INSTITUT FÜR INFORMATIK

Fachbereich Informatik und Mathematik

Diplomarbeit

Implementierung, Bewertung und Anwendung von maschinellen Lernverfahren zum automatischen Extrahieren von Instanzen einer Ontologie zum Aufbau einer A-Box

Burhan Hayber & Fabian Müller

Abgabe am 30. Mai 2011

eingereicht bei

Prof. Dr. Manfred Schmidt-Schauß

Professur für Künstliche Intelligenz und Softwaretechnologie

Erklärung

gemäss DPO §11 ABS.11

Das vorliegende Dokument wurde gemäß der nachfolgenden Aufteilung in selbständiger Arbeit verfasst von Burhan Hayber und Fabian Müller.

Kapitel	Burhan Hayber	Fabian Müller
1	X	X
2	X	
3		X
4	X	
5		X
6	X	
7		X
8	X	X
8.1	X	X
8.2	X	X
9	X	
10		X
11	X	X
12	X	X

Frankfurt am Main, den 30. Mai 2011

Burhan Hayber

Fabian Müller

Zusammenfassung

Diese Diplomarbeit umfasst eine Thematik, welche im Rahmen des Forschungsprojektes "OnToBau" der Fachhochschule Trier und des "Umwelt Campus Birkenfeld" angeboten und bearbeitet wird. Innerhalb dieses Projektes sollen, durch den Einsatz von maschinellen Lernverfahren, Informationen aus dem repräsentativen Dokumentenkorpus 'Rechnung', welcher durch die Projektleiter bereitgestellt wurde, extrahiert werden. Nach der Extraktion wird anhand eines theoretischen Modells gezeigt, wie die extrahierten Informationen in die bestehende Ontologie befüllt werden können.
Insbesondere soll getestet werden, wie die Güte der maschinellen Lernverfahren in Bezug auf die Extraktionsgenauigkeit der Daten bei durchgeführten Testfällen ist. Hauptziel dieser Arbeit wird jedoch sein, mit den Mitteln der maschinellen Lernverfahren, Konzepte zur Extraktion von Informationen aus Rechnungen zu entwickeln, zu implementieren und zu evaluieren.

Schwerpunkte

- Einarbeitung in das Forschungsgebiet der maschinellen Lernverfahren und Recherche nach verwandten Arbeiten im Bereich der Informationsextraktion.
- Entwicklung, Ausarbeitung und Darstellung von Konzepten zur Extraktion von Informationen aus Rechnungsdokumenten sowie Produktkatalogen.
- Prototypische Implementierung und ausführliche Evaluierung der erarbeiteten Konzepte.

Danksagung von Burhan Hayber

An dieser Stelle möchte ich mich besonders bei meinem Betreuer Herrn Professor Schmidt-Schauß bedanken, der mich während meiner Diplomarbeit betreut und umfangreich unterstützt hat, sowie allen Mitwirkenden des Projektes 'OnToBau' der Fachhochschule Trier.

Außerdem möchte ich mich herzlich bei meinen Eltern für die finanzielle Unterstützung bedanken. Des weiteren bei Sebastian Uhrig für die interessanten Beiträge und Stefan Kappes für die Unterstützung beim Umgang mit LaTeX. Mein ganz besonderer Dank gilt meinen Geschwistern, die mir bei vielen Formulierungen und auch bei der Korrektur der Diplomarbeit sehr hilfreich zur Seite standen.

Danksagung von Fabian Müller

Mein besonderer Dank gilt Herrn Prof. Dr. M. Schmidt-Schauß von der Universität Frankfurt für die Betreuung und Unterstützung während der Diplomarbeitszeit.
Ich bedanke mich auch bei den Professoren Herrn Prof. Dr. N. Kuhn und Herrn Prof. Dr. K. H. Bläsius, die im Rahmen des Projekts 'OnToBau' des Umwelt-Campus Birkenfeld der Fachhochschule Trier, diese Diplomarbeit initiiert haben und unseren Betreuern vor Ort, die uns gerade zu Beginn wertvolle Hinweise geben konnten.

Diese Diplomarbeit widme ich meiner Familie, ganz besonders meinen Eltern und meinem Bruder Christof Müller, die mich zu jeder Zeit meines Studiums und dieser Diplomarbeit unterstützt haben, sowie meinem Großvater der immer größtes Interesse für mein Studium und meine Arbeit zeigte.
Schließlich möchte ich mich bei all meinen Freunden und Kommilitonen bedanken, die mir während meiner Studienzeit, insbesondere in der Endphase meines Studiums moralisch zur Seite gestanden haben.

INHALTSVERZEICHNIS

KAPITEL

EINS

Einleitung

Diese Diplomarbeit umfasst eine Thematik, welche im Rahmen des Forschungsprojektes '*OnToBau*' der Fachhochschule Trier, des 'Umwelt-Campus Birkenfeld'[1], angeboten und bearbeitet wird. Ein Fernziel des Forschungsprojekt ist die Entwicklung eines komplexen Softwaresystems, welches kleine und mittelständische Unternehmen in wissensintensiven Arbeitsprozessen unterstützen kann. Im Vordergrund steht dabei die Organisation, der Umgang und das Management des aus dem Verlauf der unterschiedlichsten Arbeitsprozesse angereicherten Wissens, mit welchem durch eine Vielzahl von eingebetteten Anwendungen innerhalb des System umgegangen werden kann. Das Projekt zeigt am Beispiel des Bereichs 'Bauwesen' auf, wie "ein Wissensnetzwerk durch Extraktion von Informationen aus verschiedenen Quellen aufgebaut werden kann".

Ein Teilaspekt der Generierung von Wissen aus domänenspezifischen Vorgängen, wird mit Bearbeitung dieser Diplomarbeit abgedeckt. Die speziell konzipierte Anwendung innerhalb dieser Arbeit, wird Wissen, aus für die Domäne typischen Dokumenten, extrahieren und dieses in eine partiell bestehende Ontologie einpflegen.

Zunächst kann eine prinzipielle Erklärung zur Methodik der Generierung von Wissen innerhalb einer spezifischen Domäne gegeben werden. Der Ansatz entstammt dabei der grundsätzlichen Idee des Projektes 'OnToBau'.

Zur automatischen Extraktion des Wissens aus den unterschiedlichen Dokumenten

[1] http://www.umwelt-campus.de/ucb/index.php

und den damit verbundenen unterschiedlichsten Vorgängen innerhalb eines Unternehmens, kommen unterstützend Ontologien zum Einsatz. Die Ontologien können hierbei notwendiges Domänewissen zur Verfüng stellen, welches eine durchzuführende Informationsextraktion unterstützen kann. Ausgehend von einem vorhandenen Wissensnetzwerk, die bestehende Ontologie, sollen automatisch weitere Instanzen der Ontologie hinzugefügt werden. Zunächst kann dies in erster Linie durch die Modellierung von Klassen, Attributen und Relationen zwischen Klassen auf einer zuerst rein konzeptionellen Ebene, erfolgen. Diese Ebene wird durch sogenannte '*T-Boxen*' (siehe Unterabschnitt 2.2.4) repräsentiert.

Des Weiteren bieten die Ontologien die Möglichkeit, konkretes Wissen durch Instanziierungen abzubilden. Dies erfolgt in der Regel mit der Definition sogenannter '*A-Boxen*' (siehe Unterabschnitt 2.2.4).

Die Benutzung von Ontologien kann somit ein wichtiges Konzept für eine Vielzahl von Anwendungen sein und findet daher in vielen Bereichen der Informationsverarbeitung Einsatz, z. B. Anwendungen wie 'Informationsextraktion' und 'Informationsgewinnung', sowie die allgemeine 'natürlichsprachliche Verarbeitung' genannt werden. Ein aktuelles Anwendungsgebiet der Ontologien stellt das 'Semantic Web' (Semantisches Netz) dar, welches in Unterabschnitt 2.2.2 eingeführt wird.

