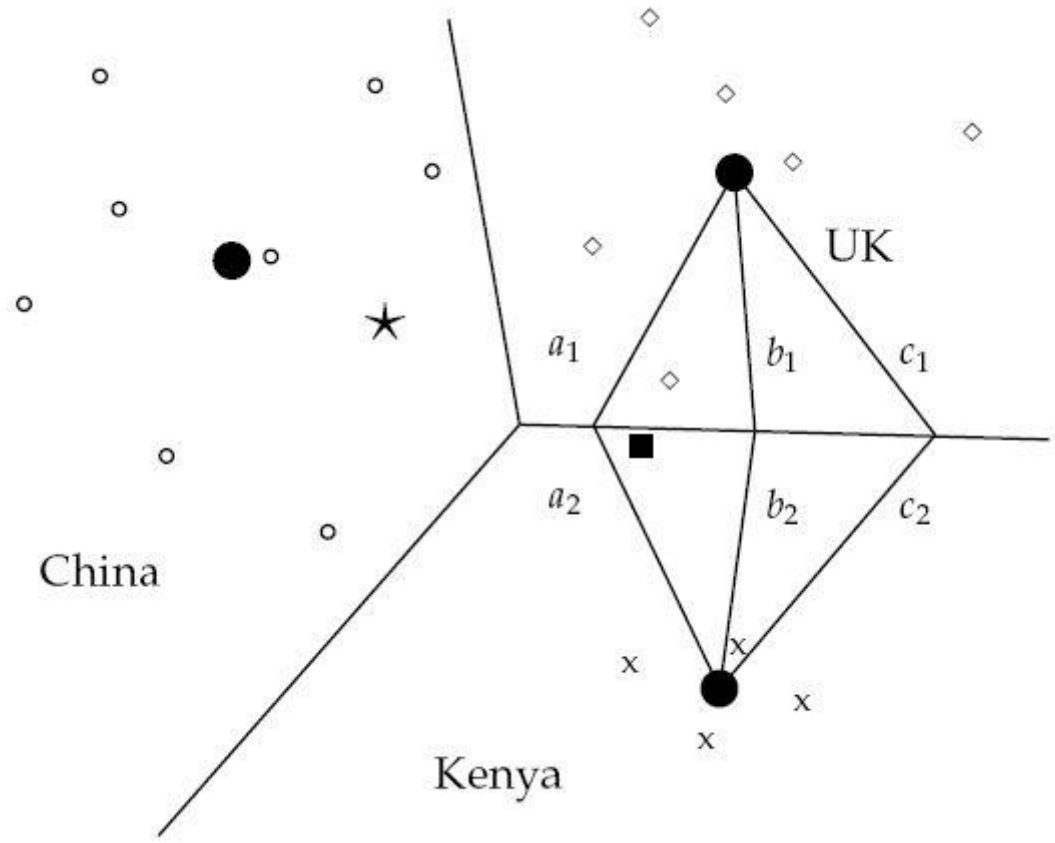


Abbildung 15: Rocchio-Klassifikation [3]

3.2.2 K-Nearest-Neighbour-Klassifikation

Anders als die Rocchio-Klassifikation ermittelt die K-Nearest-Neighbour-Klassifikation (K-NN) die Entscheidungsgrenzen zwischen den Klassen lokal, es werden also keine Parameter im Vorfeld berechnet. Es gibt keine Trainingsphase im herkömmlichen Sinn. Der Algorithmus merkt sich alle Klassifikationen aus der Trainingsmenge und vergleicht diese mit dem zu klassifizierenden Dokument. Aus diesem Grund wird die K-Nearest-Neighbour-Klassifikation zu den Memory-Based Learning-Verfahren gezählt. Durch Hinzufügen jedes neu klassifizierten Dokuments, wird die Trainingsmenge immer größer. Grundsätzlich ist es für jede Klassifikation besser, so viel wie möglich Trainingsdaten zu besitzen. Da die Testzeit jedoch linear von der Größe der Trainingsmenge abhängt, wird der Zeitaufwand der Klassifikation mit wachsender Trainingsmenge immer größer.

Während für 1-NN jedem Dokument die Klasse des ähnlichsten Dokuments zugeordnet wird, wird für K-NN jedem Dokument die Klasse zugeordnet, die unter den K ähnlichsten Dokumenten am häufigsten vorkommt. Der grundlegende

Gedanke der K-Nearest-Neighbour-Klassifikation ist, dass ein zu klassifizierendes Dokument d_j zur selben Klasse gehört wie jene Dokumente in dessen lokaler Umgebung.

Die Entscheidungsgrenzen für eine 1-NN-Klassifikation sind zusammenhängende Segmente eines sogenannten Voronoi-Mosaiks (siehe Abbildung 16). Ein Voronoi-Mosaik für eine Menge von Dokumenten teilt den Vektorraum in einzelne Voronoi-Zellen auf. Jedes Dokument entspricht genau einer solchen Zelle, dadurch ergeben sich $|D|$ verschiedene Zellen. Eine Voronoi-Zelle besteht aus allen Punkten, die näher zum eigenen Dokument als zu anderen Dokumenten sind. Dadurch wird im zweidimensionalen Raum die Ebene in $|D|$ konvexe Polygone und im mehrdimensionalen Raum in $|D|$ konvexe Polytope eingeteilt.

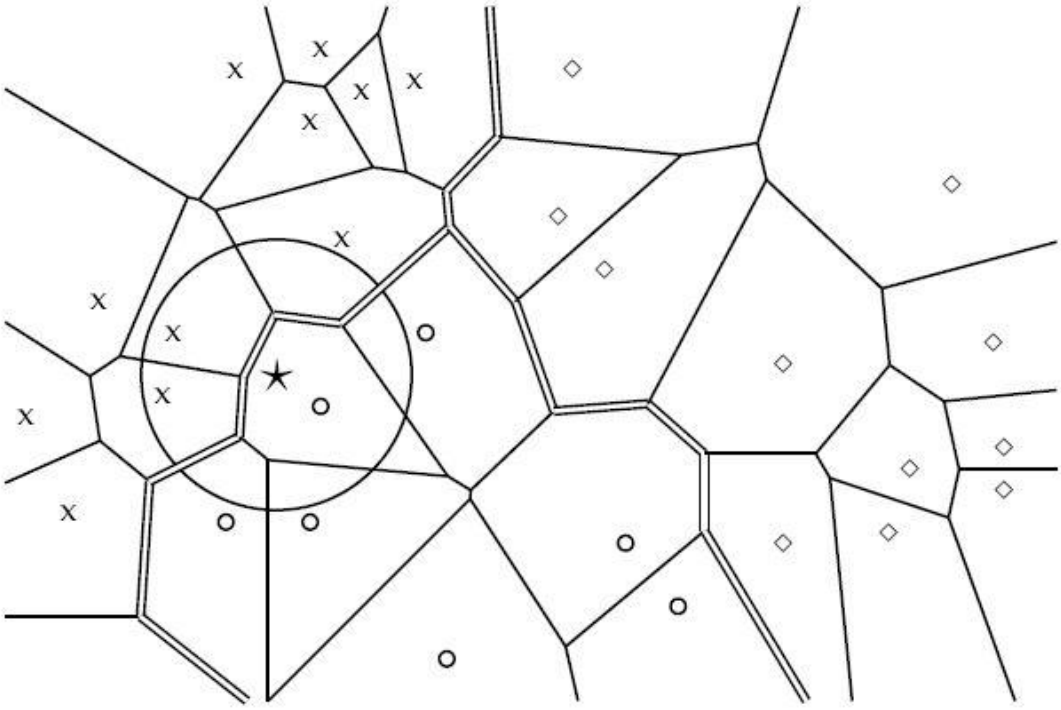


Abbildung 16: Voronoi-Zellen und Entscheidungsgrenzen für 1-NN [3]

Im Allgemeinen ist die 1-NN-Klassifikation nicht sehr robust, da die Entscheidung von nur einem Dokument in der Nachbarschaft abhängt und dieses könnte falsch klassifiziert sein. Für K-NN ist die Klassifikation robuster, da mehr Dokumente in der Nachbarschaft in den Entscheidungsprozess mit eingebunden werden.

Der Parameter K wird häufig empirisch oder durch Erfahrung gewählt. Damit immer eine eindeutige Entscheidung getroffen werden kann, muss der Parameter K ungerade sein. Häufig verwendete Werte sind 3 oder 5, aber auch weit höhere Werte

werden oft benutzt. Der Parameter K kann rechnerisch ermittelt werden, indem jenes K gewählt wird, das für eine Teilmenge der Trainingsdaten die besten Ergebnisse erzielt.

Um ähnlichere Nachbarn im Entscheidungsprozess zu favorisieren, können die einzelnen Nachbarn wie folgt gewichtet werden:

$$sim(c,d) = \sum_{d_j \in D_d} \mathrm{I_c(d_j)} \cos(\overrightarrow{d_J}, \vec{d})$$

Dabei ist $I_c(d_j) = 1$, falls das Dokument d_j zur Klasse c gehört, ansonsten $I_c(d_j) = 0$. Das Dokument wird der Klasse mit dem höchsten Ähnlichkeitswert zugeordnet.

4 Methoden der Evaluierung

Im Bereich Information Retrieval (siehe Kapitel 2) und Textklassifikation (siehe Kapitel 3) ist es nötig, neben Funktions- und Geschwindigkeits-Tests die Qualität der Ergebnisse einer Suchanfrage bzw. der Klassifikation von Dokumenten zu evaluieren. Diese Art der Evaluierung nennt man Retrieval-Performance-Evaluierung. Die Evaluierung dient dazu, verschiedene Systeme zu vergleichen, die verwendeten Information-Retrieval-Modelle zu bewerten und Änderungen der Modelle messbar zu machen. Dafür sind einheitliche Kennwerte, Suchanfragen und Referenzdaten wie beispielsweise die TREC[2]-Datensammlung nötig [2].

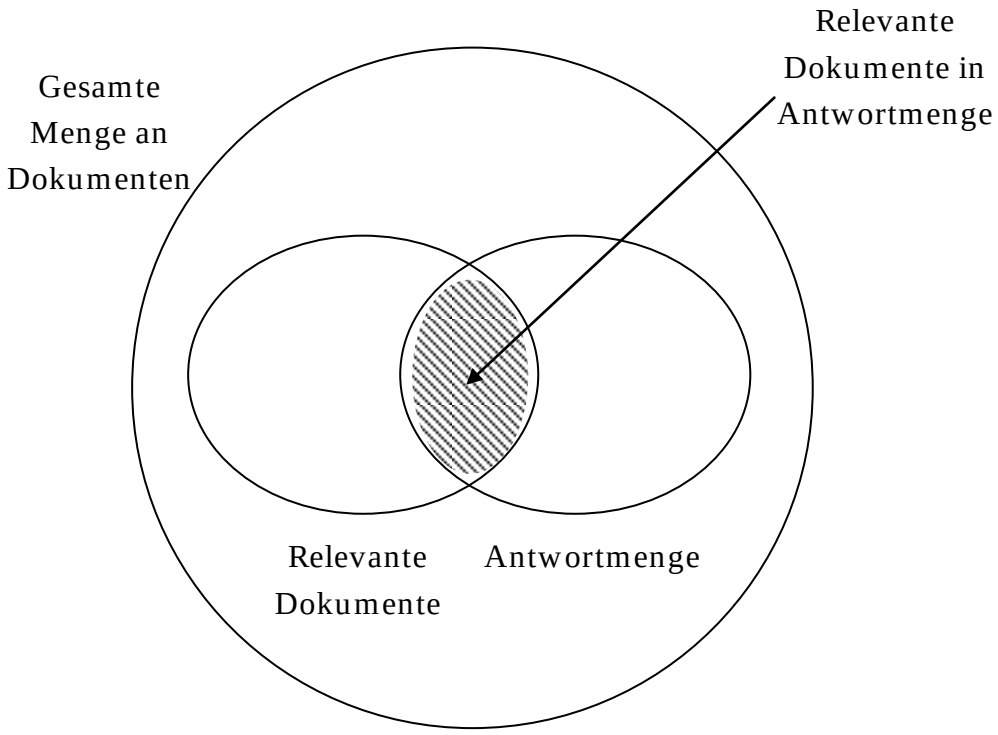


Abbildung 17: Antwortmenge für Information-Retrieval-Suchanfrage [2]

Formal sei $\tilde{\Phi}$ eine Näherungsfunktion, die binäre Entscheidungen trifft. Im Fall Information Retrieval entscheidet $\tilde{\Phi}$, ob ein Dokument d_j für eine Suchanfrage q_i relevant ist oder nicht ($\tilde{\Phi}(j,i): d_j \times q_i \rightarrow \{True, False\}$). In Abbildung 17 ist das Konzept Information Retrieval anhand eines Mengendiagramms dargestellt. Bei Textklassifikation entscheidet $\tilde{\Phi}$, ob ein Dokument d_j einer Klasse c_i richtig

[2]TREC steht für Text REtrieval Conference.

zugeordnet wurde oder nicht ($\widetilde{\Phi}(\mathrm{j},\mathrm{i}): d_j \times c_i \rightarrow \{True, False\}$). Die Zielfunktion $\Phi(\mathrm{j},\mathrm{i})$ soll dabei möglichst gut approximiert werden.

Suchanfrage q_i / Klasse c_i		**Zielfunktion Φ**	
		TRUE	FALSE
Suchfunktion $\widetilde{\Phi}$	TRUE	True Positive TP_i	False Positive FP_i
	FALSE	False Negative FN_i	True Negative TN_i

Tabelle 2: Möglichkeitstabelle für Evaluierung [11]

Information Retrieval und Textklassifikation bedienen sich ähnlicher Evaluierungsmethoden, die auf einer sogenannten Möglichkeitstabelle, wie in Tabelle 2 ersichtlich, beruhen.

Die lokale Möglichkeitstabelle für Information Retrieval wird wie folgt interpretiert:

- TP_i: Dokument d_j für Suchanfrage q_i relevant und wurde gefunden
- FP_i: Dokument d_j für Suchanfrage q_i nicht relevant, wurde aber gefunden
- FN_i: Dokument d_j für Suchanfrage q_i relevant, wurde aber nicht gefunden
- TN_i: Dokument d_j für Suchanfrage q_i nicht relevant und nicht gefunden

Die lokale Möglichkeitstabelle für Textklassifikation wird wie folgt interpretiert:

- TP_i: Dokument d_j gehört zur Klasse c_i und wurde dieser richtig zugeordnet
- FP_i: Dokument d_j gehört nicht zur Klasse c_i, wurde aber dieser zugeordnet
- FN_i: Dokument d_j gehört zur Klasse c_i, wurde aber dieser nicht zugeordnet
- TN_i: Dokument d_j gehört nicht zur Klasse c_i und wurde auch nicht zugeordnet

Die bekanntesten Kennwerte sind der Recall und die Precision. Diese werden in den folgenden Abschnitten vorgestellt [11].

4.1 Recall

Im Bereich Information Retrieval beschreibt der Recall den Anteil relevanter Dokumente am Suchergebnis. Es wird damit die Frage beantwortet, wie viele der gefundenen Dokumente für eine Suchanfrage q_i relevant sind [2].

$$Recall_i = \frac{TP_i}{(TP_i + FN_i)}$$

Für Textklassifikation wird der Recall als der Anteil richtig klassifizierter Dokumente einer Klasse c_i interpretiert [11].

4.2 Precision

Die Precision beschreibt im Bereich Information Retrieval den Anteil relevanter Dokumente am Suchergebnis gegenüber der Gesamtmenge relevanter Dokumente [2].

$$Precision_i = \frac{TP_i}{(TP_i + FP_i)}$$

Für Textklassifikation hingegen wird die Precision als Grad der Verlässlichkeit gesehen, dass ein einer Klasse zugeordnetes Dokument auch richtig zugeordnet wurde. Es wird damit die Frage beantwortet, wie verlässlich es ist, dass ein Dokument richtig klassifiziert wurde [11].

4.3 Accuracy und Error

Neben Recall und Precision können noch weitere Kennwerte abgeleitet werden, um die Qualität der Suchergebnisse bzw. die Klassifikation von Dokumenten eines Systems zu bewerten: Accuracy und Error [11].

$$Accuracy_i = \frac{TP_i + TN_i}{TP_i + FP_i + FN_i + TN_i}$$

$$Error_i = 1 - Accuracy_i$$

Die Accuracy wird hauptsächlich für das Bewerten von maschinellen Lernalgorithmen zur Textklassifikation verwendet. Im Bereich Information Retrieval gehören in den meisten Fällen 99,9 % der Dokumente zu den nicht-relevanten Dokumenten. Aus diesem Grund eignet sich Accuracy für Information Retrieval nicht besonders [3].

4.4 Fallout

Falls es in einem Information-Retrieval-System keine relevanten Dokumente gibt, ist der Recall undefiniert. Werden keine Dokumente für eine Suchanfrage gefunden oder keine Dokumente zu einer Klasse zugeordnet, ist auch die Precision undefiniert. Ein alternativer Kennwert ist der Fallout-Wert. Dieser berücksichtigt nicht-relevante Dokumente und kann wie folgt berechnet werden:

$$Fallout_i = \frac{FP_i}{(FP_i + TN_i)}$$

Dieser Wert kann als Gegenteil des Recall gesehen werden. Er bestimmt, wie wahrscheinlich es ist, ein nicht-relevantes Dokument zu finden. Der Fallout-Wert ist nur 0, falls alle gefundenen Dokumente relevant sind. Umso niedriger dieser Wert ist, desto besser [6].