Um nun eine Umsetzung der prinzipiellen Befüllung bzw. Anreicherung von Wissen aus konzeptionell erstellten Ontologien herbeizuführen, werden im Allgemeinen spezielle Lernmethoden eingesetzt.

Diesbezüglich wird ein besonderer Schwerpunkt die Anwendung sogenannter (maschineller) Lernverfahren (Hidden-Markov-Modell in Kapitel 4, Case-Based Reasoning in Kapitel 5) sein. Diese werden sinngemäß auf Instanzebene arbeiten und die Extraktion von Wissen aus Dokumenten der Domäne umsetzen. Dieser Vorgang wird die automatische Anreicherung einer Ontologie mit Instanzen, bei bereits vorgegebenem konzeptionellen Modell der Domäne, einleiten und weiterhin unterstützen. Die hier beschriebene Anwendung ist im Allgemeinen bekannt als '*Ontologiepopulation*' (Abschnitt 3.7).

Bei der Modellierung des domänenspezifischen Wissens in einer Ontologie, wird die Ontologiesprache 'Web Ontology Language', kurz 'OWL' (Unterabschnitt 2.2.3), ein-

gesetzt. Um die Aufgabe der Extraktion relevanter Informationen durchführen zu können, kann ein repräsentativer Testkorpus aus Dokumenten zur Verfügung gestellt werden. Dieser Testkorpus wird die Dokumentenklasse '*Rechnung*' aus dem Bereich 'Bauwesen' sein.

1.1 Motivation

In einem Unternehmen sammeln sich, infolge unterschiedlichster Arbeitsvorgänge und Prozesse die bewältigt werden, große Mengen von Daten und Informationen. Diese Daten können sowohl in Papierform, also in Form von schriftlichen Dokumenten, wie Rechnungen, Katalogbeständen und sonstigen textuellen Formaten, als auch bereits in digitaler Form, wie in Web-Veröffentlichungen oder in Datenbanken, vorliegen. Unabhängig vom Typ und der Struktur der Dokumente stellen diese Datenbestände Informationen der Domäne bereit, die aus unterschiedlichsten Arbeitsvorgängen entnommen werden können. So liefern unterschiedlichste Typen von Dokumenten allesamt Wissen zu einem Themenbereich, das Wissen ist demzufolge domänenspezifisch und steht in einem gemeinsamen Kontext. Werden durch spezifische Anwendungen nur für die Domäne relevante Informationen aus den jeweiligen Dokumenten gewonnen, kann eine (globale) "Wissensbasis" aufgebaut werden. Diese Wissensbasis bietet eine Wissensgrundlage für folgende Arbeitsvorgänge, darin enthaltenen Entitäten können folglich aus, im Laufe der nachfolgenden Prozesse resultierenden, weiteren neuen Daten dynamisch vervollständigt bzw. erweitert werden. Zum Zweck des effizienten Managements dieser Vielzahl von Informationen unterschiedlichster Herkunft, bietet sich der Einsatz von Ontologien, die das Wissen aus einer Domäne modellieren, strukturieren und vernetzt bereitstellen, an.
So können gerade im Managementsektor eines Unternehmens, Wissensnetzwerke (Ontologien) zur Erleichterung des Umgangs mit unternehmensrelevanten Informationen eingesetzt werden.
Es hat sich allerdings herausgestellt, dass gerade eine manuelle Erstellung einer Ontologie und eine darauffolgende Erweiterung dieser mit weiterem zusätzlichen Wissen zeitaufwendig, und aufgrund der Menge von Daten auch sehr arbeitsintensiv ist. Aus

diesem Grund haben sich eine Vielzahl unterschiedlichster Methoden zur automatischen Erstellung und zur Befüllung der Ontologien mit weiteren Instanzen entwickelt. Diese Methoden werden allgemein den Bereichen des 'Ontologielernens' und speziell der 'Ontologiepopulation' zugeordnet, und der Aspekt des "automatischen Lernens" kann von maschinellen Lernverfahren unterstützt werden.

Um der Problematik der aufwendig, da manuell durchgeführten Arbeit eines Anwenders, mit Hilfe einer Ontologie entgegen zu wirken, kann eine Systematik gefunden werden, welches die Arbeit und damit die Aufgabe einer "Population" einer Ontologie durch Instanzen automatisiert. Eine Automatisierung des Vorgangs mit Hilfe von maschinellen Lernverfahren kann sich dabei auf die Informationsextraktion aus Dokumenten beschränken. Die aus dem automatisierten Vorgang der Extraktion gewonnenen Daten werden dann mit geeigneten Ansätzen der Ontologiepopulation erkannt und die extrahierten Entitäten als Instanzen der Ontologie zugeordnet. Ideen aus diversen Ansätzen können berücksichtigt werden. So kann eine bestehende Ontologie die Extraktion von Informationen durch Einbringen von "Hintergrundwissen", unterstützen.

Gerade bei der Umsetzung der konzeptionellen Schritte, Informationsextraktion und Ontologiepopulation, ergibt sich eine besondere Herausforderung. Diese betrifft die Problematik des sogenannten 'Knowledge Acquisitation Bottleneck', die eine allgemeine Schwierigkeit genereller Vorgänge der "Wissensanhäufung" beschreibt. In der Probleminstanz des Anwendungssystems wird dies speziell die relative Begrenztheit der Quellen zur Anreicherung des weiteren Wissens sein.

Eine Auswertung nach Kriterien der Eignung eines möglichen Ansatzes zum Extrahieren von Informationen aus Text, kann in einer vereinfachten Testumgebung anhand von Testdaten vorgenommen werden. Ergebnisse einer Evaluation geben Aufschluss über die *Genauigkeit* und *Vollständigkeit* des Extraktionsvorgangs. Diese können mit einer repräsentativen Konkretisierung einer Testklasse, Beispiele der Problemklasse 'Rechnung', getestet und ausgewertet werden.

1.2 Zielsetzung

Um das ganzheitliche Vorhaben des effektiven Umgangs mit fachspezifischen Wissen aus unterschiedlichsten Prozessen in einer Domäne umsetzen zu können, soll unter anderem ein spezifischer Prozess, der in dieser Arbeit behandelt wird, beitragen. Eine Ansammlung verschiedenster textueller Daten, die als Dokumente einer Domäne vorliegen, soll eine "globale Wissensbasis" erweitern. Die Daten werden in digitale Form überführt und eine Extraktion relevanter Daten vollzogen. Diese werden an eine Stelle, die Ontologie mit ihren Konzepten und Attributen in Relation, eingepflegt, um einen späteren Zugriff und weiteren Umgang der Informationen zu ermöglichen. Somit erweitern die aus dem Extraktionsprozess gewonnenen Daten, bereits vorhandenes Wissen der Domäne. Der zugrundeliegende Prozess soll folglich in automatisierter Weise durchgeführt werden, da eine manuelle Durchführung sowohl einen hohen Arbeits-, als auch Zeitaufwand bedeutet. Diesbezüglich werden maschinelle Lernverfahren, wie sie in der Künstlichen Intelligenz eingesetzt werden, eingebunden.

Der Schwerpunkt der Ausarbeitung wird daher darauf liegen, geeignete Verfahren zur Extraktion von Informationen aus domänentypischen Dokumenten zu finden. Mit den dann automatisch extrahierten Daten aus einem bekannten Testkorpus sollen in einem zweiten Schritt weitere neue Instanzen einer gegeben Ontologie aus dem Anwendungsgebiet generiert werden.

Zum Zweck der Informationsextraktion werden zwei ausgewählte Lernverfahren auf Eignung zur spezifischen Anwendung in der Domäne "Bauwesen" getestet und ausgewertet. Die Lernverfahren, die zum Einsatz kommen werden, sind zum einen das '*Hidden Markov Model*', und zum anderen das '*Case-Based Reasoning*'. Die Verfahren werden die Aufgabenstellung durch zwei prinzipiell unterschiedliche Vorgehensweisen bewältigen, so dass eine klare Gegenüberstellung erfolgen kann.
Ersteres ist ein stochastisches Verfahren, welches unter anderem Anwendung im Bereich der 'Natürlichsprachlichen Verarbeitung' für unterschiedlichste textuelle Dokumentstrukturen erfahren hat. Diese Annahme gilt es, in der konkreten Problem-

klasse zur 'Extraktion von Informationen aus Rechnungsdokumenten' aufzuzeigen. Zweiter Ansatz wird besonders im Anwendungsbereich der 'Help-Desk Systeme' aber auch z. B. im Gebiet des 'Information Retrieval', eingesetzt. Die Idee des 'textuellen Case-Based Reasoning' liefert dabei einen fundamentalen Ansatz zur Umsetzung des Systems. So gilt es, die Eignung des Einsatzes in besagter Problemklasse ebenfalls zu prüfen.

Beide Ansätze werden auf die Problemdomäne angepasst. Hierbei muss zunächst eine geeignete interne Repräsentation der Daten aus dem Testkorpus für die Systeme gefunden werden, bevor ein eigener Implementierungsanteil eine Schnittstelle zwischen dem Framework der Lernverfahren und der Problemdomäne herstellen wird.
Der Gesamtprozess wird anhand eines repräsentativen Beispiels, der strukturierten Dokumentenklasse 'Rechnung' durchgeführt. Die Struktur und damit verbundenene feststellbare Regelmäßigkeiten des repräsentativen Dokuments, wird von Bedeutung sein.

Hinsichtlich der anschließend durchgeführten Evaluation der in der Testimplementation angewandten maschinellen Lernverfahren, werden die Resultate, der eigens implementierten Systeme, ausgewertet und verglichen. Dabei sollen diese möglichst gute Erfolgsraten bei der Erkennung und Extraktion von relevanten Informationen aus typischen Dokumenten liefern.
Ein einfacher Kriterienkatalog wird die Bewertung der Resultate der einzelnen Systeme mittels eigens aufgestellten Auswertungskriterien ermöglichen. Die Kriterien sollen zur Einschätzung der Leistung der angewandten Systeme dienen. So werden die bekannten Retrievalmaße "Precision" und "Recall" zur Beurteilung der Güte der Anwendungen beitragen. Ein Kriterium liefert Ergebnisse bei einer minimale Anzahl von Trainingsdaten bzw. minimalen Grad an initialen Wissen des Systems. Ferner wird abschließend ein Test auf einem komplett unbekannten bzw. nicht aufbereiteten Testkorpus durchgeführt, um das Verhalten der Systeme einstufen zu können. Folglich gilt es zu zeigen, ob eine Anpassung der Systeme und damit eine Übertragbarkeit der Anwendungen auf ähnliche Dokumentarten der Problemklasse gewährleistet werden kann. Insbesondere kann geklärt werden, in wie weit die Performanz der Prozesse

zur Informationsextraktion von der Korrektheit der Eingabedaten abhängt.

Eine Verwertung der extrahierten Daten wird im abschließenden Arbeitsschritt erfolgen. Eine theoretische Umsetzung wird dabei eine potentielle "Methodik der Befüllung" der Ontologie beschreiben, die es ermöglicht, unter Verwendung der definierten Domänenontologie weitere Instanzen, die in einer 'A-Box' definiert werden können, auszubilden. Hierbei kann eine theoretische Modellierung gefunden werden, und die Instantiierung im Sinne eines formalen 'A-Box' Statements vorgenommen werden. Hierzu werden Ansätze der 'Ontologiepopulation' benutzt.

1.3 Gliederung

Zunächst wird die eigentliche Problemstellung und die darauf basierende Idee dieser Ausarbeitung verdeutlicht, um in der Zielsetzung den Ansatz zur Anwendung von Ontologien in Verbindung mit maschinellen Lernverfahren zur automatischen Informationsextraktion zu begründen und konkrete Umsetzungen zu beschreiben.
Die Arbeit wird dabei im ersten Kapitel einen Einblick in das allgemeine Themengebiet der 'Ontologien' geben. Hierbei wird insbesondere auf die Ontologiesprache 'OWL' (Unterabschnitt 2.2.3) eingegangen, die generell für die Modellierung und den Aufbau einer Ontologie genutzt werden kann. Zusätzlich wird der Aspekt einer Beschreibungslogik zur Formalisierung und Konstruktion einer Ontologie, in Form der Statements 'T-Box' und 'A-Box', aufgezeigt.

Im Weiteren wird ein Überblick über den aktuellen Stand der Forschung, durch Aufzeigen theoretischer Konzepte und Ansätze aus dem Bereich des 'Ontologielernens' (Kapitel 3), vermittelt. In diesem Kapitel werden außerdem ontologiebasierte Methoden zur Wissensextraktion vorgestellt und kategorisiert. Darüber hinaus wird auf spezifische Anwendungen aus dem Bereich des 'Ontologielernens' eingegangen. Insbesondere können typische Verfahren des Ontologielernens und der Ontologiepopulation aufgezeigt werden.

Die darauffolgenden Kapitel dienen zur Vorstellung der theoretische Grundlagen der

im Prozess zur Informationsextraktion eingesetzten maschinellen Lernverfahren. Dabei wird zum einen explizit das Lernverfahren des 'Hidden-Markov Modell' (Kapitel 4), und zum anderen der Ansatz des 'Case-Based Reasoning' (Kapitel 5) beschrieben. Neben den grundsätzlichen Methodiken der Verfahren, werden jeweils Anwendungen und Modelle zur Informationsextraktion vorgestellt, die einen wesentlichen Einfluss auf den Implementierungsteil dieser Arbeit haben werden.

Die Aufbereitung des theoretischen Grundwissen zu den beiden Verfahren wird die Erstellung des Konzepts und den Entwurf der Systeme, die jeweils auf den beiden Lernansätzen basieren, ermöglichen. Die Lernverfahren werden dabei auf die Spezifikationen und die Problemstellung der Domäne angepasst, so dass die spezifische Anwendung der Informationsextraktion aus domäneeigenen Dokumenten durchgeführt werden kann. So wird jeweils ein Design der beiden auf maschinellen Lernverfahren basierenden Systeme in diesen Kapiteln angegeben und ein auf die Problemstellung angepasstes Prozessmodell konzipiert.
Die Entwürfe zur Anwendung der eigentlichen Aufgabenstellung, der Extraktion von relevanten Informationen zur Population einer Ontologie, führen in den folgenden Kapiteln zur eigentlichen Beschreibung der Implementierung der beiden Systeme. Es werden dabei Klassenbeschreibungen der umgesetzten Systeme vorgenommen und auf Funktionalitäten der Anwendungen eingegangen. Die Umsetzung der Systeme wird auf schon einsatzfähigen Frameworks basieren, die durch Interfaces auf die Domäne und damit auf das Extraktionsproblem abgestimmt werden.