4.5 Mittelwerte

Der Recall und die Precision sind lokale Kennwerte. Das bedeutet, sie sind von einer bestimmten Suchanfrage q_i oder Klasse c_i abhängig. Für den Vergleich verschiedener Systeme sind Recall und Precision ungeeignet. Um Systeme miteinander zu vergleichen, verwendet man globale Kennwerte. Man unterscheidet generell zwischen Mikromittelwert und Makromittelwert [11]. In diesem Abschnitt wird der Mikro- und Makromittelwert am Beispiel von Recall und Precision veranschaulicht. Es können jedoch auch von weiteren lokalen Kennwerten die folgenden Mittelwerte gebildet werden.

Für den Mikromittelwert wird eine globale Möglichkeitstabelle (siehe Tabelle 2) errechnet, d.h. es werden alle entsprechenden Suchanfragen bzw. Klassen berücksichtigt [11].

$$Recall^{\mu} = \frac{\sum_{i=1}^{|Q|} TP_i}{\sum_{i=1}^{|Q|}(TP_i + FN_i)} \; bzw. \; Recall^{\mu} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)}$$

$$Precision^{\mu} = \frac{\sum_{i=1}^{|Q|} TP_i}{\sum_{i=1}^{|Q|}(TP_i + FP_i)} \; bzw. \; Precision^{\mu} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)}$$

Der Mikromittelwert wird in [4] auch systemorientierte Bewertung genannt, da die relevanten Dokumente bzw. die richtig klassifizierten Dokumente die Basis der Bewertung sind.

Der Makromittelwert summiert alle entsprechenden lokalen Kennwerte auf und dividiert anschließend durch die gesamte Anzahl der Suchanfragen bzw. Klassen [11].

$$Recall^{M} = \frac{\sum_{i=1}^{|Q|} Recall_i}{|Q|} \; bzw. \; Recall^{M} = \frac{\sum_{i=1}^{|C|} Recall_i}{|C|}$$

$$Precision^M = \frac{\sum_{i=1}^{|Q|} Precision_i}{|Q|} \; bzw. \; Precision^M = \frac{\sum_{i=1}^{|C|} Precision_i}{|C|}$$

In [4] wird diese Art der Bewertung auch nutzungsorientiert genannt. Jede Suchanfrage bzw. Klassifikation fällt gleich stark ins Gewicht [4].

4.6 F-Maß und E-Maß

Recall und Precision im Einzelnen sind trotz ihrer Popularität nicht für jede Evaluierung die beste Wahl. In vielen Fällen ist einer der beiden Werte wichtiger als der andere. Ein Web-Surfer wünscht sich beispielsweise, dass eine Web-Suche auf der ersten Ergebnisseite ausschließlich relevante Dokumente findet (hohe Precision). Ein Benutzer, der seine Festplatte durchsucht, ist eher an einem hohen Recall interessiert. Es muss immer ein Kompromiss zwischen Recall und Precision eingegangen werden, da die Erhöhung eines Wertes die Verringerung des anderen mit sich bringt [3].

Aus diesem Grund wurden im Laufe der Zeit alternative Kennwerte entwickelt, die es erlauben, Recall und Precision unterschiedlich gewichtet zu kombinieren [2].

Das F-Maß ist der gewichtete harmonische Mittelwert zwischen Precision und Recall.

$$F(i) = \frac{1}{\alpha \frac{1}{Precison_i} + (1-\alpha)\frac{1}{Recall_i}}$$

Für

$$\beta^2 = \frac{1-\alpha}{\alpha}$$

gilt:

$$F(i) = \frac{(\beta^2+1) \times Precison_i \times Recall_i}{\beta^2 \times Precison_i + Recall_i}$$

Für α gilt $\alpha \epsilon [0,1]$ und für β gilt $\beta^2 \epsilon [0,\infty]$. Recall und Precision werden für $\alpha = \frac{1}{2}$ bzw. $\beta = 1$ gleich stark gewichtet. Im ausbalancierten Fall wird das F-Maß üblicherweise mit $F_{\beta=1} = F_1$ bezeichnet und die Berechnung vereinfacht sich dafür wie folgt [3]:

$$F_1(i) = \frac{2 \times Precison_i \times Recall_i}{Precison_i + Recall_i}$$

Ein anderes Maß, welches Recall und Precision kombiniert, ist das sogenannte E-Maß. Es wurde von Rijsbergen [19] erstmals im Jahre 1979 vorgestellt.

$$E(i) = 1 - \frac{1 + b^2}{\frac{b^2}{Recall_i} + \frac{1}{Precison_i}}$$

Wie beim F-Maß kann der Benutzer die Gewichtung zwischen Recall und Precision festlegen. Für $b = 1$ verhält sich das E-Maß wie das Kompliment vom F-Maß. Wird $b > 1$ gewählt, favorisiert der Benutzer die Precision, ansonsten den Recall.

4.7 Precision-Recall-Diagramm

Der Recall und die Precision werden für die Evaluierung unsortierter Ergebnismengen verwendet. Dafür muss die gesamte Antwortmenge berücksichtigt werden und es kann keine Aussage getroffen werden, welches Dokument d_j bzw. welche Klasse c_j in der Ergebnismenge relevanter ist. Da man nicht immer die gesamte Antwortmenge berücksichtigen will oder dies nicht möglich ist bzw. in der Ergebnismenge relevantere Einträge als erstes erhalten will, wird die Ergebnismengen nach Relevanz absteigend sortiert. Beim Betrachten der absteigend sortierten Ergebnismenge ändert sich das Empfinden des Benutzers über Recall und Precision. Die Idee ist nun, den Recall und die Precision für ausgewählte Punkte festzuhalten und sie in einem Precision-Recall-Diagramm darzustellen.

Üblicherweise wird in einem Information-Retrieval-System eine bestimmte Anzahl der gefundenen Dokumente sortiert zurückgeliefert. Um daraus das Precision-Recall-Diagramm zu erhalten, werden für jedes Dokument jeweils der Recall und die Precision berechnet und als Diagramm wie in Abbildung 18 dargestellt.

Das Ergebnis ist meist eine Sägezahnkurve. Das kommt daher, da die Precision sinkt und der Recall gleich bleibt, falls das nächste betrachtete Dokument nicht relevant ist. Die Precision und der Recall steigen jedoch, falls das nächste betrachtete Dokument relevant ist. In vielen Fällen möchte man dieses Rauschen unterdrücken, indem man die Precision wie folgt interpoliert:

$$Precision_{interp}(Recall) = max_{Recall' \geq Recall} Precision(Recall')$$

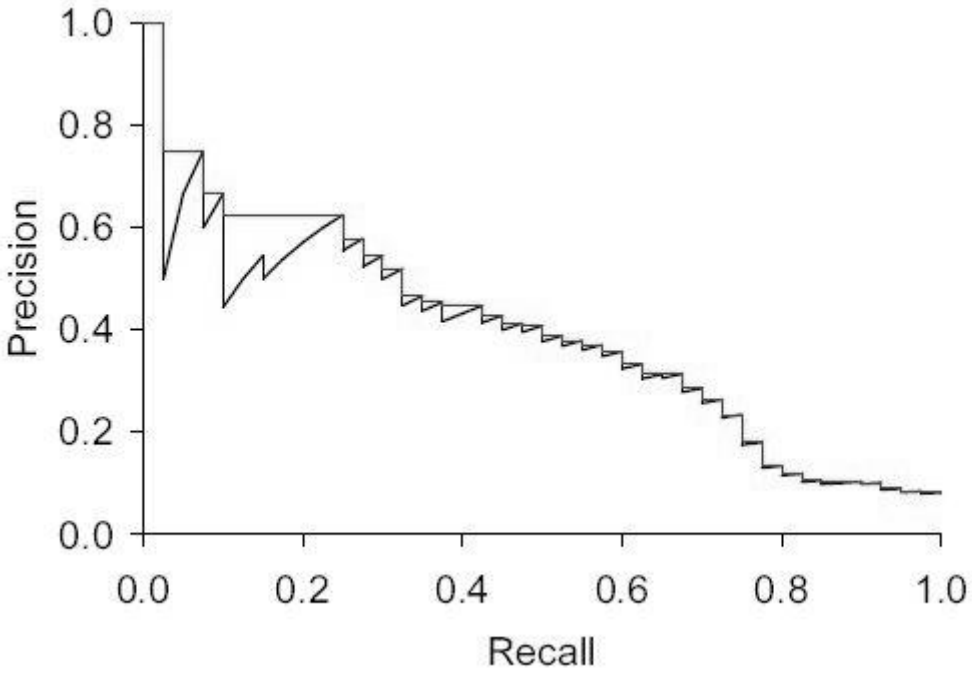


Abbildung 18: Interpoliertes Precision-Recall-Diagramm [3]

Die Begründung für diese Definition liegt darin, dass viele Benutzer weitere Dokumente betrachten würden, um einen höheren Anteil relevanter Dokumente zu finden.

Um die Bewertung von unterschiedlichen Precision-Recall-Diagrammen zu vereinfachen, hat man sich auf 11 Standard Recall Levels $\{0\%, 10\%, 20\%, \ldots, 100\%\}$ geeinigt (siehe Abbildung 19). Da in der Praxis die Werte sehr wahrscheinlich nicht auf diesen Punkten liegen werden, werden die zuvor interpolierten Werte für die 11 Standard Recall Levels arithmetisch gemittelt.

In Abbildung 20 ist dargestellt, wie ein ideales Precision-Recall-Diagramm aussieht. Solange relevante Dokumente gefunden werden, bleibt die Precision maximal. Der Recall startet bei 0 und steigt stetig bis alle relevanten Dokumente gefunden wurden. Sind alle relevanten Dokumente gefunden, können nur mehr nicht-relevante Dokumente gefunden werden. Der Recall bleibt ab diesem Zeitpunkt maximal, da dieser von gefundenen nicht-relevanten Dokumenten nicht beeinflusst wird. Die Precision hingegen nähert sich jedoch der Null-Linie an [6].

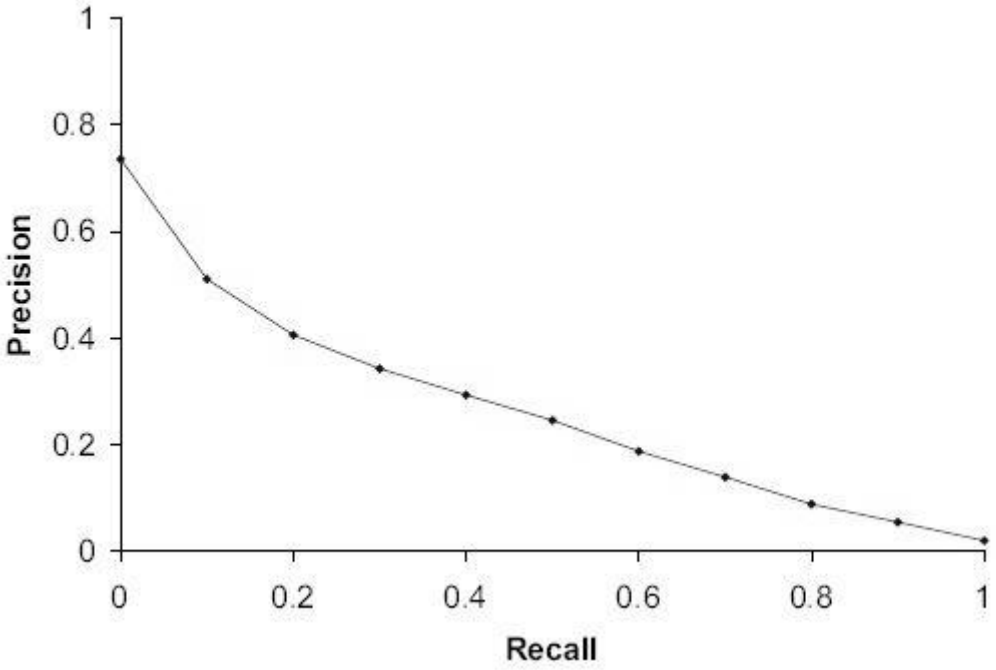


Abbildung 19: Precision-Recall-Diagramm für 11-Standard-Recall-Levels [3]

Abbildung 20: Ideales Precision-Recall-Diagramm [6]

5 JCOP-Wissensmanagementsystem

Im Rahmen dieser Arbeit wurde für das Unternehmen NXP Semiconductors ein Wissensmanagementsystem entwickelt, um den Zugang der JCOP-Spezifikationen zu optimieren. Dieses Kapitel behandelt alle praktischen Aspekte, die zur Entwicklung dieses Systems nötig waren. Dazu gehört die Ausarbeitung der Anwendungsfälle (siehe Abschnitt 5.1), die Entwicklung einer Systemarchitektur (siehe Abschnitt 5.2), eines geeigneten Datenbankschemas (siehe Abschnitt 5.3) und einer geeigneten Indexstruktur (siehe Abschnitt 5.5). Die verwendeten Algorithmen für die Volltextsuche und die Klassifikation der Dokumente werden im Abschnitt 5.7. Das resultierende System wird von NXP Semiconductors produktiv zur Verwaltung der JCOP-Spezifikationen eingesetzt.

5.1 Anwendungsfälle

Aus den funktionalen Anforderungen (siehe Abschnitt 1.4.2) an das System lassen sich unterschiedliche Anwendungsfälle für den Benutzer ableiten. Diese sind in Abbildung 21 zusammengefasst.

Um den Benutzer den Zugriff auf das JCOP-Wissensmanagementsystem und die Wartung von unterschiedlichen Standorten zu gewähren, ist das System als Web-Applikation implementiert.

Die zentrale Aufgabe des Systems ist es, JCOP-Spezifikationen verwalten zu können. Das System erlaubt es dem Benutzer Spezifikationen mit Metadaten im System anzulegen, zu verwalten und zu verknüpfen. Zwischen Spezifikationen können beliebig (gerichtete) Beziehungen definiert werden, um schneller erkennen zu können, welche Spezifikation von welchen anderen und in welcher Form abhängen. Durch einen gerichteten Graphen werden diese Beziehungen visualisiert. Damit der Benutzer die Übersicht über alle Spezifikationen behält, erlaubt das System die Einbettung der Spezifikationen in eine vom Benutzer definierte virtuelle Ordnerstruktur. Alle vom Benutzer ins System eingetragenen Daten können nachträglich verändert bzw. gelöscht werden.

Ist für eine Spezifikation die entsprechende PDF-Datei verfügbar, kann diese zu den Metadaten der entsprechenden Spezifikation hinzugefügt und ins System hochgeladen werden. Nachträglich kann die PDF-Datei einer Spezifikation gelöscht oder mit einer anderen überschrieben werden. Wurde für eine Spezifikationen eine PDF-Datei ins System hochgeladen, kann diese vom Benutzer zum Index

hinzugefügt bzw. vom Index entfernt werden. Eine Indexoptimierung erlaubt es dem Benutzer den Index automatisch aufzuräumen, um so die Reaktionszeit der Volltextsuche zu steigern.

Durch Formulierung einer Suchanfrage kann der Benutzer den Volltext der indizierten Spezifikationen durchsuchen. Die besten 10 Treffer werden sortiert nach Relevanz dargestellt. Für jede indizierte Spezifikation kann das System, basierend auf dem Inhalt der PDF-Datei, nach ähnlichen Spezifikationen durchsucht werden. Die 10 besten Treffer werden dem Benutzer zurückgeliefert. Mittels einer automatischen Klassifikation kann der Benutzer für jede Spezifikation überprüfen, wo in der virtuellen Ordnerstruktur das System die Spezifikation zuordnen würde. Über einen RSS Feed wird der Benutzer benachrichtigt, falls sich die Metadaten einer Spezifikation verändert haben.