In den Kapiteln der abschließenden Evaluierung wird eine Testphase unter vorgegebenen Bedingungen und Kriterien durchgeführt. Die dazu durchgeführte Erstellung eigener Kriterien dient der Bewertung der angewendeten Lernverfahren und der Prüfung auf Anwendbarkeit in der Problemstellung. Dazu wird ein Abschlussverfahren mit unbekanntem Testkorpus durchgeführt. Infolgedessen wird eine qualitative Auswertung der Güte des Lernverfahrens nach bekannten Maßen des Information Retrieval (Precision und Recall) erfolgen.

Ein weiterer Abschnitt wird die Ergebnisse und Erkenntnisse der Auswertung auf-

greifen, und eine theoretische Umsetzung der Population der Ontologie mit den extrahierten Daten beschreiben. Hierbei geht es um die Beschreibung einer geeigneten Methodik zur 'Ontologiepopulation', welche die aus dem Vorgang der Extraktion resultierenden Konzepte und Attribute in die Ontologie instantiieren wird. Die relevanten Informationen werden in die Konzepte der Ontologie klassifiziert und zur Befüllung durch Instanzen der Domäneontologie genutzt. Zuvor wird eine einfache Beschreibung der Ontologie aus der Domäne "Bauwesen" gegeben, anhand dieser die Population von Instanzen vollzogen wird.

Das abschließende Kapitel nimmt einen Vergleich der in der Arbeit umgesetzten Anwendungen vor. Diesbezüglich wird ein kurzes Fazit zur Beurteilung der Verfahren und der anschließender Abschätzung der Eignungsfähigkeit des Einsatzes der Ansatze innerhalb der Problemdomäne gezogen.
Zuletzt wird ein Ausblick bezüglich weiterer Verbesserungen und neuer Ideen, aufgrund der Erkenntnisse dieser Arbeit, für weitere Projekte im Anwendungsbereich gegeben.

KAPITEL

ZWEI

Grundlagen

2.1 Projekt OnToBau

Die Informationen zum Projekt OnToBau entspricht der Kurzfassung von Markus Schwinn [35].

Das Forschungsprojekt Ontologie basierte Prozessunterstützung im Bauwesen (OnToBau) hat das Ziel mit Hilfe eines ontologie-basierten Wissensarchivs kleinen und mittelständischen Unternehmen die Möglichkeit zu geben, ihr Wissen zu klassifizieren, zu archivieren und effektiv einzusetzen. Dieses Archiv soll Mitarbeiter innerhalb eines Unternehmens proaktiv mit Informationen versorgen und somit bei der Bearbeitung von Geschäftsprozessen unterstützen. Diese Arbeit konzeptioniert ein Informationsextraktions-System (IE-System) zum Aufbau des Wissensarchivs. Die Prozessunterstützung der Angebotserstellung in der Heizungsund Sanitärfirma Schottler GmbH ist Ausgangspunkt für die Anforderungsanalyse an das IE-System. Damit die unterschiedlichen Ressourcen innerhalb eines Unternehmens (E-Mails, Kalendereinträge, Papierakten usw.) einheitlich verarbeitet werden können, wird eine XML-basierte Repräsentationssprache (OnToBau Representation Language – ORL) entwickelt. Zur Transformation der Ressourcen in die ORL werden Konverter eingesetzt. Anschließend erfolgt eine Vorverarbeitung (Preprocessing) der aus der Ressource entnommenen textuellen Informationen. Dieses Preprocessing umfasst mehre-

re Bearbeitungsschritte (Segmentierung, Stoppwortfilterung, usw.). Diese in sich geschlossenen Preprocessing-Module (Filter) verarbeiten den Text sequenziell und bilden damit eine Bearbeitungskette (Pipeline). Diese Pipeline soll eine größtmögliche Varianz an Kombinationsmöglichkeiten der Filter garantieren. Deshalb wird das Pipes-and-Filters-Entwurfsmuster als Grundlage zur Konzeptionierung des Preprocessing herangezogen und mit notwendigen Erweiterungen versehen. Die Kernkomponente des IE-Systems stellt das Inferenzsystem dar, welches Informationen aus den vorverarbeiteten Ressourcen extrahieren soll. Dazu werden Konzepte zur Realisierung eines Klassifikationsmoduls zur Extraktion von Dokumentrelationen und Informationen mittels Templates und zur Bestimmung von Indextermen entworfen. Es wird vorgeschlagen, die Modellierung der Templates und Klassen mit Hilfe einer deskriptiven Sprache vorzunehmen, deren Anforderungen ebenfalls erarbeitet werden. Die extrahierten Informationen werden im Wissensarchiv durch Ontologien repräsentiert. Es werden zwei aktuelle Ontologiesprachen auf Tauglichkeit hin untersucht. Aufgrund der Ergebnisse des Vergleichs wird sich für die Web Ontology Language zum Aufbau des Wissensarchivs entschieden. Abschließend wird mit Hilfe eines Testkorpus von Angebots- und Rechnungsdokumenten ein prototypisches System zur Extraktion von Relationen und deren Visualisierung in einer grafischen Benutzungsoberfläche implementiert.

2.2 Ontologie

In diesem Kapitel wird ein Überblick über das Thema "Ontologie" gegeben. Es wird im Folgenden erklärt, warum das Thema so wichtig ist und welche Art von Repräsentationsmöglichkeiten überhaupt existieren.
Es wird des Weiteren gezeigt, welche Einsatzmöglichkeiten für Ontologien existieren. Im letzten Teil dieses Kapitels wird dann die Unterteilung der Ontologie in einer T-Box und A-Box erklärt.

2.2.1 Definitionen

Der Begriff **Ontologie** ist ein überlieferter Begriff aus der Philosophie und beschäftigt sich mit dem Thema des "Seienden". Schon in der Antike hat sich beispielsweise der Philosoph Aristoteles mit dem Begriff der Ontologie beschäftigt. So stellt Aristoteles zur Klassifikation aller Dinge, über die Aussagen getroffen werden können ein System von Kategorien auf (siehe Abbildung 2.1).

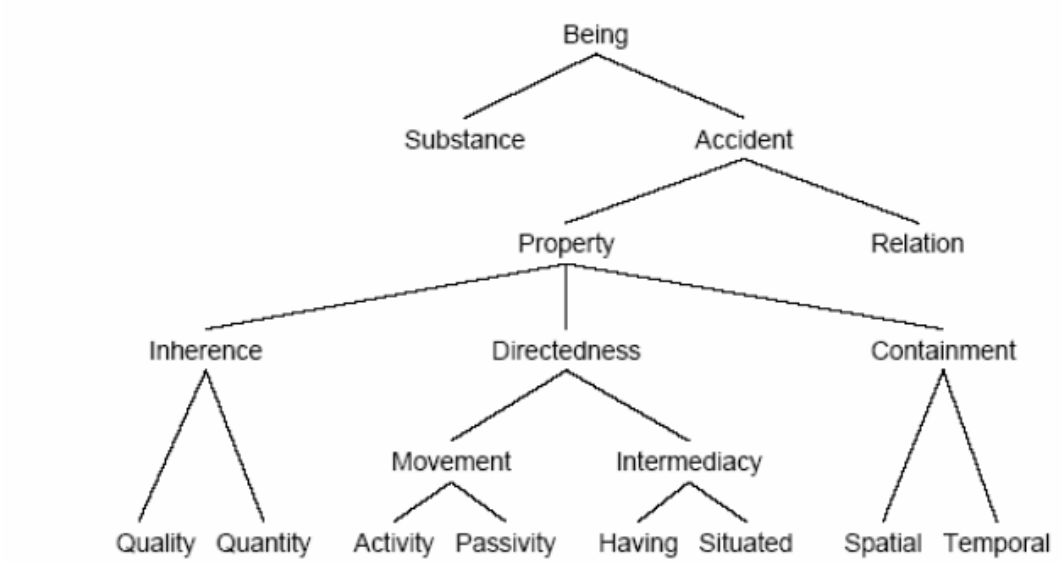


Abbildung 2.1: Eine Ontologie aus der Sicht des berühmten Philosophen Aristoteles Quelle: http://users.minet.uni-jena.de/∼sack/WS0607/semanticweb.htm

In der Informatik jedoch steht der Begriff "Ontologien" für eine konzeptuelle Formalisierung von Wissensbereichen (Domänen und Subdomänen), beziehungsweise sie stellt ein formalisiertes Modell der Welt (auch Domäne genannt) oder ein Teil der Welt (Subdomäne) dar. Es kann als eine Wissensdatenbank betrachtet werden. Damit ein Computersystem die Daten in der Ontologie (Wissensdatenbank) verwerten kann, muss die Ontologie formal definiert sein.
Man nehme an, daß der Computer und der Mensch das selbe Sprachvokabular haben, um gemeinsam miteinander zu kommunizieren. Über Symbole assoziieren sie Gegenstände aus der realen Welt. Nur was passiert, wenn die Symbole nicht eindeutig von einander unterschieden werden können ? Genauer gesagt, wenn es für ein und das gleiche Symbol zwei von einander verschiedene Bedeutungen gibt? Es

gibt für das Wort "Bank" zwei Bedeutungen. Ist nun die Bank gemeint, wo man sich etwa daraufsetzen kann oder ist die Bank gemeint, bei der man sein Geld einzahlen kann? Es geht hervor, dass einzelne Symbole im Allgemeinen nicht ausreichen und nur aus dem Kontext heraus ersichtlich wird, was eigentlich gemeint ist. Eben an dieser Stelle kommen Ontologien zum Einsatz. Sie stellen sogenannte Wissensmodelle bereit, die auf der einen Seite eine konzeptuelle und auf der anderen Seite eine terminologische Klarheit schaffen. Begriffe werden eindeutig definiert, so dass klar ist, was der Begriff beutetet. Eine Ontologie ist wie eine Bedienungsanleitung für eine ganze Sprache innerhalb einer Domäne oder Subdomäne, wobei der Inhalt der Sprache Domänen abhängig ist.
Die Funktionalität einer Ontologie lässt sich in vier Punkten zusammenfassen: [21]

1. Kommunikation
2. automatische Schlussfolgerung
3. Repräsentation
4. Wiederverwendung von Wissen

Eine Ontologie wird als Kommunikationsmittel benutzt, indem man eine gemeinsame Basis für die Anschauung der Dinge (Objekte der Realen Welt) für beide Kommunikationspartner schafft. Zusätzlich können sich Programme der Ontologie bedienen, um logische Schlüsse anhand der Ableitungsregeln die in einer Ontologie definiert sind, selber ziehen zu können. Wie schon auch vorher erwähnt, eigenen sich Ontologien gut als Wissensdatenbanken, was eine Wiederverwendbarkeit der Daten ermöglicht. Jedoch sollte die Ontologie selber nicht zu spezifiziert sein, da sie ansonsten schlechter wieder verwendet und auch schlechter erweitert werden kann. Um Übersetzungsfehler zu vermeiden sollte die Ontologie außerdem konsistent arbeiten.
Folgendes Bild soll ein Beispiel für eine Ontologie darstellen: Anhand der Abbildung 2.2 kann man einige wichtige Strukturen erkennen, die bei Ontologien sehr häufig vorkommen. Dabei stellen die Vierecke in der Grafik (zum Beispiel Lebewesen, Politiker) die *Konzepte* dar. *Konzepte* sind im Grunde genau Klassen. Und Klassen wiederum sind Mengen von Instanzen mit gleichem beziehungsweise ähnlichen Eigenschaften [18].

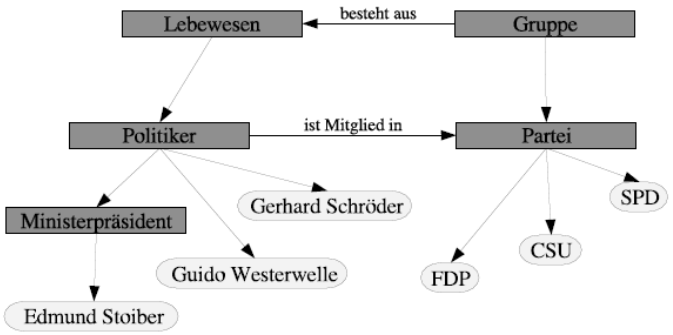


Abbildung 2.2: Eine Beispielontologie aus der Politik-Domäne [46]

Die Kanten des Graphen stellen die *Relationen* dar. Sie stellen die Zusammenhänge in der Ontologie dar. Die unbenannten Beziehungen stellen die sogenannten "taxonomischen Beziehungen" dar. In diesem Graphen zum Beispiel, ist das Konzept Partei ein Unterkonzept des Konzepts Gruppe. Wie in Objekt orientierten Programmiersprachen auch, werden hierbei auch die Eigenschaften (Properties, Attributen) einer Klasse/Konzept weiter vererbt.
Die abgerundeten Vierecke stellen *Instanzen* dar. Instanzen sind Objekte die zu einer Klasse gehören. Zum Beispiel ist "Gerhard Schröder" eine Instanz der Klasse Politiker. Die Klasse Politiker muss nicht immer ein Konzept sein, sondern kann sich je nach Kontext ändern. In diesem Beispiel ist Politiker vielleicht ein Konzept, aber in einem anderen Kontext betrachtet, kann es z. B. eine Instanz der Klasse Mensch sein.
„*Etwas formaler lässt sich eine Ontologie beschreiben als ein Tupel*": [43]

$$\vartheta = (C, R, H_C, rel, A_\vartheta) \tag{2.1}$$

hierbei ist

C eine Menge von Konzepten in der Ontologie gemeint. Konzepte die auch Entitäten genannt werden, stellen Objekte beziehungsweise Dinge der Realen Welt dar. Die Konzepte einer Ontologie hängen von der jeweiligen Domäne ab.

R bezeichnet die Menge der Relationen. Also die Menge der nicht-taxonomischen Beziehungen. Wobei die Funktion rel: $R \rightarrow C \times C$ die Bezeichner der Relationen auf die tatsächlichen Relationen abbildet.

A_ϑ stellt die Menge der Axiome dar. Diese Axiome beschreiben die zusätzlichen Bedingungen an die Ontologie.

$H_C \subseteq C \times C$ ist die Menge der taxonomischen Beziehungen zwischen den Konzepten. Dadurch ergeben sich in der Ontologie Hierarchische Strukturen.

Unter anderem werden in [13] drei Kriterien für Ontologien vorgestellt :

Deckung Die Konzepte einer Domäne müssen vorhanden sein. Das heisst, dass die Ontologie ausreichend bevölkert sein muss.

Konsens Unternehmen welche sich entschliessen in einer gemeinsamen Domäne zusammenzuarbeiten, müssen sich zuerst über viele grundlegende Sachverhalte einigen. Dazu gehört auch ein Konsens über die Geschäfts-Domäne. Diese gemeinsame Sichtweise muss sich in der Domänen-Ontologie widerspiegeln.

Zugänglichkeit Eine Ontologie muss leicht zugänglich sein, d.h. es werden Werkzeuge benötigt um Ontologien leicht in Anwendungen integrieren zu können. Dies ist nötig, um auch die Vorteile einer Ontologie schnell demonstrieren zu können, wie z. B. die verbesserte Möglichkeit Informationen über das Web auszutauschen.