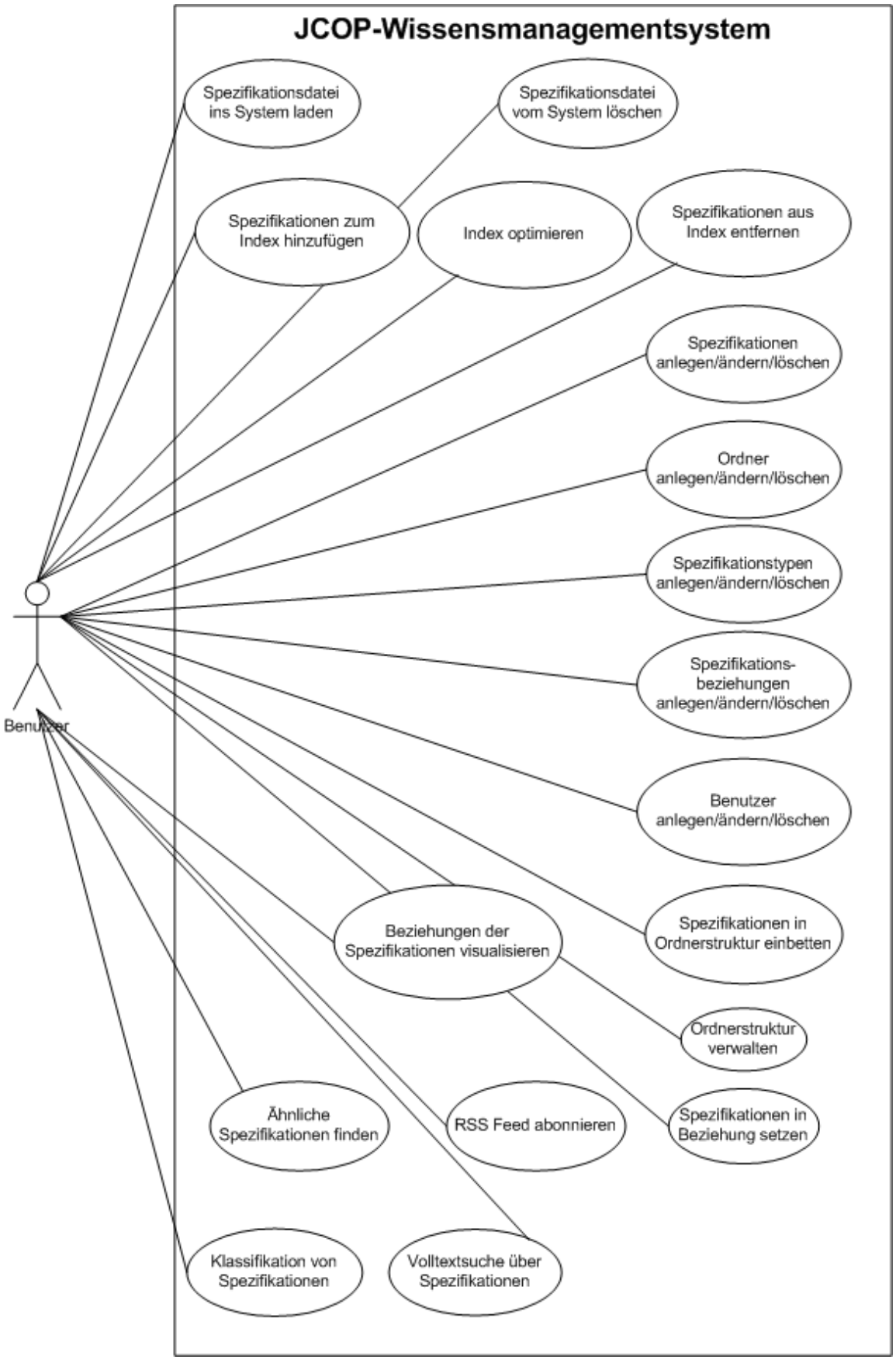


Abbildung 21: Anwendungsfälle

5.2 Systemarchitektur

Das JCOP-Wissensmanagementsystem ist eine webbasierte PHP-Applikation, welche den Zugang zu JCOP-Spezifikationen optimiert. Anfragen eines Clients werden via HTTP Request an den Server gesendet und das Ergebnis als HTML in Form eines HTTP Response an den Client zurückgeliefert. Der Kern der Applikation ist ein in PHP geschriebenes MVC Web Framework namens Symfony Version 1.2.8 (http:/ / www.symfony-project.org) und eine PHP Search Engine namens Lucene aus dem Zend Framework Version 1.7.8 (http:/ / framework.zend.com). Für die

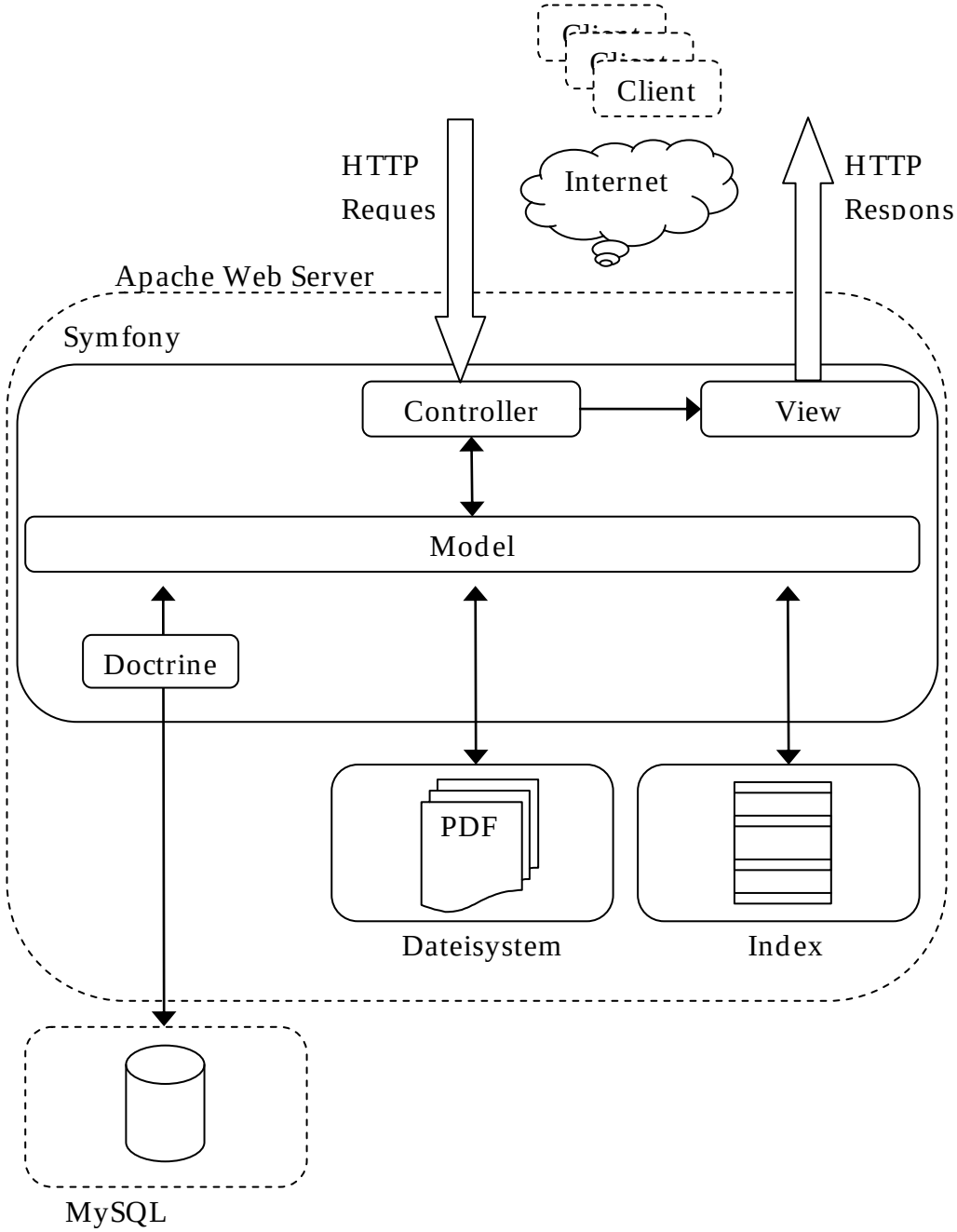


Abbildung 22: Systemarchitektur

Visualisierung der Spezifikationsbeziehungen bzw. der virtuellen Ordnerstruktur am Client wurde JavaScript InfoVis Toolkit Version 1.1.3 (http:/ / thejit.org) bzw. dTree Version 2.05 (http:/ / destroydrop.com/ javascripts/ tree) verwendet.

Der Benutzer kann JCOP-Spezifikationen zusammen mit einer dazugehörigen PDF-Datei ins System einpflegen. Dabei wird die Datei im Dateisystem des Webservers abgelegt. Der Inhalt der PDF-Dateien wird mit dem XPDF-Tools Version 3.02 (http:/ / www.foolabs.com/ xpdf) extrahiert, mit dem in Lucene integrierten StandardAnalyzer (http:/ / codefury.net/ projects/ standardanalyzer) gefiltert und im Index abgelegt. Damit wird eine Volltextsuche, Ähnlichkeitssuche und die Klassifikation von Dokumenten ermöglicht. JCOP-Spezifikationen können vom Benutzer mit Metadaten versehen werden. All diese Metadaten werden in einer MySQL-Datenbank verwaltet und neben dem jeweiligen Inhalt des Dokuments im Index abgelegt. Zur Kommunikation mit der Datenbank wird ein Object-Relational Mapper (ORM) von Symfony namens Doctrine verwendet. In Abbildung 22 ist die gesamte System Architektur dargestellt.

Alle verwendeten Tools nochmals im Überblick:

- Symfony Version 1.2.8 (http:/ / www.symfony-project.org)
- Zend Framework Version 1.7.8 (http:/ / framework.zend.com)
- JavaScript InfoVis Toolkit Version 1.1.3 (http:/ / thejit.org)
- XPDF Version 3.02 (http:/ / www.foolabs.com/ xpdf)
- dTree Version 2.05 (http:/ / destroydrop.com/ javascripts/ tree)
- StandardAnalyzer (http:/ / codefury.net/ projects/ standardanalyzer)

5.3 Datenbankschema

Um die Spezifikationen und die Benutzer des JCOP-Wissensmanagementsystems zu verwalten, wird eine relationale Datenbank verwendet. Diese ist in neun Tabellen organisiert (siehe Abbildung 23 und Abbildung 24).

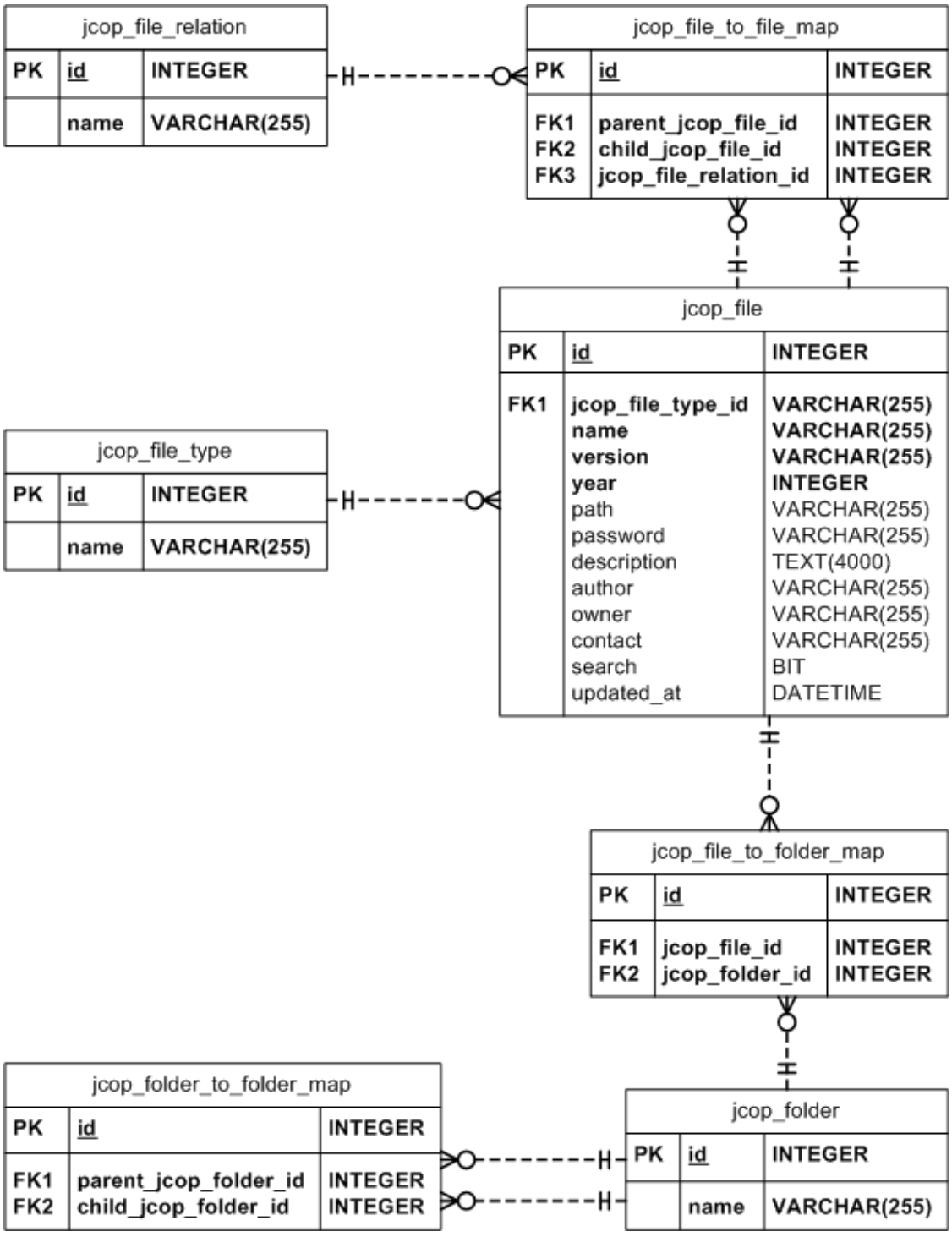


Abbildung 23: Datenbankschema für die Spezifikationsverwaltung

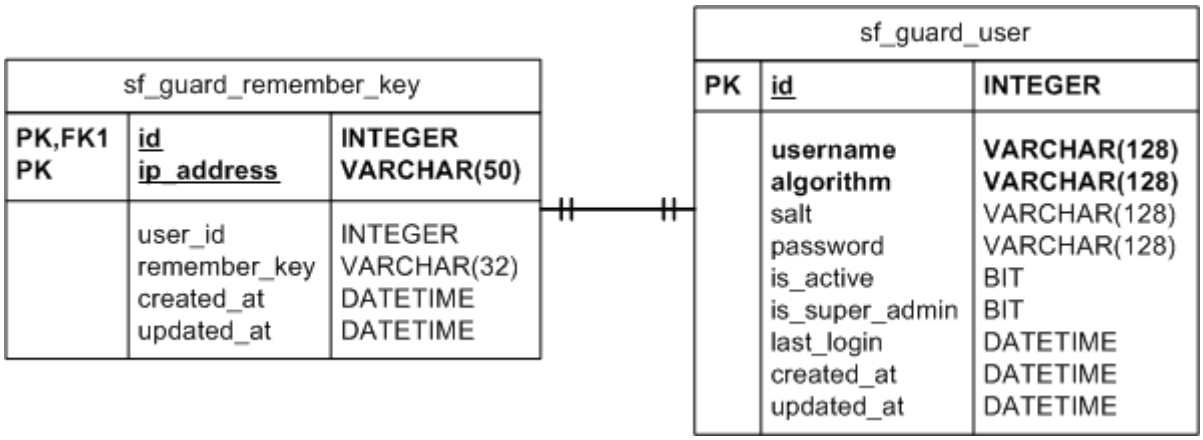


Abbildung 24: Datenbankschema für die Benutzerverwaltung

Das Kernstück der Datenbank ist die jcop_file-Tabelle. Diese beinhaltet die Metadaten aller Spezifikationen. Zu den Pflichtfeldern gehören das name-Feld, das version-Feld, das year-Feld und das jcop_file_type_id-Feld. Weitere interessante Felder in der jcop_file-Tabelle sind das path-Feld, das password-Feld und das search-Feld. Falls eine Spezifikation eine dazugehörende PDF-Datei besitzt, wird im path-Feld ihr Dateiname gespeichert. Für eine PDF-Datei mit Passwortschutz, kann im password-Feld das Passwort gespeichert werden. Das search-Feld ist auf 1 gesetzt, falls eine Spezifikation indiziert wurde, ansonsten ist es auf 0 gesetzt. Nicht jede Spezifikation kann indiziert werden. Manche bestehen nur aus Bildern oder sind passwortgeschützt und können nicht in Text umgewandelt werden.

Alle möglichen Spezifikationstypen werden in einer getrennten Tabelle namens jcop_file_type verwaltet und bestehen lediglich aus einem Namen. Jeder Spezifikation kann genau nur ein Spezifikationstyp zugeteilt werden.

Die virtuelle Ordnerhierarchie des Systems bzw. welche Spezifikationen zu welchen Ordnern gehören, wird nicht im Dateisystem des Servers abgebildet sondern ausschließlich virtuell in der Datenbank.

In der jcop_folder-Tabelle werden die Namen aller Ordner im System verwaltet. Die virtuelle Ordnerhierarchie wird in der jcop_folder_to_folder_map-Tabelle gespeichert. Dabei kann ein Ordner mehrere Unterordner haben bzw. selbst Unterordner von mehreren anderen Ordnern sein (n:n-Beziehung).

Die Beziehung, welche Spezifikationen zu welchen Ordnern zugeordnet sind, erfolgt in der jcop_file_to_folde_map-Tabelle. Dabei können einem Ordner mehrere Spezifikationen zugeordnet werden bzw. kann eine Spezifikation mehreren verschiedenen Ordner gehören (n:n-Beziehung).

Verschiedene Spezifikationen können zu einander in Beziehung stehen. Diese Abbildung erfolgt in der jcop_file_to_file_map-Tabelle. Der Beziehungstyp wird ebenfalls in dieser Tabelle gespeichert. Mehrere Spezifikationen können auf dieselbe Spezifikation verweisen bzw. eine Spezifikation kann auf mehrere andere verweisen (n:n-Beziehung).

Alle Beziehungstypen sind in der jcop_file_relation-Tabelle gespeichert und bestehen lediglich aus einem Namen.

Für die Benutzerverwaltung werden die sf_guard_user-Tabellen und sf_guard_user_remember-Tabelle verwendet. Jeder Benutzer erhält einen eindeutigen Benutzernamen. Das jeweilige Passwort wird verschlüsselt gespeichert. Optional kann ein Benutzer eingeloggt bleiben. Dafür werden der Benutzername und die dazugehörige IP-Adresse gespeichert.

5.4 Index-Feldtypen

Im Index können fünf verschiedene Typen von Feldern gespeichert werden (siehe Tabelle 3). Jedes dieser Felder hat unterschiedliche Eigenschaften.

Feldtyp	Gespeichert	Indiziert	In Tokens zerlegt	Binär
Keyword	Ja	Ja	Nein	Nein
UnIndexed	Ja	Nein	Nein	Nein
Binary	Ja	Nein	Nein	Ja
Text	Ja	Ja	Ja	Nein
UnStored	Nein	Ja	Ja	Nein

Tabelle 3: Index-Feldtypen [31]

Keyword-Felder werden indiziert und gespeichert und werden somit im Suchergebnis mitgeliefert. Informationen wie Zeitstempel und Primärschlüssel müssen meist nicht durchsuchbar sein. Es genügt, wenn sie im Suchergebnis mit geliefert werden. Für solche Daten sind UnIndexed-Felder geeignet. Jegliche Art von binären Daten kann in Binary-Felder gespeichert werden. Diese werden weder indiziert noch in Tokens aufgeteilt, sondern ausschließlich gespeichert und werden ebenfalls im Suchergebnis zurückgeliefert. Informationen wie Themen und Überschriften werden üblicherweise in Text-Felder gespeichert. Diese werden in Tokens zerlegt, gespeichert und indiziert. Umfangreiche Texte, die durchsuchbar sein sollen, aber nicht wieder ausgegeben werden müssen, können in UnStored-Felder gespeichert werden.