2.2.2 Anwendungsgebiete

Wie schon auch im obigen Unterkapitel erwähnt, benutzt man Ontologien im wesentlichen zur Wissensrepräsentation.
Das heißt, dass man semantisches Wissen speichern beziehungsweise darstellen kann. Das hat den Vorteil, daß man sich bei der Kommunikation auf eine gemeinsame semantische Basis konzentrieren und somit schlußfolgern kann, welchen semantischen Inhalt die Nachricht des anderen Gesprächspartner besitzt. In unserem Beispiel 2.2 würde zum Beispiel ein System erkennen, dass sich bei dem Satz „Edmund Stoiber redet mit Guido Westerwelle“ um zwei Politiker handelt die miteinander reden.

Aufgrund dieser Eigenschaft des "Verstehens" werden Ontologien in verschiedenen Bereichen eingesetzt. Die Anwendungsgebiete sollen in diesem Unterkapitel erwähnt werden.

Semantic Web

Einer der vielen Einsatzgebiete von Ontologien ist das so genannte "Semantic Web", auch "Web Ontologies" genannt. Das Semantic Web ist eine Vision von Tim Berners-Lee der unter anderem auch das World Wide Web erfunden hat [4]. Es geht beim Semantic Web darum, dass die Daten im Netz automatisch bzw. semi-automatisch von Maschinen verarbeitet werden können. Eine Maschine soll also im Stande sein, die anfallenden Informationen (z. B. Dokumente) zu lesen, verarbeiten, transformieren und auf sinnvolle Art zusammenstellen. So können Maschinen auf diesen Daten sogar operieren.
Damit das Semantic Web überhaupt möglich ist, müssen semantische Metadaten zu Informationsquellen hinzugefügt werden, sodass Maschinen die Daten anhand der beschreibenden semantischen Informationen effektiv verarbeiten können. Dadurch kann ein System Rückschlüsse über die Daten ziehen. Das heißt, dass System weiß, um was es sich bei der Datenressource handelt und in welchem Zusammenhang sie mit anderen Daten steht.
Eine Beispiel Technologie ist HTML (HyperText Markup Language). HTML-Dokumente können beispielsweise Informationen über den Autor einer Webseite enthalten oder relevante Schlüsselwörter, mit denen Suchmaschinen die Dokumente leichter finden können. Auf folgender Abbildung 2.3 lässt sich erkennen, wie eine solche Web Ontologie funktioniert.

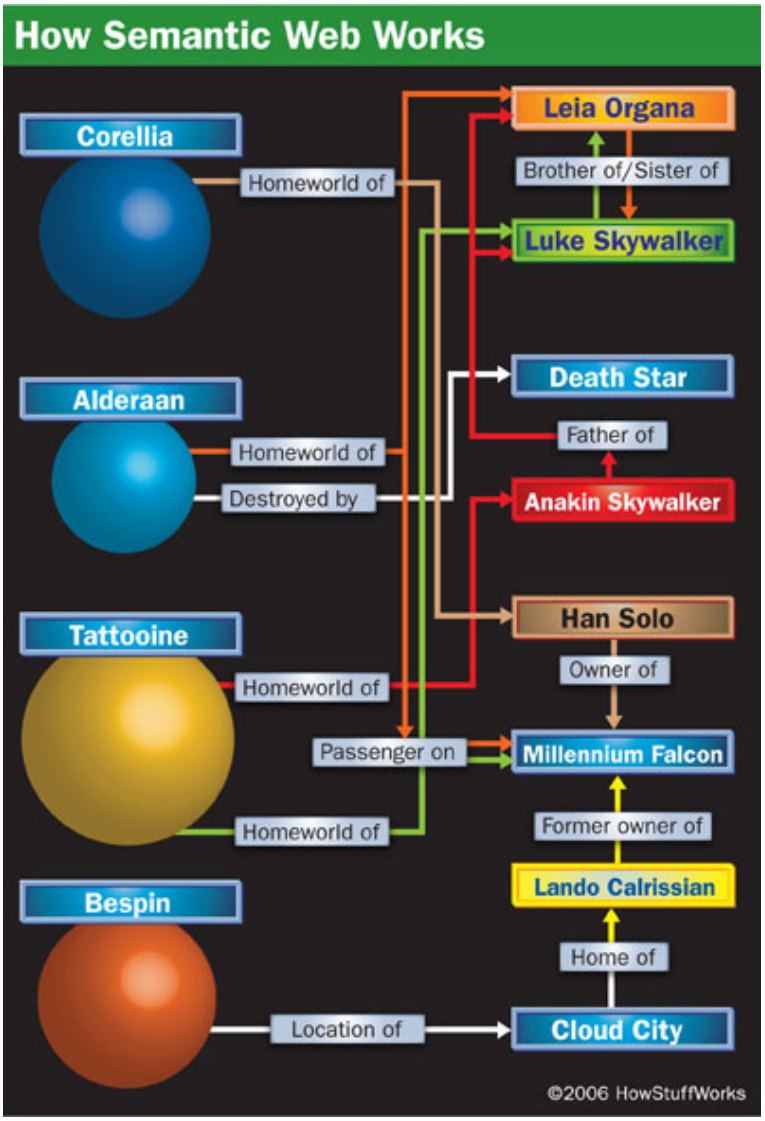


Abbildung 2.3: Eine Beispiel-Webontologie, die zeigen soll, wie das Semantic Web arbeitet. Quelle: http://static.howstuffworks.com/gif/semantic-web-4.jpg

Wissensmanagement

Hierbei dient eine Ontologie als eine Art "Unternehmensgedächtnis" und versorgt die Mitarbeiter eines Unternehmens mit Wissen(Informationen) zur Erledigung ihrer Aufgaben. Des Weiteren dient sie als Unterstützung für die Visualisierung, Suche, Abfrage und Personalisierung von Wissen. Neben der lexikalischen Suche ermöglicht es eine Ontologie auch semantische Recherchen durchzuführen. In der Praxis einge-

setzte Wissensmanagement-Lösungen sind zum Beispiel der *SemanticGuide* und der *SemanticMiner*[1].

Text Mining

Auch im Bereich des Text Mining kommen Ontologien zum Einsatz, wobei es voralem darum geht, aus den verarbeiteten Texten Informationen zu extrahieren. Ganz besonders bei natürlichsprachlichen Texten lohnt es sich eine Ontologie einzusetzen, um zusätzlich Hintergrundwissen bei der Extrahierung von Informationen zur Verfügung zu haben. Folgendes Beispiel sei gegeben :

> *Später spricht Bundeskanzler Gerhard Schröder in Köln.*
> *Er hatte* ***Stoiber*** *schon gestern "Realitätsverlust" vorgeworfen.* ***Der bayrische Ministerpräsident*** *hatte die Arbeitsmarktpolitik der rot-grünen Regierung für den Erfolg der NPD verantwortlich gemacht.*

In diesem Beispiel besteht das Problem darin, dass man ohne Hintergrundwissen nicht wissen kann, ob sich "Ministerpräsident" auf Gerhard Schröder oder auf Edmund Stoiber bezieht. Doch durch den Einsatz einer Ontologie wird das Problem behoben (indem es Hintergrundwissen zur Verfügung stellt).