5.5 Indexstruktur

Im Index des JCOP-Wissensmanagementsystems werden für jede indizierbare Spezifikation der Primärschlüssel, der Volltext, alle Metadaten sowie Ordner- und Spezifikationsbeziehungen eingetragen. Zu Beginn der Implementierung wurde die Indexstruktur in Tabelle 4 verwendet. Da diese Indexstruktur zu schlechten Ergebnissen der Klassifikation führte (siehe Abschnitt 6.2), wurde diese währender der Implementierung erweitert (siehe Tabelle 5).

Um die zu den Spezifikationen im Ergebnis der Volltextsuche passenden Metadaten aus der Datenbank zu finden, wird der Primärschlüssel im Suchergebnis mitgeliefert. Daraus ergibt sich, dass der Primärschlüssel einer Spezifikation im Index in einem UnIndexed-Feld eingetragen wird.

Feldname	Feldtyp	Beschreibung
pk	UnIndexed	Primärschlüssel der Spezifikation
content	UnStored	Volltext der Spezifikation aus PDF-Datei

Tabelle 4: Kleine Indexstruktur

Der Volltext der Dokumente sowie alle Metadaten müssen nur durchsuchbar sein. Aus diesem Grund werden der Volltext und die Metadaten als UnStored-Felder im Index gespeichert.

Feldname	Feldtyp	Beschreibung
pk	UnIndexed	Primärschlüssel der Spezifikation
content	UnStored	Volltext der Spezifikation aus PDF-Datei
type	UnStored	Typ der Spezifikation
name	UnStored	Name der Spezifikation
version	UnStored	Version der Spezifikation
year	UnStored	Erscheinungsjahr der Spezifikation
description	UnStored	Beschreibung der Spezifikation
author	UnStored	Autor der Spezifikation
owner	UnStored	Besitzer der Spezifikation
contact	UnStored	Kontakt der Spezifikation
parentfolders	UnStored	Namen der Ordner der Spezifikation
parentfiles	UnStored	Namen der übergeordneten Spezifikationen
childfiles	UnStored	Namen der untergeordneten Spezifikationen

Tabelle 5: Große Indexstruktur

5.6 Dateiverwaltung

Alle Spezifikationsdateien werden im uploads-Verzeichnis des Webwurzel-Verzeichnisses im Dateisystem des Webservers hinterlegt. Damit jede hochgeladene Spezifikationsdatei einen eindeutigen Namen erhält, wird ein SHA-1-Hashwert aus dem Dateinamen, Dateiendung und einer Zufallszahl zwischen 11111 und 99999 berechnet und als neuer Dateiname verwendet:

$$hashwert = sha1(dateiname, dateiendung, zufallszahl)$$

Dieser ist immer unterschiedlich, auch wenn dieselbe Datei mehrmals hochgeladen wird. In der Datenbank wird der berechnete Dateiname in der jcop_file-Tabelle im path-Feld hinterlegt. Eine bereits vorhandene Spezifikationsdatei wird beim erneuten Hochladen einer Datei vorher aus dem Dateisystem gelöscht. Dadurch sind im Dateisystem des Webservers keine veralteten oder nicht mehr benutze Spezifikationsdateien zu finden. Es werden ausschließlich PDF-Dateien mit einer Dateigröße bis zu 100 Megabyte akzeptiert.

5.7 Algorithmen

In diesem Abschnitt werden anhand von Zustandsdiagrammen die verwendeten Algorithmen für die Volltextsuche (siehe Abschnitt 5.7.1), Indizierung (siehe Abschnitt 5.7.2), Indexterm-Vektor-Generierung (siehe Abschnitt 5.7.3), Ähnlichkeitssuche (siehe Abschnitt 5.7.4) und Klassifikation (siehe Abschnitt 5.7.5) erläutert.

5.7.1 Volltextsuche

Für die Volltextsuche wird der Standardalgorithmus von Lucene verwendet. Lucene berechnet die Relevanz eines Dokuments d für eine Suchanfrage q wie folgt:

$$sim(q,d) = sum\big(tf(t\ in\ d) * idf(t) * getBoost(t.field\ in\ d) * lengthNorm(t.field\ in\ d)\big) * coord(q,d) * queryNorm(q)$$

Dabei ist die Gewichtungsfunktion schon integriert und sieht wie folgt aus:

$$w_{t,d} = tf(t\ in\ d) * idf(t) * getBoost(t.field\ in\ d) * lengthNorm(t.field\ in\ d)$$

Die Term-Frequenz eines Terms wird dabei mit der Inverse-Term-Frequenz, dem Boost-Faktor und dem Normierungswert multipliziert und aufsummiert. Der Boost-Faktor steigert die Relevanz eines Indexfeldes und ist standardmäßig 1. Der Coord-Faktor favorisiert Dokumente, die Teile der Suchanfrage enthalten. Anschließend wird noch ein Normierungswert für die Suchabfrage multipliziert. Dieser

beeinträchtigt das Ranking der Dokumente nicht, dient jedoch dazu verschiedene Suchanfragen vergleichbar zu machen. Nähere Informationen wie die einzelnen Werte berechnet werden, kann auf der Webseite vom Zend Framework (http:/ / framework.zend.com/ manual/ de/ zend.search.lucene.extending.html) nachgelesen werden.

5.7.2 Indizierung

Wird für eine Spezifikation eine PDF-Datei ins System hochgeladen, kann diese nachträglich indiziert werden. Es ist auch möglich nachträgliche einzelne indizierte Spezifikationen aus dem Index zu löschen bzw. wieder hinzuzufügen.

Bei der Indizierung einer Spezifikation muss diese, falls sie schon im Index enthalten ist, aus dem Index gelöscht werden. Der Grund dafür liegt darin, dass das Zend Framework das Aktualisieren eines Dokumentes im Index nicht unterstützt. Nach Entfernen des Dokumentes aus dem Index wird die Spezifikation in der Datenbank (search-Feld) als nicht indiziert markiert. Die zu indizierende PDF-Datei wird dann in Text konvertiert und gemeinsam mit allen Metadaten (siehe Abschnitt 5.5) in den Index eingetragen. In der Datenbank wird die Spezifikation wieder als indiziert markiert. Falls die PDF-Datei mit einem Passwort geschützt ist bzw. nur Bilder enthält, kann keine Textdatei erstellt werden und die Datei ist somit nicht indizierbar.

Der zu indizierende Text wird von Lucene mit Hilfe des StandardAnalyzers (siehe Abschnitt 5.2) verschiedenen Textoperationen unterzogen. Zuerst wird der Text in Terme zerlegt. Dabei gilt jedes Zeichen, das weder eine Zahl oder ein Buchstabe ist, als Trennzeichen. Alle so erhaltenen Terme werden dann in Kleinbuchstaben umgewandelt und jene Terme, die kürzer als drei Zeichen sind, gelöscht. Dann werden alle Stoppwörter (siehe Anhang A) aus den übrigen Termen entfernt und mittels Porter-Stemming-Algorithmus (siehe 2.4.3.1) auf ihren Wortstamm reduziert. Alle übrigen Terme werden indiziert.

Für die Ähnlichkeitssuche bzw. die Klassifikation von Dokumenten wird der Term-Vektor eines Dokumentes benötigt. Da das Zend Framework die Term-Vektor-Generierung eines im Index enthaltenen Dokumentes nicht unterstützt, musste eine Möglichkeit gefunden werden den Term-Vektor zu generieren. Um die Geschwindigkeit der Datenbankzugriffe bzw. die Suche im Index nicht zu beeinträchtigen, wird der Text der Spezifikation nicht in der Datenbank bzw. im Index (content-Feld ist vom Typ UnStored) gespeichert. Als Lösung dieses Problems wird vor der eigentlichen Lucene Indizierung einer Spezifikation der extrahierte Text jeweils in einer Textdatei gespeichert. Diese Textdatei wird dann denselben

Textoperationen (außer Porter-Stemming-Algorithmus) unterzogen, die bei der eigentlichen Lucene Indizierung verwendet werden. Anschließend wird für jeden Term die Term-Frequenz errechnet und zusammen in einer Datei gespeichert. Dieser Schritt kann auch als K-Nearest-Neighbour-Vorverarbeitung gesehen werden. In Abbildung 25 ist der gesamte Indizierungsprozess zusammengefasst.

Abbildung 25: Zustandsdiagramm für Indizierung

5.7.3 Indexterm-Vektor

Die Term-Vektor-Generierung dient dazu, ähnliche Dokumente zu finden und in einem weiteren Schritt Dokumente zu klassifizieren (siehe Abbildung 26). Als erstes wird für das entsprechende Dokument jene Datei eingelesen, die vor der Lucene Indizierung mit allen gefilterten Termen und Term-Frequenzen angelegt wurde. Falls keine solche Datei vorhanden ist, konnte das Dokument nicht indiziert werden und somit ist auch die Term-Vektor-Generierung nicht möglich. Für die Gewichtung der Terme wird die Lucene Gewichtungsfunktion verwendet (siehe 5.7.1). Anschließend werden alle Terme nach Relevanz sortiert in einem Vektor zusammengefasst. Dieser Vektor wird dann auf eine bestimmte Länge gekürzt. Der resultierende Vektor stellt den Term-Vektor eines Dokumentes dar und wird für die Ähnlichkeitssuche bzw. Klassifikation von Dokumenten verwendet.

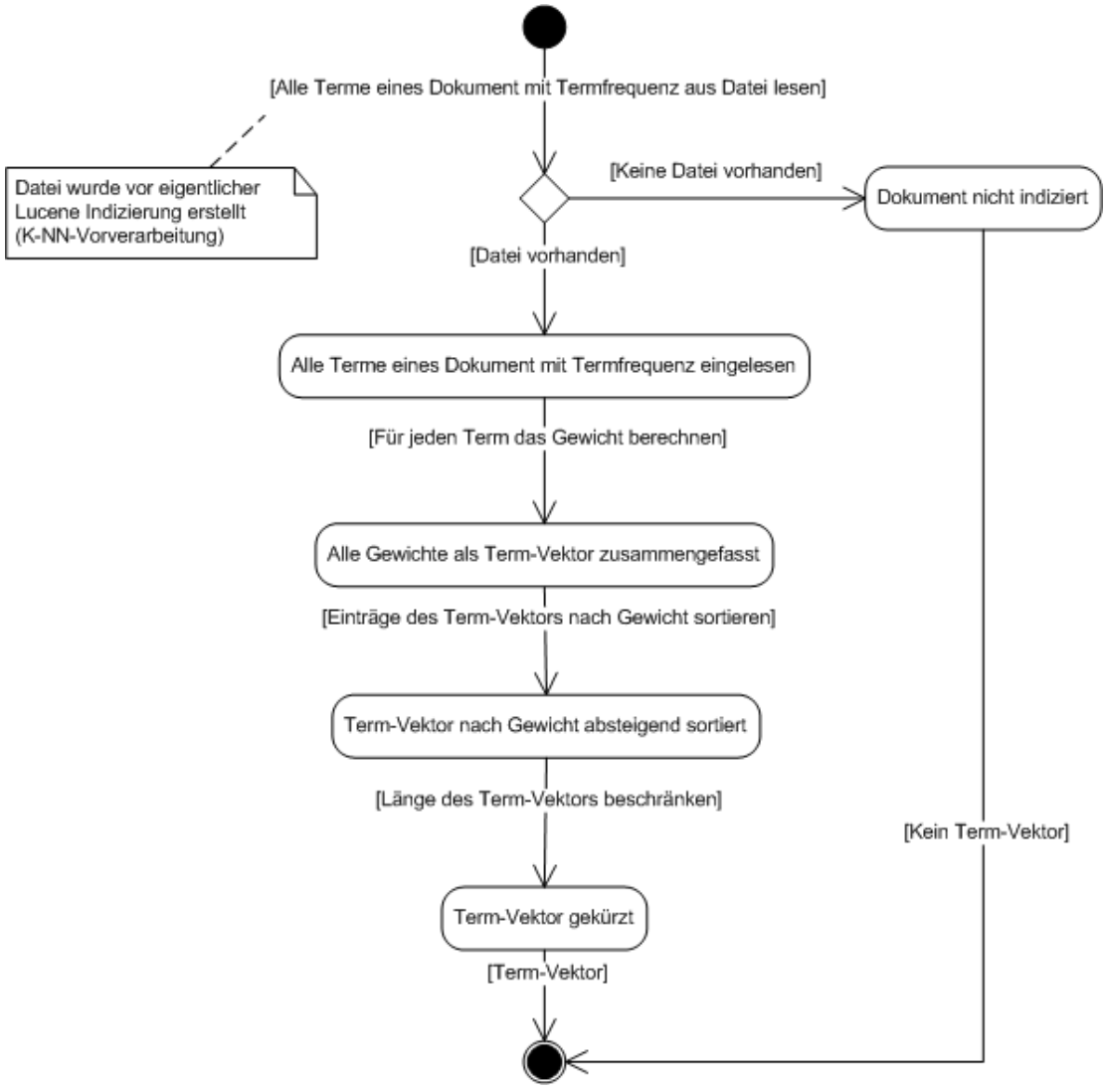


Abbildung 26: Zustandsdiagramm für Indexterm-Vektor

5.7.4 Dokumentenähnlichkeit

Um für ein Dokument ähnliche Dokumente zu finden, wird der Term-Vektor des Dokumentes berechnet (siehe Abschnitt 5.7.3). Der Term-Vektor erhält jene Terme, die das Dokument am besten beschreiben und sich am besten zur Unterscheidung anderer Dokumente eignen. Falls kein Term-Vektor generiert werden kann, ist das Dokument nicht indiziert oder nicht indizierbar und es können somit auch keine ähnlichen Dokumente gesucht werden. Alle Term-Vektor-Einträge werden mit dem logischen Operator „OR“ verknüpft und als Suchanfrage zusammengefasst. Diese Suchanfrage wird mittels Lucene Suche (siehe Abschnitt 5.7.1) ausgeführt. Die Ergebnisse dieser Volltextsuche stellen die ähnlichsten Dokumente dar. In Abbildung 27 wird der Ablauf der Ähnlichkeitssuche von Dokumenten als Zustandsdiagramm dargestellt.

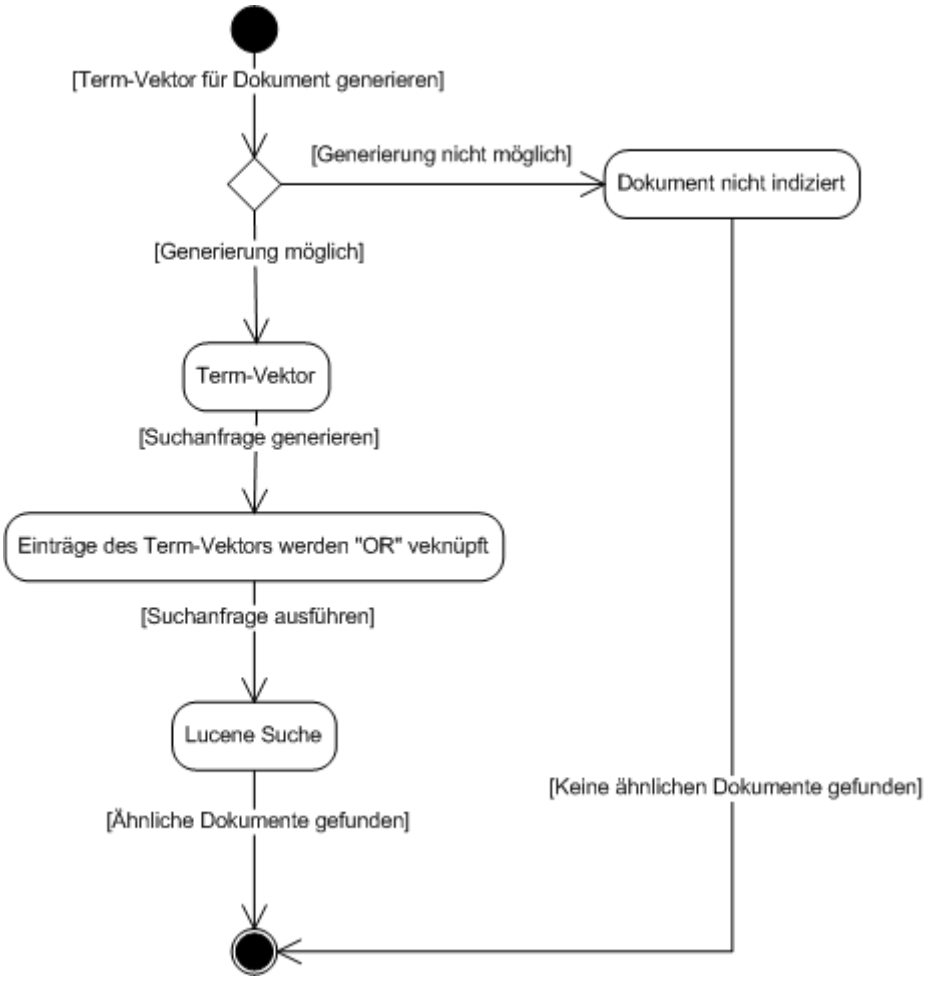


Abbildung 27: Zustandsdiagramm für Dokumentenähnlichkeit

5.7.5 Klassifikation

Aufgrund der schlechten Ergebnisse der Klassifikation basierend auf der kleine bzw. der große Indexstruktur (siehe Abschnitt 6.2) vereint die K-Nearest-Neighbour-Klassifikation zwei unterschiedliche Aspekte (siehe Abbildung 28). Zum einen werden die K ähnlichsten Dokumente basierend auf der große Indexstruktur (siehe Abschnitt 5.7.4) ermittelt und zum anderen werden die K ähnlichsten Dokumente rein basierend auf den vom Benutzer vergebenen Namen der Spezifikationen ermittelt. Für die letzteres wird die PHP-Funktion similar_text (http:/ / php.net/ manual/ en/ function.similar-text.php) verwendet. In beiden Fällen werden für jedes der K Dokumente die zugeordneten Ordner betrachtet und daraus die jeweilige Ordnerhäufigkeit berechnet. Anschließend wird der arithmetische Mittelwert der Ordnerhäufigkeiten zwischen beiden Fällen berechnet. Der Ordner mit der höchsten Häufigkeit wird als empfohlene Klasse für das Dokument vorgeschlagen.