2.2.3 Websprache OWL

Das World Wide Web Consortium entwickelte im Rahmen der Notwendigkeit für das Semantic Web die Sprache *Web Ontology Language* (kurz OWL). Sie wurde entwickelt, damit es anhand dieser formalen Beschreibungssprache möglich ist, Ontologien zu erstellen und weiter verteilen zu können. Dadurch sind Computerprogramme zum Beispiel in der Lage, eine Domäne zu verstehen. So ähnlich ist es auch bei der Sprache XML. XML ist eine Sprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten. XML wird unter anderem für den Plattform unabhängigen Austausch von Daten zwischen Computersystemen eingesetzt. Beruhend auf der Syntax von XML ist RDF ein flexibles System für die Codierung, den Austausch und die Wiederverwendung von Metadaten. Und da OWL syntaktisch auf XML

[1] http://www.ontoprise.de/de/loesungen/semanticguide/

und RDF (Resource Description Framework) basiert, ermöglicht es die maschinelle Verarbeitung der in OWL repräsentierten Informationen. OWL-Dokumente sind im wesentlichen RDF-Dokumente, die aus einem Header, Body und einem Footer bestehen.

Sie sind wie folgt aufgebaut :

Header

- enthält allgemeine Informationen über die Ontologie
- Namensräume (rdf, rdfs, owl, dc, etc)
- enthält Kommentare
- Versionskontrolle
- Einschließen anderer Ontologien

Ein Beispiel für den Kopf eines OWL-Dokuments

```
<owl:Ontology rdf:about="">
 <rdfs:comment rdf:datatype="http://www.w3.org/2001/
 XMLSchema#string">
 SWRC Ontologie in der Version vom Dezember 2005
 </rdfs:comment>
 <owl:versionInfo>v0.5</owl:versionInfo>
 <owl:imports rdf:resource="http://www.semanticwebgrundlagen.
 de/foo"/>
 <owl:priorVersion rdf:resource="http://ontoware.org/
 projects/swrc"/>
</owl:Ontology>
```

Body

- Im Body werden die Klassen und Properties der Ontologie spezifiziert
- Definieren von Objekteigenschaften
- Definieren von Datentypeigenschaften

Ein Beispiel-Code für den Body eines OWL-Dokuments

```
<owl:ObjectProperty rdf:ID="wohntIn">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range rdf:resource="#Gebiet"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="bewohntStadt">
<rdfs:subPropertyOf rdf:resource="#wohntIn"/>
<rdfs:range rdf:resource="#Stadt"/>
</owl:ObjectProperty>

<owl:DataTypeProperty rdf:ID="Alter">
<rdfs:domain rdf:resource="#Person" />
<rdfs:range rdf:resource="&dt;Alter"/>
</owl:DataTypeProperty>
```

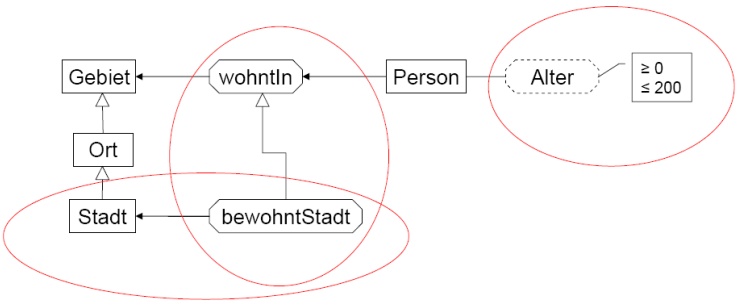


Abbildung 2.4: Zu obigem Code-Beispiel der dazu gehörige Graph [24]

Footer Der Footer ist ein Tag der das Ende einer Ontologie kennzeichnet und hat folgende Form *</rdf:RDF>*.

2.2.4 T-Box und A-Box

In der Ontologie besteht die Wissensbasis im wesentlichen aus zwei wichtigen Komponenten. Das ist einmal die T-Box (terminological Box) und die A-Box (assertional Box). Die T-Box enthält hierbei das Wissen über die Konzepte einer Domäne, also das so genannte terminologische Wissen. Die A-Box hingegen enthält das Wissen über Entitäten oder Instanzen dieser Konzepte und deren Beziehungen untereinander. Das wiederum repräsentiert den Zustand der modellierten Welt. In Bezug auf Objektorientierte Programmiersprachen wäre in diesem Falle unsere T-Box die Wissensbasis über unsere Klassen, die die Konzepte der Domäne darstellen würden. Diese Klassen besitzen wiederum Unterklassen, die die Eigenschaften beziehungsweise die Attribute und Methoden der Oberklasse erben. Dieses Wissen über die Klassen und Unterklassen und ihre zugehörigen Eigenschaften ergeben eben eine T-Box. Die A-Box wäre dann hierbei die Instanzen dieser Klassen.
In diesem kleinen Code Segment wäre zum Beispiel Affe eine Instanz von der Klasse Tier und mit dem Namen Spencer.

```
Tier Affe = new Tier(); // neue Tier-Instanz mit dem Namen "Affe"
Affe.Name = Spencer;    // dem "Namen"-Attribut wird der Wert
                           "Spencer" zugewiesen
```

Folgende Grafik soll das Prinzip von der *T-Box* und *A-Box* verdeutlichen.

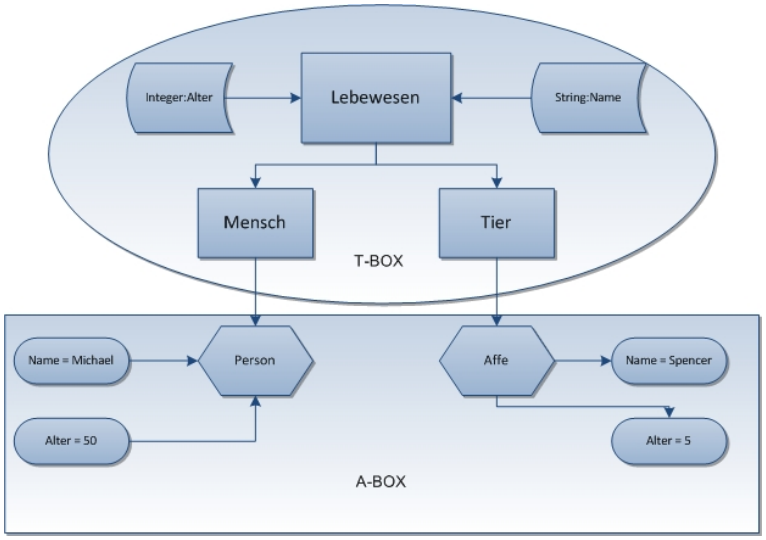


Abbildung 2.5: Verdeutlichung des Prinzips einer T-Box und A-Box anhand einer Beispiel-Ontologie

In Abbildung 2.5 ist zu erkennen, wie eine mögliche Ontologie aufgebaut sein kann. Die Rechtecke auf dem Bild stellen die Konzepte dar und die Rechtecke unter dem Konzept Lebewesen sind Unterkonzepte, welche die Attribute Alter vom Typ Integer und Name vom Typ String erben. Das alles zusammen (die Konzepte zusammen mit den Unterkonzepten und Attributen) ergeben die *T-Box*. Die auf der Grafik zu sehende Hexagons stellen die Instanzen dar. Eine Person ist beispielsweise eine Instanz von dem Konzept Mensch und befüllt die Attribut Werte "Name" mit Michael und "Alter" mit 50. Somit bildet die Instanz Person zusammen mit den dazugehörigen Werten eine *A-Box*.