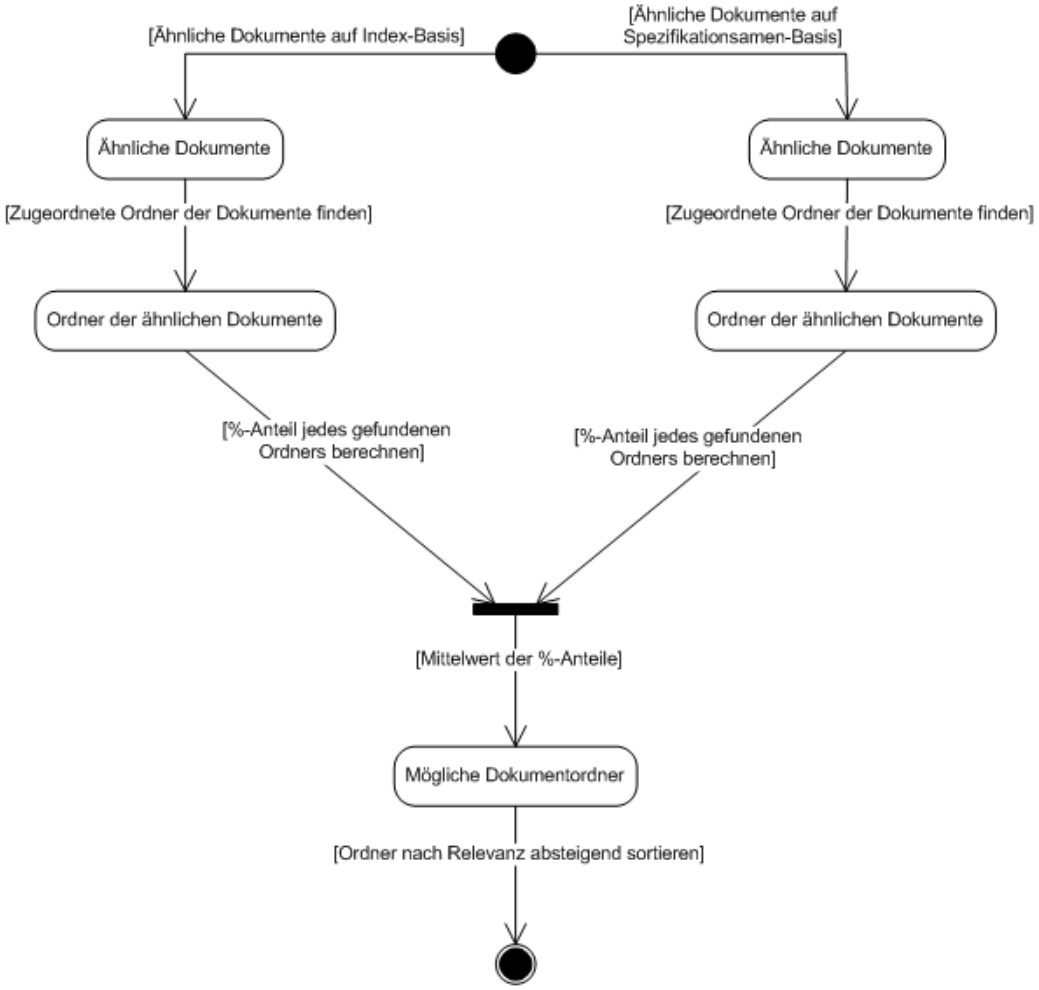


Abbildung 28: Zustandsdiagramm für Klassifikation

5.8 Abfragesprache

In diesem Abschnitt wird die Lucene Abfragesprache des JCOP-Wissensmanagementsystems behandelt, die es erlaubt, Suchanfragen für die Volltextsuche zu bilden. Die vollständige Beschreibung der Abfragesprache kann auf der Webseite http:/ / framework.zend.com/ manual/ de/ zend.search.lucene.query-language.html nachgelesen werden.

5.8.1 Ausdrücke und Phrasen

Die einfachste Suchanfrage besteht lediglich aus einem einzigen Ausdruck (siehe Abbildung 29). Suchanfragen bestehend aus mehreren einzelnen Ausdrücken werden Phrasen (siehe Abbildung 30) genannt. Phrasen erlauben dem Benutzer nach mehreren Ausdrücken auf einmal zu suchen. Um die einzelnen Ausdrücke zu verknüpfen, muss kein Operator verwendet werden, da Ausdrücke standardmäßig mit dem „OR“-Operator verknüpft werden.

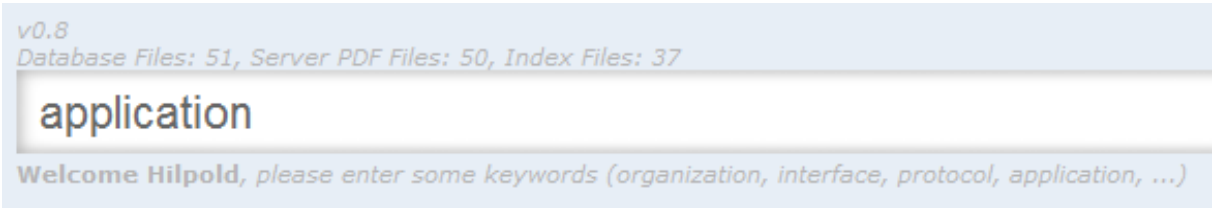


Abbildung 29: Suchanfrage mit einem einzigen Ausdruck

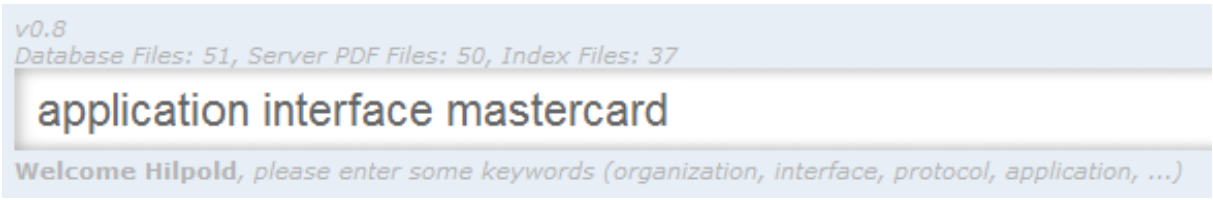


Abbildung 30: Suchanfrage mit mehreren Ausdrücken

5.8.2 Wildcards

Innerhalb einzelner Ausdrücke sind Wildcards erlaubt, jedoch nicht innerhalb von Phrasen. Der „?“-Operator (siehe Abbildung 31) steht für ein Zeichen während der „*“-Operator (siehe Abbildung 32) für mehrere Zeichen steht.

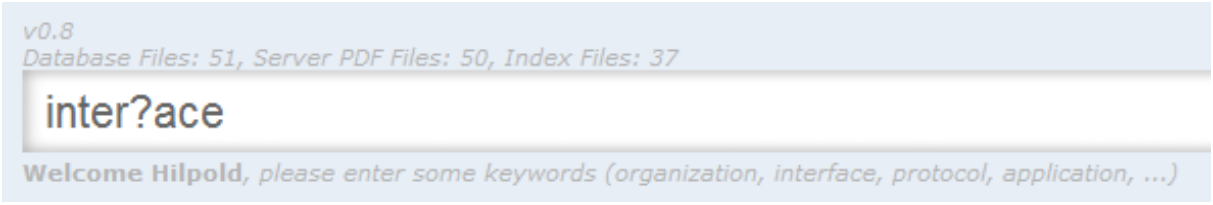


Abbildung 31: Suchanfrage mit Wildcard-Operator für ein Zeichen

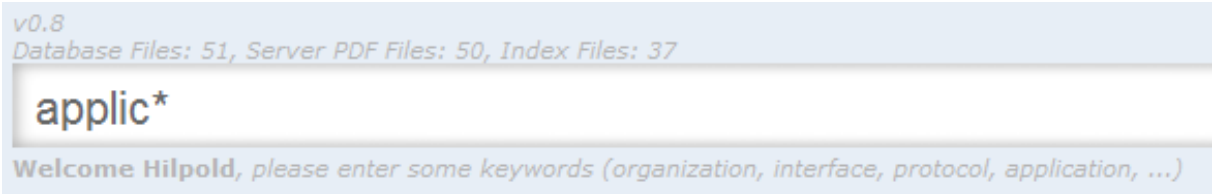


Abbildung 32: Suchanfrage mit Wildcard-Operator für mehrere Zeichen

5.8.3 Fuzzy-Suche

Basierend auf der Levenshtein-Distanz, erlaubt die Fuzzy-Suche dem Benutzer die Spezifikationen nach ähnlichen Ausdrücken zu durchsuchen. Dabei wird der "~"-Operator verwendet (siehe Abbildung 33), der nach einem Ausdruck mit dem Ähnlichkeitsfaktor angegeben wird. Falls nur der "~"-Operator verwendet wird, wird standardmäßig der Ähnlichkeitsfaktor 0,5 verwendet.

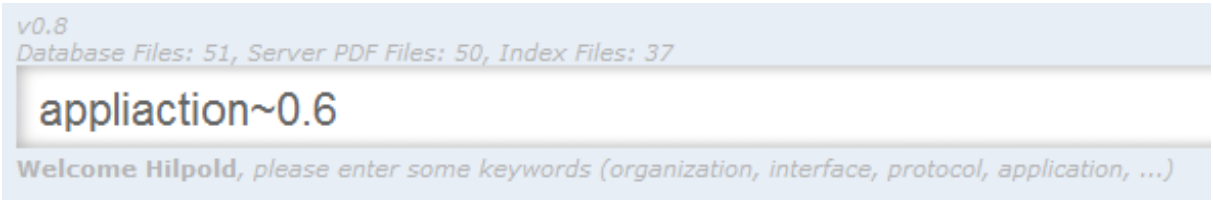


Abbildung 33: Suchanfrage mit Fuzzy-Suche und Ähnlichkeitsfaktor

5.8.4 Angenäherte Suche

Mit dem "~"-Operator (siehe Abbildung 34) können Spezifikationen nach Wörtern aus einer Phrase innerhalb eines gewissen Wortabstands durchsucht werden. Der maximale Wortabstand wird nach dem "~"-Operator angegeben.

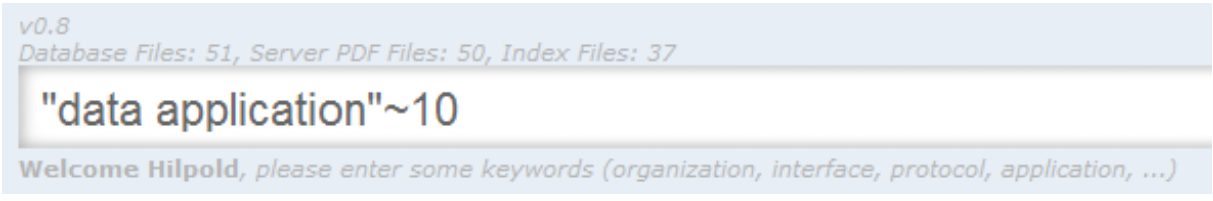


Abbildung 34: Suchanfrage mit angenäherter Suche

5.8.5 Boost-Faktor

In manchen Fällen möchte man die Relevanz eines Ausdruckes oder einer Phrase innerhalb einer Suchanfrage erhöhen. Dadurch erscheinen Spezifikationen, die

diesen Ausdruck enthalten, im Ranking höher. Der Standardwert ist 1, es können mit Werten kleiner als 1 auch Ausdrücke abgeschwächt werden. Um die Relevanz von Ausdrücken oder Phrasen zu verstärken, wird der „^“-Operator (siehe Abbildung 35 und Abbildung 36) verwendet.

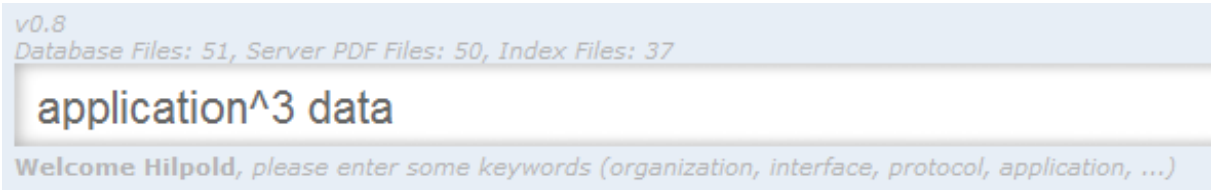


Abbildung 35: Suchanfrage mit Boost-Faktor für einen Ausdruck

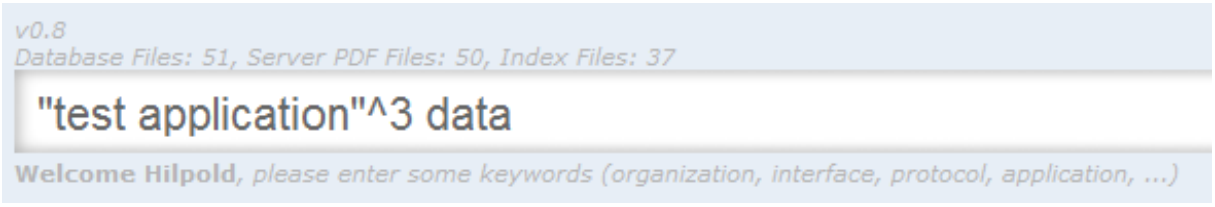


Abbildung 36: Suchanfrage mit Boost-Faktor für eine Phrase

5.8.6 Boolesche Operatoren

Mit den booleschen Operatoren „AND“, „OR“ und „NOT“ können Ausdrücke und Phrasen verknüpft werden (siehe Abbildung 37 und Abbildung 38).

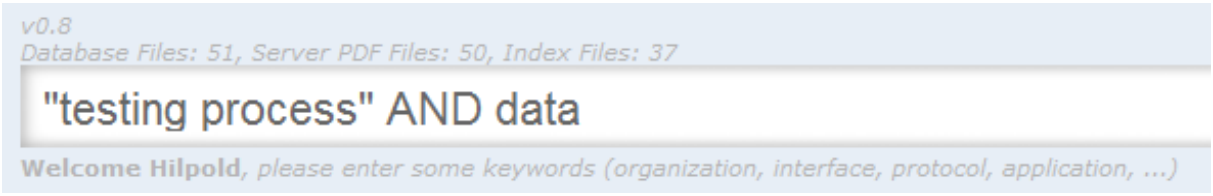


Abbildung 37: Suchanfrage mit booleschem „AND“-Operator

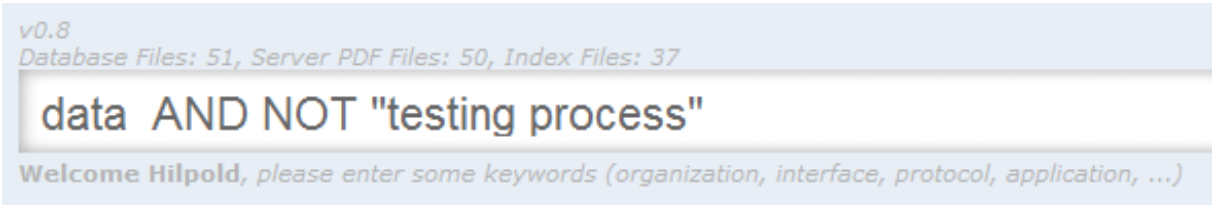


Abbildung 38: Suchanfrage mit booleschem „NOT“-Operator

5.8.7 Include/Exclude-Operatoren

Durch Benutzung des „+“-Operators (siehe Abbildung 39) vor einem Ausdruck oder einer Phrase werden alle Spezifikationen aufgelistet, die diese enthalten. Das Umgekehrte gilt für den „-“-Operator (siehe Abbildung 40).

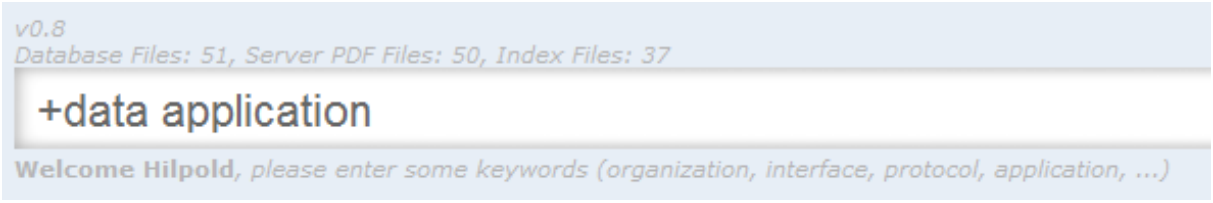


Abbildung 39: Suchanfrage mit Include-Operator

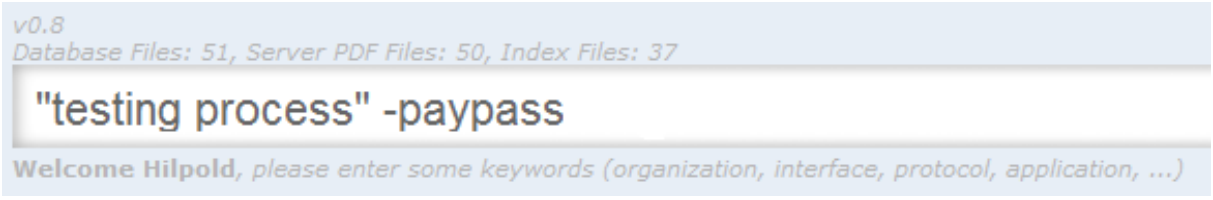


Abbildung 40: Suchanfrage mit Exclude-Operator

5.8.8 Gruppierung

Mit runden Klammern könne Ausdrücke gruppiert werden, um Sub-Suchanfragen zu formulieren (siehe Abbildung 41).

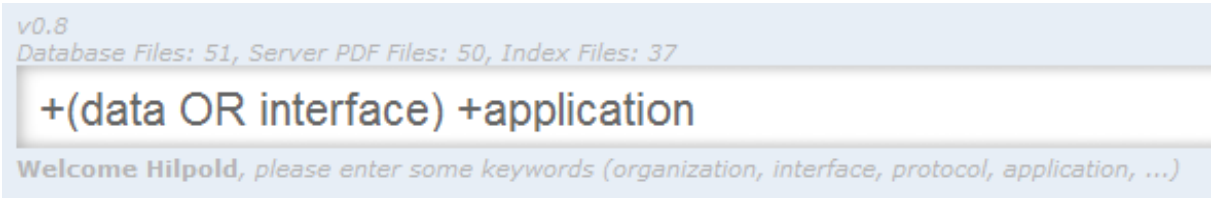


Abbildung 41: Suchanfrage mit gruppierten Ausdrücken

5.8.9 Escaping

Mit „\ “ können spezifische Zeichen der Abfragesprache maskiert werden (siehe Abbildung 42).

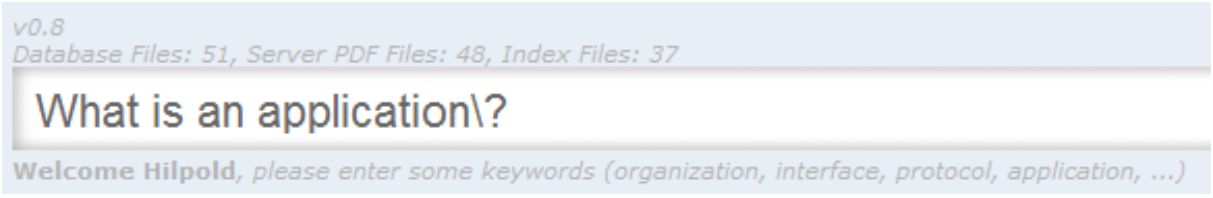


Abbildung 42: Suchanfrage mit Escaping

6 Systemevaluierung

Um die Qualität des JCOP-Wissensmanagementsystems zu bewerten, wird im Abschnitt 6.1 eine Evaluierung der funktionalen Systemanforderungen (siehe Abschnitt 1.4.2) und der Klassifikation der Spezifikationen durchgeführt. Als Referenz für die Klassifikation gilt die Benutzergruppierung der Spezifikationen in verschiedene Ordner. Die Volltextsuche bzw. die Ähnlichkeitssuche wurde nicht evaluiert, da keine Referenzdaten dafür vorhanden sind.

6.1 Anwendungsfälle

Obwohl das JCOP-Wissensmanagementsystem nur innerhalb vom Firmennetzwerk von NXP Semiconductors erreichbar ist, ist das System nur für registrierte JCOP-Mitarbeiter zugänglich (siehe Abbildung 43).

Abbildung 43: Screenshot Login

Nachdem erfolgreichen Login, wird auf jeder Seite des JCOP-Wissensmanagementsystems am oberen Bildrand eine Navigationsleiste (siehe Abbildung 44) eingebunden. Diese Navigationsleiste beinhaltet in erster Linie das Eingabefeld für die Volltextsuche und die Links zu allen Bereichen des Systems (siehe Tabelle 6). Über dem Eingabefeld für die Volltextsuche wird dargestellt wie viele Spezifikationen sich gerade in der Datenbank befinden, wie viele davon eine Spezifikationsdatei besitzen und wie viele davon im System indiziert sind. Ein Link zum Index optimieren wird ebenfalls oberhalb des Eingabefeldes in der Navigationsleiste angeboten. Am unteren Rand der Navigationsleiste werden, die zuletzt geänderten Spezifikationen angezeigt.

Abbildung 44: Screenshot Navigationsleiste

Symbol	Bereich
	Virtuelle Ordnerstruktur
	Graph mit Spezifikationsbeziehungen
	Verwaltung der Spezifikationen
	Verwaltung der virtuellen Ordner
	Verwaltung der Spezifikationstypen
	Verwaltung der Spezifikationsbeziehungen
	Verwaltung der Benutzer
	Benutzer-Hilfe
	Ausloggen

Tabelle 6: Symbole der Navigationsleiste

Sobald man erfolgreich eingeloggt ist, gelangt man auf Seite der virtuellen Ordnerstruktur (siehe Abbildung 45). Falls eine Spezifikation eine PDF-Datei besitzt, wird vor dem Name der Spezifikation ein PDF-Symbol dargestellt. Bei Klick auf dieses Symbol öffnet sich die Spezifikationsdatei in einem neuen Fenster. Falls keine PDF-Datei ins System hochgeladen wurde, wird ein Standard-Symbol dargestellt. Klick der Benutzer auf dieses Standard-Symbol, wird er zum Änderungsformular für die jeweilige Spezifikation weitergeleitet. Zum Graphen mit den jeweiligen Spezifikationsbeziehungen (siehe Abbildung 46) gelangt man mit einem Klick auf den Spezifikationsnamen in der virtuellen Ordnerstruktur.

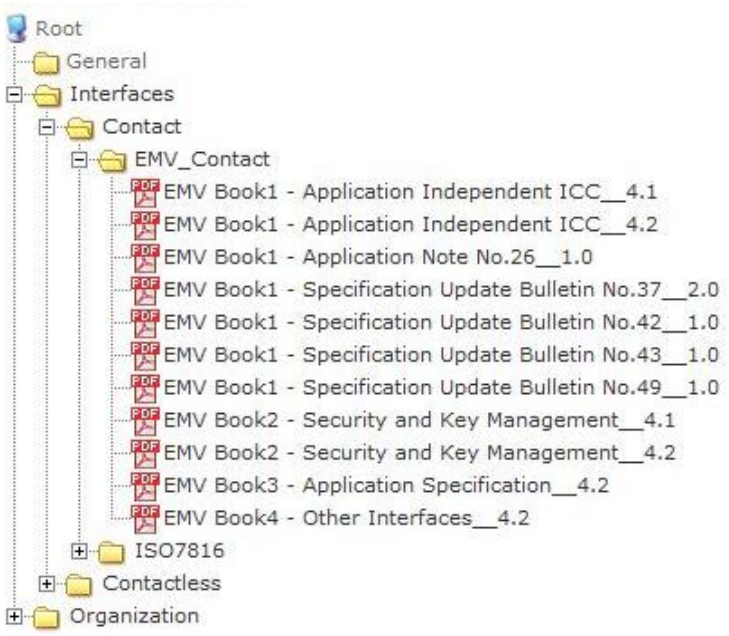


Abbildung 45: Screenshot virtuelle Ordnerstruktur

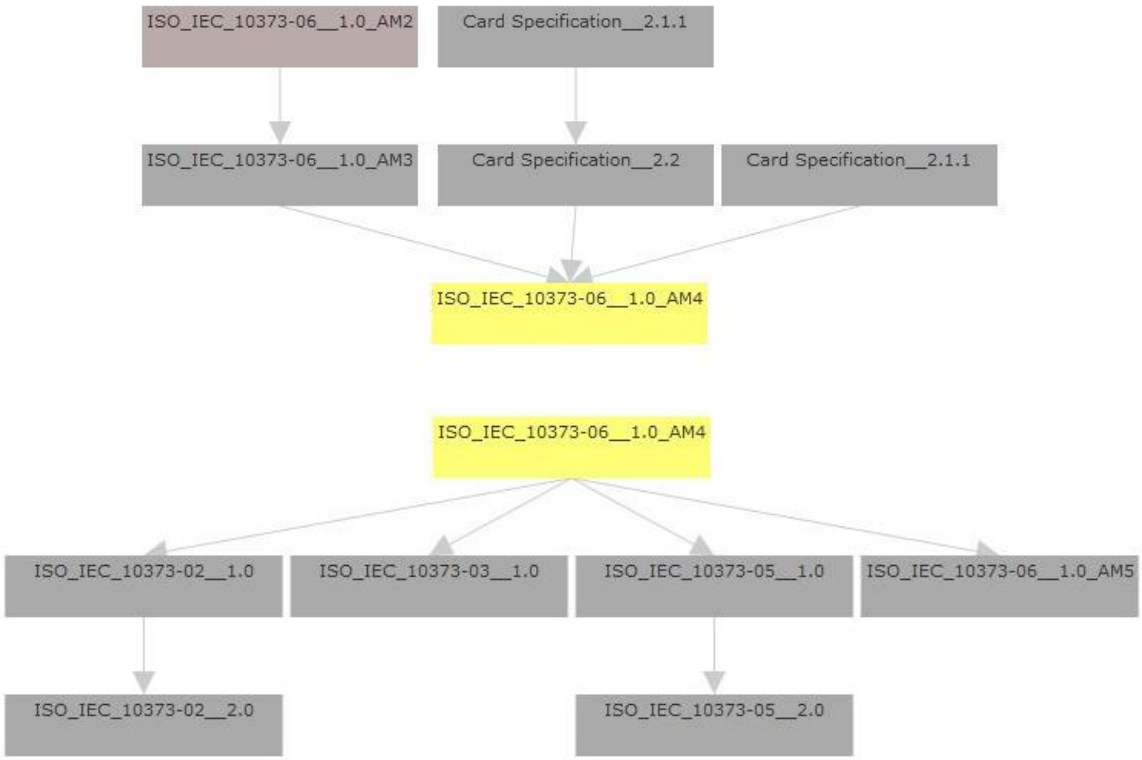


Abbildung 46: Screenshot Graph der Spezifikationsbeziehungen

Der gerichtete Graph der Spezifikationsbeziehungen stellt übergeordnete und untergeordnete Spezifikationen zu genau einer Spezifikation und einem Spezifikationsbeziehungstyp dar. Es werden nur zwei Ebenen an übergeordneten

bzw. untergeordneten Spezifikationen angezeigt. Je dunkler sich die Boxen der Spezifikationen färben, desto mehr weitere Spezifikationen stecken im Verborgenen. Bei Klick auf eine der Boxen, wandert diese in die Mitte des Graphen und falls verborgenen Spezifikationen vorhanden sind werden diese sichtbar.

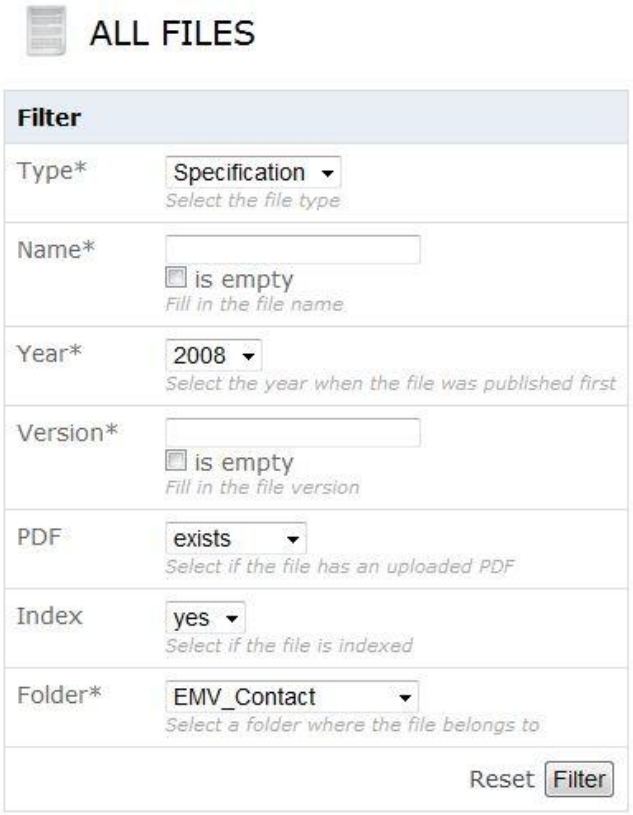


Abbildung 47: Screenshot Filter im Verwaltungsbereich der Spezifikationen

In jedem Verwaltungsbereich des Systems (siehe Tabelle 6) gibt es einen Datenbank-Filter (siehe Abbildung 47), mit dem die jeweilige Auflistung (siehe Abbildung 48) des Verwaltungsbereiches gefiltert werden kann. Für jeden Eintrag dieser Auflistung stehen in den unterschiedlichen Verwaltungsbereichen unterschiedliche Aktion zu Verfügung (siehe Tabelle 7). Dabei sind die Aktionen einen Eintrag dieser Auflistungen zu erstellen (Formular wird geöffnet), zu editieren (Formular wird geöffnet) und zu löschen in jedem Verwaltungsbereich vorhanden.

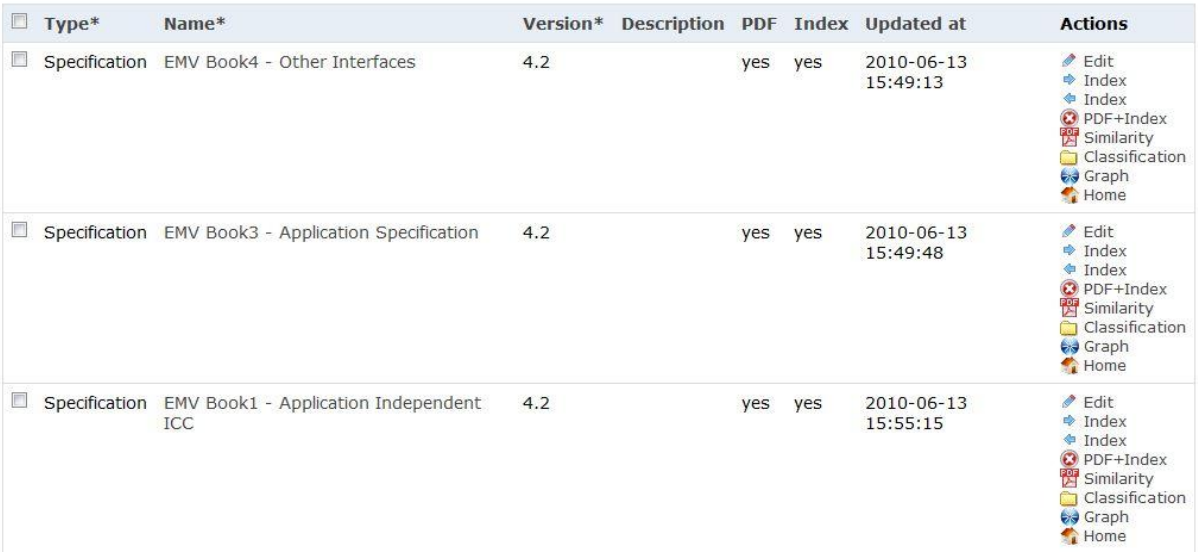


Abbildung 48: Screenshot Auflistung im Verwaltungsbereich der Spezifikationen

Symbol	Beschreibung
	Spezifikation erstellen
	Spezifikation löschen
	Spezifikation editieren
	Spezifikation zum Index hinzufügen
	Spezifikation aus Index entfernen
	Spezifikationsdatei mit Indexeintrag löschen
	Ähnliche Spezifikationen suchen
	Spezifikation klassifizieren
	Graph mit Spezifikationsbeziehungen für Spezifikation öffnen
	Virtuelle Ordnerstruktur für Spezifikation aufklappen

Tabelle 7: Aktionen im Verwaltungsbereich der Spezifikationen

In Abbildung 49 wird eines der zentralen Formulare im System dargestellt. Es handelt sich um das Formular aus dem Verwaltungsbereich der Spezifikationen um

eine Spezifikation zu Ordnern zu zuweisen bzw. mit anderen Spezifikationen zu verknüpfen.

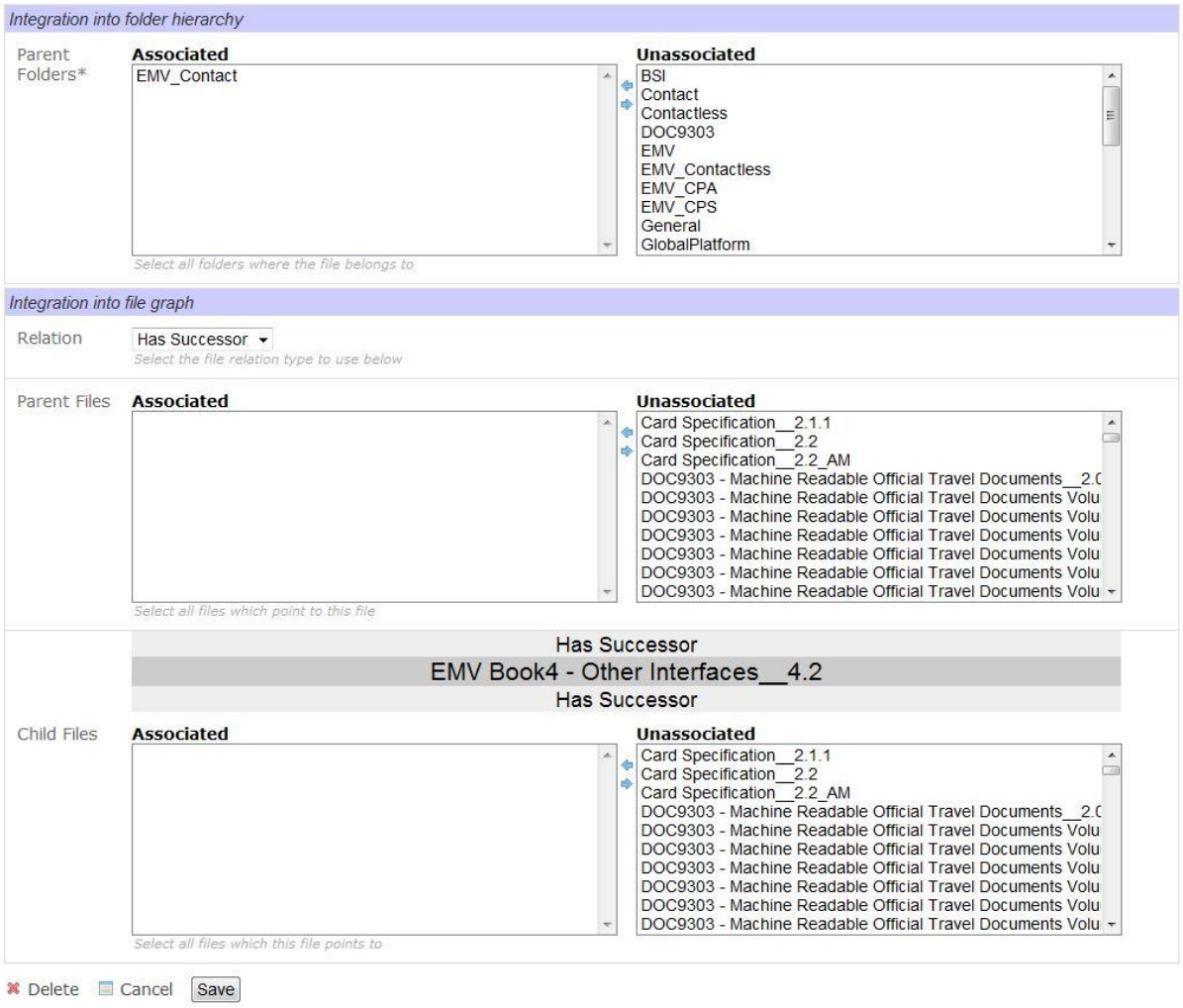


Abbildung 49: Screenshot Spezifikationsbeziehungen editieren

In Abbildung 50 und in Abbildung 51 werden die Ergebnisseiten einer Volltextsuche bzw. einer Klassifikation dargestellt. Die Ergebnisseite der Such von ähnlichen Dokumenten hat dieselbe Struktur wie in Abbildung 50.

FILE SEARCH RESULT

Type	Name	Version	Score
Specification	Java Card 3.0.1 Application Programming Interface	1.0_Classic	0.20818146073129
Specification	Java Card 2.2.2 Application Programming Interface	1.0	0.20607236332202
Specification	Java Card 2.1.1 Application Programming Interface	1.0	0.20607236332202
Specification	TR03105 Part-4	2.01.1	0.0016280208099836
Specification	Java Card 3.0.1 Runtime Environment	1.0_Classic	0.0015546912091276
Specification	Java Card 2.2.2 Runtime Environment	1.0	0.0013159135540465
Specification	Java Card 2.1.1 Runtime Environment	1.0	0.0012134550666359
Specification	Java Card 2.1.1 Relase Notes	1.0	0.0010853472066557
Specification	Java Card 2.2.2 Virtual Machine	1.0	0.00075536968731435
Specification	Java Card 2.1.1 Virtual Machine Specification	1.0	0.00072807303998151

Abbildung 50: Screenshot Ergebnis Volltextsuche

FILE CLASSIFICATION RESULT

Current Folder	File
EMV_Contact	EMV Book4 - Other Interfaces 4.2

Suggested Folder	Percentage
EMV_Contact	50 %
JavaCard	50 %

Abbildung 51: Screenshot Ergebnis Klassifikation

6.2 Klassifikation

In folgenden Abschnitten werden der für die Systemevaluierung verwendete Datensatz (siehe Abschnitt 6.2.1) sowie Kennwerte und Parameter (siehe Abschnitt 6.2.2) beschrieben. Eine Auswertung (siehe Abschnitt 6.2.3) der Ergebnisse erfolgt am Ende dieses Abschnittes.

6.2.1 Datensatz

Für die Evaluierung standen 118 Spezifikationen zur Verfügung (siehe Abschnitt 1.5). Davon können 105 indiziert werden. Die restlichen Spezifikationen sind entweder passwortgeschützt oder enthalten ausschließlich Bilder. In dieser Arbeit wurde das automatische Extrahieren von Text aus Spezifikationen ausschließlich mit Bildern nicht berücksichtigt. Jede Spezifikation gehört zu mindestens einem Ordner. Diese Zuteilung wurde in Zusammenarbeit mit NXP erarbeitet. Durch die Anzahl der indizierten Spezifikationen und der 24 möglichen Ordner wird während der Klassifikation 2520 Mal entschieden, ob eine Spezifikation zu einem Ordner gehört oder nicht. Im Index sind nach der Indizierung 13.594 Indexterme enthalten.

6.2.2 Kennwerte und Parameter

Für die Klassifikation von Spezifikation wird der K-Nearest-Neighbour-Algorithmus (siehe Abschnitt 5.7.5) verwendet. Es gibt keinen empfohlenen Wert für K. Der Parameter K hängt stark von dem verwendeten Datensatz ab und muss empirisch ermittelt werden. Im Laufe der Evaluierung werden für K die Werte 3, 7 und 11 verwendet.

Um für die Klassifikation die Ähnlichkeit von Dokumenten zu berechnen, muss der sogenannte Term-Vektor der zu vergleichenden Spezifikation erstellt werden. Die Länge des Term-Vektors ist ein wesentlicher Parameter für die Klassifikation. Er beeinflusst stark die Geschwindigkeit der Ähnlichkeitsberechnung. Aus diesem Grund muss die Länge des Term-Vektors limitiert werden. Auch hierfür gibt es keine empfohlene Länge. Die Länge des Term-Vektors hängt ebenfalls stark von dem verwendeten Datensatz ab und muss ebenfalls empirisch ermittelt werden. Während der Evaluierung wird die Länge des Term-Vektors von 10 bis 50 in 10er Schritten variiert. Um die Qualität der Klassifikation zu bewerten, werden folgende klassische Kennwerte aus dem Bereich Information Retrieval verwendet:

- Mikromittelwert Accuracy (4.3)
- Mikromittelwert Recall (4.1)
- Mikromittelwert Fallout (4.4)

Für jede Kombination einer indizierten Spezifikation und eines möglichen Ordners wird eine binäre Entscheidung getroffen, nämlich ob eine Spezifikation in einen Ordner gehört oder nicht. Da jede indizierte Spezifikation nur zum höchst wahrscheinlichsten (also genau einen) Ordner klassifiziert wird, erhält man in der Evaluierung immer gleich viele False-Negative- und False-Positive-Klassifikationen. Daher ist der Recall identisch mit der Precision und das F1-Maß wird somit hinfällig.

6.2.3 Auswertung

In diesen Abschnitten werden die Ergebnisse der Evaluierung graphisch als Diagramme dargestellt. Dabei wird auf der y-Achse immer der entsprechende Kennwert und auf der x-Achse die Länge des Term-Vektors aufgetragen. Für die unterschiedlichen Parametereinstellungen des K-Nearest-Neighbour-Algorithmus (K=3, 7, 11) wird jeweils eine eigene Kurve in den Diagrammen dargestellt. Im Laufe der Implementierung wurde die Struktur des Index verändert. Jene Ergebnisse, die sich auf die kleine Indexstruktur (siehe Tabelle 4) beziehen, sind mit dem zusätzlichen Namen „Klein“ versehen. Die Ergebnisse mit dem zusätzlichen Namen „Groß“ beziehen sich auf die große Indexstruktur (siehe Tabelle 5). Um den Recall der Klassifikation mit der kleine bzw. großen Indexstruktur (siehe Abbildung 53) zu verbessern, wurde mit der großen Indexstruktur zusätzlich die Ähnlichkeit der Spezifikationsnamen (siehe Abschnitt 5.7.5) berücksichtig. Diese Ergebnisse sind in den Diagrammen mit „Name“ gekennzeichnet.

Die Klassifikation basierend auf die kleine bzw. die große Indexstruktur liefert für die Accuracy (siehe Abschnitt 4.3) und den Fallout (siehe Abschnitt 4.4) sehr gute Werte (siehe Abbildung 52 und Abbildung 54). Die Wahl des Parameter K für den K-Nearest-Neighbour-Algorithmus bzw. die Veränderung der Indexstruktur hat nur einen geringen Einfluss auf die Ergebnisse. Der Grund für diese sehr guten Accuracy- und Fallout-Werte liegt darin, dass der K-Nearest-Neighbour-Algorithmus Spezifikationen jeweils zu genau einen Ordner zuweist. Daraus ergibt sich ständig ein sehr hoher True-Negative-Wert. Die Ergebnisse für den Recall (siehe Abschnitt 4.1) fallen im Vergleich zur Accuracy sehr mager aus und sind in der Praxis nicht brauchbar (siehe Abbildung 53).

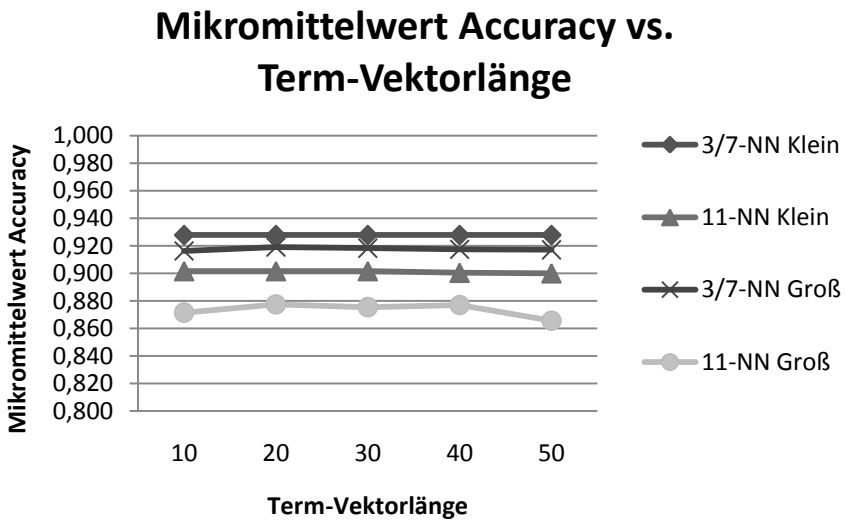


Abbildung 52: Mikromittelwert Accuracy vs Term-Vektorlänge (Klein/Groß)

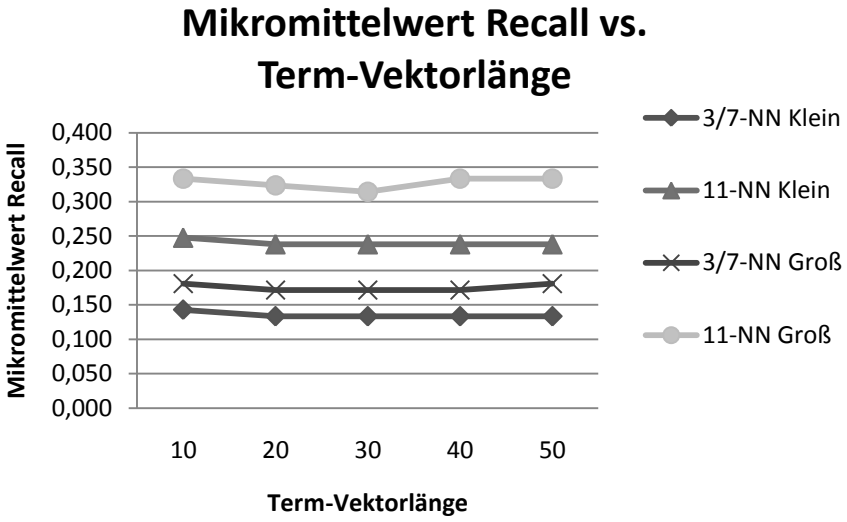


Abbildung 53: Mikromittelwert Recall vs. Term-Vektorlänge (Klein/Groß)

Durch Kombination der großen Indexstruktur (siehe Abschnitt 5.5) mit der Namensähnlichkeiten von Spezifikationen (siehe Abschnitt 5.7.5) konnte der Recall der Klassifikation deutlich gesteigert werden (siehe Abbildung 56). Dadurch wurden auch die Werte für den Fallout und die Accuracy nochmals gesteigert (siehe Abbildung 57 und Abbildung 55). Der Grund für die Steigerung durch Namensähnlichkeit liegt darin, dass sich die Namen der Spezifikationen für ein JCOP-Thema (Spezifikationen für ein Thema sind meist zusammen in einen Ordner gruppiert) oft sehr ähnlich sind, und daher relative einfach klassifiziert werden können (siehe Abschnitt 5.7.5)

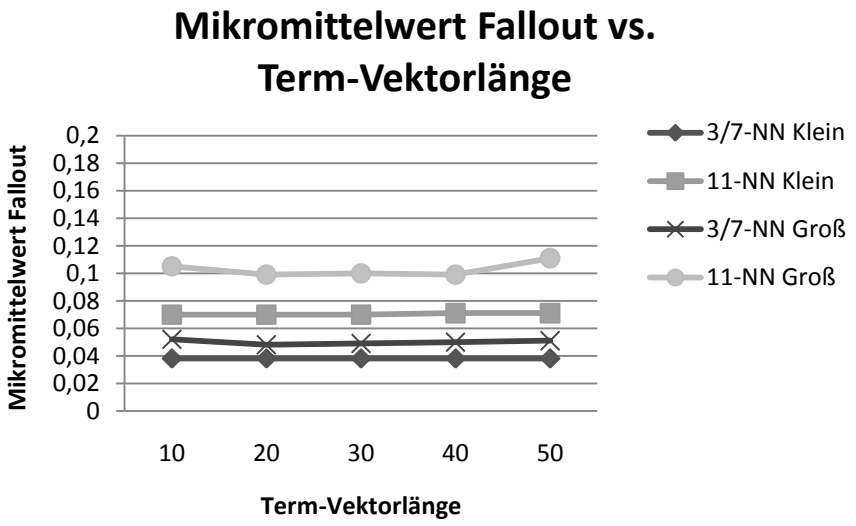


Abbildung 54: Mikromittelwert Fallout vs. Term-Vektorlänge (Klein/Groß)

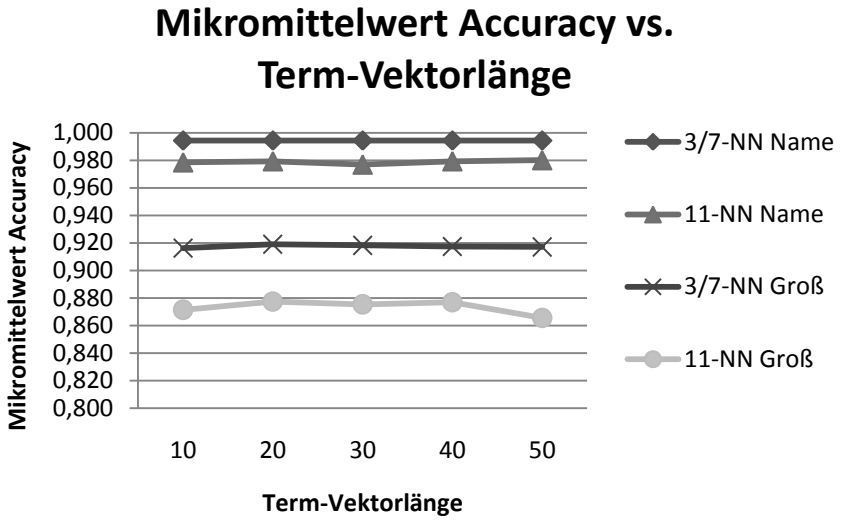


Abbildung 55: Mikromittelwert Accuracy vs. Term-Vektorlänge (Name/Groß)

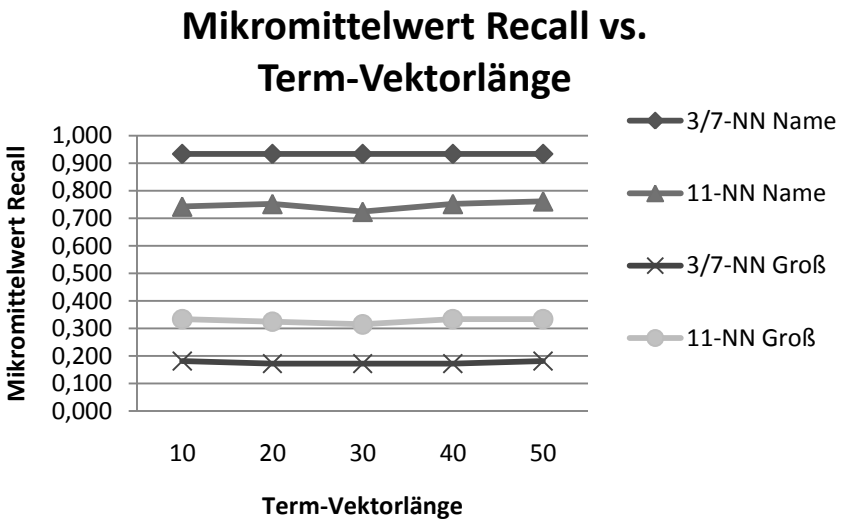


Abbildung 56: Mikromittelwert Recall vs. Term-Vektorlänge (Name/Groß)

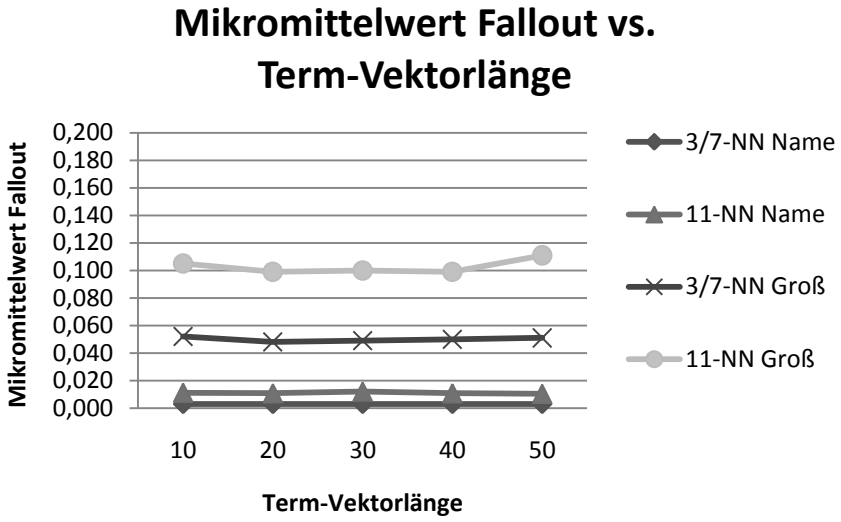


Abbildung 57: Mikromittelwert Fallout vs. Term-Vektorlänge (Name/Groß)

7 Zusammenfassung

Die Ergebnisse der Klassifikation mit der kleinen bzw. der großen Indexstruktur, waren nicht zufriedenstellend. Damit die Klassifikation in der Praxis verwendet werden kann, wurde daher die Namensähnlichkeit der Spezifikationen mit berücksichtigt. Die Qualität der Klassifikation ohne Namensähnlichkeit kann trotzdem weiter verbessert werden.

Der erste Schritt dies zu erreichen, ist das System mit mehr Spezifikationen zu füllen und diese je nach Inhalt bzw. Konzept möglichst passend in Ordnern zu strukturieren. Durch das Vorhandensein einer größeren Datenmenge wird eine bessere Ausgangslage für die Optimierung der Klassifikation geschaffen. Dabei ist von Vorteil, dass der K-Nearest-Neighbour-Algorithmus zu den Lazy-Learning-Verfahren gehört und anstelle der Benutzung von begrenzten Trainingsdaten, den Arbeitspunkt für jede neue Klassifikation an alle vorhandenen Daten anpasst.

Die Anzahl der betrachteten Nachbarn ist beim K-Nearest-Neighbour-Algorithmus für die Qualität der Klassifikation ausschlaggebend. Für zu kleine Werte kann das Rauschen in den Daten das Ergebnis verschlechtern und für zu große Werte besteht die Gefahr zu viele unähnliche Dokumente in den Entscheidungsprozess mit einzubeziehen. Abhilfe dafür kann eine gewichtete Abstandfunktion schaffen, die ähnlichere Dokumente höher gewichtet. In der Evaluierung hatte der Parameter K (K=3, 7, 11) nur sehr geringe Auswirkungen.

Ein weiterer Optimierungsschritt besteht darin, die Dokumentinhalte durch Filterung vor der Indizierung zu verfeinern. Dies kann mittels eines sogenannten Hauptwortfilters erreicht werden. Dabei werden alle Terme einer Spezifikation, die nicht Hauptwörter sind, entfernt. Hauptwörter sind als Indexterme besonders geeignet, da sie auch ohne Kontext selbsterklärend sind. Verben, Adjektive und andere Wortarten sind ohne Kontext weniger aussagekräftig und können daher vernachlässigt werden. Eine Stoppwortliste wird dadurch nahezu überflüssig, kann jedoch trotzdem verwendet und angepasst werden, um wenig aussagekräftige Hauptwörter zu filtern. Durch diese Optimierung wird zudem Speicherplatz gespart, da der Index kleiner wird, und gleichzeitig die Qualität der Suche gesteigert.

Das JCOP-Wissensmanagementsystem verwendet als Gewichtungsfunktion für die Suche die klassische Methode mit Term Frequency und Inverse Term Frequency und als Ähnlichkeitsfunktion das Cosinus-Maß. Durch weiteres Anpassen und Variieren dieser Funktion an die bestehenden Daten können bessere Ergebnisse erzielt werden.

Ein weiterer wichtiger Parameter für die Klassifikation von Spezifikationen ist der Term-Vektor. Um Spezifikationen zu vergleichen, werden nicht alle Indexterme einer Spezifikation verwendet, sondern nur eine gewisse Anzahl. Diese ausgewählten Indexterme bilden zusammen den Term-Vektor. Unterschiedliche Längen des Term-Vektors führen zu unterschiedlichen Ergebnissen. Wird die Länge zu kurz gewählt, werden vielleicht wichtige Indexterme einer Spezifikation in der Klassifikation nicht berücksichtig. Bei einer zu großen Länge beeinflussen hingegen unwichtige Indexterme den Suchprozess. Es gilt hier die geeignete Länge des Term-Vektors für den zugrundeliegenden Datensatz zu finden. Da auch die unterschiedlichen Term-Vektorlängen nur geringe Auswirkungen auf die Ergebnisse hatten ist die Term-Vektorlänge 10 für das JCOP-Wissensmanagementsystem, um die Klassifikation zu beschleunigen.

Die Volltextsuche liefert jene Spezifikationen zurück, die am besten zur Suchanfrage des Benutzers passen. Falls darin Spezifikationen mit weit über 100 Seiten vorkommen, kann das Suchen der gewünschten Information sehr mühsam sein. Um dem Benutzer exaktere Ergebnisse für eine Suchanfrage zu liefern, können Spezifikationen für die Indizierung in Teildokumente zerlegt werden. Ein solches Teildokument kann mehrere Seiten lang sein, je nachdem, wie genau die Suchergebnisse gewünscht werden. Die Volltextsuche liefert dann nicht mehr die originalen Spezifikationen zurück, sondern eine Liste von Teildokumenten, die am besten mit der Suchanfrage des Benutzers übereinstimmt. Natürlich sollte die Verbindung zwischen der ursprünglichen Spezifikation und deren Teildokumenten erhalten bleiben.

Bei der Volltextsuche kann man durch Relevance Feedback die jeweilige Suchanfrage automatisch verfeinern, um die Ergebnismenge zu optimieren. Als Beispiel kann der Benutzer die Einträge in der Ergebnismenge als relevant oder nicht relevant markieren. Daraufhin wird die Suchanfrage, ohne dass der Benutzer etwas davon bemerkt, angepasst. Das geschieht entweder durch Veränderung der bestehenden Term-Gewichte der Suchanfrage (Term Reweighting) oder durch Erweiterung der Suchanfrage mit neuen Termen (Query Expansion).

Die Optimierung eines solchen Systems ist im Allgemeinen ein mühsamer Prozess. Grundvoraussetzung für eine Optimierung dieser Art ist immer eine große Datenmenge als Basis.

A Stoppwortliste

Die folgende Stoppwortliste wurde im entwickelten JCOP-Wissensmanagementsystem verwendet:

a, about, above, across, after, afterwards, again, against, all, almost, alone, along, already, also, although, always, am, among, amongst, amoungst, amount, an, and, another, any, anyhow, anyone, anything, anyway, anywhere, are, around, as, at, back, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, bill, both, bottom, but, by, call, can, cannot, cant, co, computer, con, could, couldnt, cry, de, describe, detail, do, done, down, due, during, each, eg, eight, either, eleven, else, elsewhere, empty, enough, etc, even, ever, every, everyone, everything, everywhere, except, few, fifteen, fify, fill, find, fire, first, five, for, former, formerly, forty, found, four, from, front, full, further, get, give, gohad, has, hasnt, have, he, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, him, himself, his, how, however, hundred, i, ie, if, in, inc, indeed, interest, into, is, it, ist, itself, keep, last, latter, latterly, least, less, ltd, made, many, may, me, meanwhile, might, mill, mine, more, moreover, most, mostly, move, much, must, my, myself, name, namely, neither, never, nevertheless, next, nine, no, nobody, none, noone, nor, not, nothing, now, nowhere, of, off, often, on, once, one, only, onto, or, other, others, otherwise, our, ours, ourselves, out, over, own, part, per, perhaps, please, put, rather, re, same, see, seem, seemed, seeming, seems, serious, several, she, should, show, side, since, sincere, six, sixty, so, some, somehow, someone, something, sometime, sometimes, somewhere, still, such, system, take, ten, than, that, the, their, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, thereupon, these, they, thick, thin, third, this, those, though, three, through, throughout, thru, thus, to, together, too, top, toward, towards, twelve, twenty, two, un, under, until, up, upon, us, very, via, was, we, well, were, what, whatever, when, whence, , whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, with, within, without, would, yet, you, your, yours, yourself, yourselves

Im Laufe der Systementwicklung wurden folgende Stoppwörter hinzugefügt:

version, january, february, march, april, may, june, july, august, september, october , november, december, figure, table

B Ergebnisse

3-NN	7-NN	11-NN	Term-Vektorlänge	Kennwert
0,928	0,927	0,902	10	Micro Average Accuracy
0,928	0,928	0,902	20	Micro Average Accuracy
0,928	0,928	0,902	30	Micro Average Accuracy
0,928	0,928	0,900	40	Micro Average Accuracy
0,928	0,928	0,900	50	Micro Average Accuracy
0,072	0,073	0,098	10	Micro Average Error
0,072	0,072	0,098	20	Micro Average Error
0,072	0,072	0,098	30	Micro Average Error
0,072	0,072	0,100	40	Micro Average Error
0,072	0,072	0,100	50	Micro Average Error
0,143	0,143	0,248	10	Micro Average Recall/ Precision/ F1
0,133	0,133	0,238	20	Micro Average Recall/ Precision/ F1
0,133	0,133	0,238	30	Micro Average Recall/ Precision/ F1
0,133	0,133	0,238	40	Micro Average Recall/ Precision/ F1
0,133	0,133	0,238	50	Micro Average Recall/ Precision/ F1
0,038	0,039	0,07	10	Micro Average Fallout
0,038	0,038	0,07	20	Micro Average Fallout
0,038	0,038	0,07	30	Micro Average Fallout
0,038	0,038	0,071	40	Micro Average Fallout
0,038	0,038	0,071	50	Micro Average Fallout

Tabelle 8: Ergebnisse mit kleiner Indexstruktur

3-NN	7-NN	11-NN	Term-Vektorlänge	Kennwert
0,916	0,916	0,871	10	Micro Average Accuracy
0,919	0,918	0,877	20	Micro Average Accuracy
0,918	0,917	0,875	30	Micro Average Accuracy
0,917	0,918	0,877	40	Micro Average Accuracy
0,917	0,916	0,865	50	Micro Average Accuracy
0,084	0,084	0,129	10	Micro Average Error
0,081	0,082	0,123	20	Micro Average Error
0,082	0,083	0,125	30	Micro Average Error
0,083	0,082	0,123	40	Micro Average Error
0,083	0,084	0,135	50	Micro Average Error
0,181	0,190	0,333	10	Micro Average Recall/ Precision/ F1
0,171	0,171	0,324	20	Micro Average Recall/ Precision/ F1
0,171	0,171	0,314	30	Micro Average Recall/ Precision/ F1
0,171	0,171	0,333	40	Micro Average Recall/ Precision/ F1
0,181	0,181	0,333	50	Micro Average Recall/ Precision/ F1
0,052	0,053	0,105	10	Micro Average Fallout
0,048	0,043	0,099	20	Micro Average Fallout
0,049	0,050	0,100	30	Micro Average Fallout
0,050	0,050	0,099	40	Micro Average Fallout
0,051	0,052	0,111	50	Micro Average Fallout

Tabelle 9: Ergebnisse mit großer Indexstruktur

3-NN	7-NN	11-NN	Term-Vektorlänge	Kennwert
0,994	0,994	0,979	10	Micro Average Accuracy
0,994	0,994	0,979	20	Micro Average Accuracy
0,994	0,994	0,977	30	Micro Average Accuracy
0,994	0,994	0,979	40	Micro Average Accuracy
0,994	0,994	0,980	50	Micro Average Accuracy
0,006	0,006	0,021	10	Micro Average Error
0,006	0,006	0,021	20	Micro Average Error
0,006	0,006	0,023	30	Micro Average Error
0,006	0,006	0,021	40	Micro Average Error
0,006	0,006	0,020	50	Micro Average Error
0,933	0,933	0,743	10	Micro Average Recall/ Precision/ F1
0,933	0,933	0,752	20	Micro Average Recall/ Precision/ F1
0,933	0,933	0,724	30	Micro Average Recall/ Precision/ F1
0,933	0,933	0,752	40	Micro Average Recall/ Precision/ F1
0,933	0,933	0,762	50	Micro Average Recall/ Precision/ F1
0,003	0,003	0,011	10	Micro Average Fallout
0,003	0,003	0,011	20	Micro Average Fallout
0,003	0,003	0,012	30	Micro Average Fallout
0,003	0,003	0,011	40	Micro Average Fallout
0,003	0,003	0,010	50	Micro Average Fallout

Tabelle 10: Ergebnisse mit großer Indexstruktur und Namensähnlichkeit

Literaturverzeichnis

[1] Dirk Lewandowski. Web Information Retrieval. Technologien zur Informationssuche im Internet. DGI, 2005, ISBN: 3-925474-55-2.

[2] Ricardo Baeza-Yates und Berthier Ribeiro-Neto. Modern Information Retrieval. Addison Wesley, 2008, ISBN: 0-201-39829-X.

[3] Christopher D. Manning, Prabhakar Raghavan und Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008, ISBN: 978-0-521-86571-5.

[4] Reginald Ferber. Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web. Dpunkt, 2003, ISBN: 3-89864-213-5

[5] V. Gudivada, V. Raghavan, W. Grosky und R. Kasanagottu. Information retrieval on the World Wide Web. IEEE Internet Computing, Sep-Oct, 1997.

[6] Gerald Kowalski. Information Retrieval Systems. Theory and Implementation. Kluwer Academic Publishers, 1997, ISBN: 0-7923-9926-9

[7] H. Chu: Information Representation and Retrieval in the Digital Age. Medford, NJ: Information Today, 2003.

[8] Belkin, N. J.; Croft, W. B. (1987): Retrieval Techniques. Annual Review of Information Science and Technology 22, 109-145

[9] G. Salton. The SMART Retrieval-System – Experiments in Automatic Document Processing. Prentice Hall Inc., Englewood Cliffs, NJ, 1971

[10] William P. Jones und George W. Furnas. Pictures of Relevance: A Geometric Analysis of Similarity Measures. Journal oft he American Society for Information Science, 38(6) Seite 420-442, 1987.

[11] Michael Granitzer. Hierarchical Text Classification using Methods from Machine Learning. Master's Thesis, Graz University of Technology, 2003.

[12] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, No. 1, March 2002.

[13] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. Journal oft the ACM, 15(1):8-36, January 1968.

[14] G. Salton and M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill Book Co., New York, 1963.

[15] G. Salton und C. Buckley. Term-weighting approaches in automatic retrieval. Information Processing Management, 24(5):513-523, 1988.

[16] S. E. Robertson und K. Sparck Jones. Relevance weighting of search terms. Journal oft he American Society for Information Sciences, 27(3):129-146, 1976.

[17] M.D. Lennon, D. Pierce, B. Tarry und P. Willett. An Evaluation of Some Conflation Algorithms for Information Retrieval. Journal of Information Science 3, 177-83, 1981.

[18] D. Harman. How Effective is Suffixing? Journal oft he American Society for Information Science, 42(1), 7-15, 1991.

[19] C. J.van Rijsbergen. Information Retreival. Butterworths, 1979.

[20] Andreas Lessiak. Performance Optimization of the JCOP Java Card Operating System based on HW/ SW Co-Design. Master's Thesis, Graz University of Technology, 2008.

[21] C. Fox. Lexical analysis and stoplists. In W. Frankes and R. Baeza-Yates, editors, Information Retrieval: Data Structures & Algorithms, pages 102-130. Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[22] Peter Roget. Roget's II The New Thesaurus. Houghton Mifflin Company, Boston, USA, 1988.

[23] D. J. Foskett. Thesaurus. In K. Sparck Jones and P. Willet, editors, Readings in Information Retrieval, pages 111-134. Morgan Kaufmann Publishers, Inc., 1997.

[24] C. J. van Rijsbergen, S. E. Robertson und M. F. Porter. New models in probabilistic information retrieval. London: British Library, 1980.

[25] Rob Hooper. The Design and Optimisation of Stemming Algorithms. 2005.

[26] NXP Semiconductors. Online im Internet: http:/ / www.nxp.com [Stand: 12.09.2010].

[27] D.C. Blair. Language and Representation in Information Retrieval. Elsevier, 1990, ISBN: 0-444-88437-8

[28] G. Salton. Automatic Text Processing. The Transformation, Analysis and Retrieval of Information by Computer. Addison Wesley, 1989, ISBN: 0-201-12227-8.

[29] F. Burkowski. An Algebra for Hierarchically Organized Text-Dominated Databases. Information Processing and Management, 28(3):333-348, 1992.

[30] R. Baeza-Yates and G. Navarro. Integrating Contents and Structure in Text Retrieval. ACM SIGMOD Record, 25(1):67-79, 1996.

[31] Christian Gütl. Information Search and Retrieval. Technische Universität Graz, Vorlesungsunterlagen, Wintersemester 2008.