KAPITEL

DREI

Ontologielernen

Die Anwendung *'Ontologielernen'* versucht ontologisches Wissen aus unterschiedlichsten Ressourcen und Datenformen zu gewinnen.[49] In der Wissenschaft allgemein kann das Lernen von Ontologien auch als Unteraufgabe der Informationsextraktion angesehen werden. Ein entsprechender Prozess zum Lernen einer Ontologie kann aus den Komponenten Extraktion von Informationen, Erzeugung von ontologischen Entitäten und Akquisition von Wissen bestehen.[30, 49]
Da der Aufbau einer Ontologie generell manuell erfolgte, und der damit verbundene Prozess des 'Ontology Engineering'[1] zeitaufwändig war, wurden im Forschungsgebiet 'Künstliche Intelligenz' Verfahren entwickelt, welche die Konstruktion von Domänen automatisieren und zugleich vereinfachen.
Bei einer automatischen Generierung von domänenspezifischen Ontologien, werden im Allgemeinen natürlichsprachliche Verarbeitungstechniken und maschinelle Lernverfahren angewandt. So kann aus den unterschiedlichsten Dokumentarten Wissen extrahiert werden. In der Regel kann dieser Vorgang sowohl mit statistischen als auch mit linguistischen Verfahren umgesetzt werden.[43, 30]

[1] Entwicklung einer Ontologie

Ein typisches Lernsystem übernimmt folgende Aufgaben: [29]

- Erkennung von Konzepten und deren taxonomischen Relationen,
- Extraktion von nicht-ontologischen Relationen zwischen Konzepten,
- Identifikation von Attributen und das Generieren von Instanzen der definierten Ontologie.

Die Ontologien akquirieren durch automatisierte Prozesse weiteres Wissen und nehmen an Umfang und Struktur zu.
Das Lernen einer Ontologie kann in einem Flussdiagramm vereinfacht festgehalten werden. Der Prozess durchläuft mehrere Schritte, wie in folgender Abbildung zu sehen ist.[49]

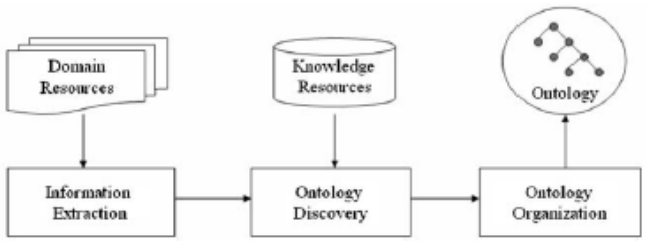


Abbildung 3.1: Flussdiagramm zum Erstellen einer Ontologie [49]

3.1 Ideologie

Das 'Ontologielernen' (engl.: ontologie learning) verweist auf die automatisierte Entdeckung und Erschaffung von ontologischem Wissen unter Verwendung von maschinellen Lerntechniken. Wird außerdem eine manuelle Interaktion des Anwenders eingebettet, so kann Wissen in einem großen Umfang und in relativ kurzer Zeit generiert werden. Damit unterstützt das Ontologielernen die Verbesserung und Erweiterung von existierenden Ontologien durch Einbindung neu erlangtem Wissen.
Wie bereits im Grundlagenkapitel 2 beschrieben, ist ein spezielles Einsatzgebiet der Ontologien im Bereich des Semantik Web festzustellen. Gerade am Beispiel des

'World Wide Web' mit seiner Vielfalt an Informationen zeigt sich, dass die Erstellung und Konstruktion einer Ontologie extrem zeitaufwändig und vor allem sehr arbeitsintensiv ist.
Ziel ist es also, effiziente Methoden zu entwickeln, um die Erstellung von Ontologien automatisch zu erlernen und später, schon definierte Ontologien mit weiterem Wissen zu befüllen. Im Mittelpunkt steht ein ständiges Bestreben nach Automatisierung der Konstruktionsprozesse und Wissensakquirierung. Eine Vielzahl von Systemen sind hierzu entwickelt worden, die auf eine Automatisierung der Vorgänge im Bereich der Entwicklung von Ontologien abzielen. Die Verwendung von unterstützenden 'Tools' wie 'OntoEdit', ermöglicht ein automatisches Lernen, indem mit einem Entwickler kooperativ zusammengearbeitet wird. Ein Benutzer bindet sein Wissen und daraus folgenden Entscheidungen in den Entwicklungsprozess ein.
Bisher entwickelte Systeme unterscheiden sich dabei in einigen grundlegenden Faktoren, haben aber auch zum Teil sehr ähnliche Systemfeatures.

Das 'Ontologie Lernen' versucht ontologisches Wissen aus verschiedenen Ausprägungen der Daten automatisch, beziehungsweise semi-automatisch zu erfassen. Hauptprobleme im Aufbau und in der Verwendung einer Ontologie sind die Überwindung des sogenannte *'Bottleneck of Knowledge Acquisitation'*[2] und die zeitaufwändige Konstruktion, beziehungsweise Integration verschiedenster Ontologien [36]. Lernen im Kontext des 'Ontologielernens' bezieht sich dabei auf das Extrahieren von Ontologischen Elementen (konzeptionelles Wissen) aus den Eingaben und der Erstellung der Ontologie mit Hilfe dieses gefundenen Wissens. Der manuelle Aufbau ist zum Einen mühsam, Kosten und Zeit raubend, zum anderen kann dieser ebenfalls sehr fehleranfällig sein. Eine Automatisierung eliminiert diese Kosten und kann auf Anwendungsebene besser angepasste Lösungen liefern.

Die Verwendung maschineller Lerntechniken kann zunächst eine automatische Entdeckung und Erschaffung von ontologischem Wissen gewährleisten. Wird zusätzlich eine manuelle Komponente integriert, kann im Sinne eines überwachten Lernprozes-

[2] Unterschiedlichste Herausforderungen die sich im Umgang mit der Anreicherung des Wissens ergeben

ses ein erheblich besseres Resultat erzielt werden. Daher sind gerade in der Entwurfsphase eines Systems grundsätzliche Überlegungen zu treffen, um einem geeigneten Lösungsansatz zu finden. Fernziel bleibt dabei die Entwicklung von erfolgversprechenden Methoden, Methodiken, Tools und Algorithmen zur Akquirierung und dem Lernen von Ontologien auf eine bestenfalls (semi-)maschinelle Weise, da ein möglichst gutes Endresultat zu erwarten ist. Häufig kann eine Integration von schon existierenden Ontologien in den Prozess der Wissensgewinnung erfolgen. Diese sogenannten 'Sourceontologien' stehen den unterschiedlichsten Prozessen unterstützend zur Seite.

In diesem Kapitel werden folgende Schwerpunkte behandelt: Im ersten Teil wird eine grundsätzliche Überlegung geliefert, wie ein einfaches System zum Ontologielernen aussehen kann und welche wichtigen Aspekte beim Entwurf bedacht werden müssen. Danach wird ein kurzer Überblick über bestehende Lernansätze, die unter anderem kategorisiert werden können, gegeben. Um den aktuellen Stand der Forschung des Bereichs 'Ontologielernen' aufzeigen zu können, werden klassische Verfahren vorgestellt. Im Anschhluss erfolgt die Angabe einer generischen Architektur zum Lernen und zur Konstruktion einer Ontologie. Der letzte Teil wird sich mit einer speziellen Anwendung des Bereich Ontologielernen, der Population einer Ontologie, befassen. Hier wird ein grundsätzlicher und allgemeiner Ansatz präsentiert.

3.2 Einfacher Systementwurf zum Ontologielernen

Eine allgemeine Beschreibung zur Herangehensweise für den Entwurf eines geeignetem Lernsystems, kann auf Grundlage der Arbeit von *Shamsfard* und *Barforoush* [36] gegeben werden. Es sollen die wichtigsten und grundsätzlichen Aspekte, die beim Entwurf eines einfachen Systems im Mittelpunkt stehen, aufgezeigt werden.

Lernelemente Zunächst sollte festgehalten werden, welche Elemente im Zuge des Ontologielernens akquiriert werden können. So werden konzeptionelle Strukturen unterschiedlichster Art gelernt. Die gelernten Elemente werden unterschieden in ontologische Wissenselemente und einer Kombination aus lexikalischem und ontologischem Wissen. Die wichtigsten lexikalischen Elemente sind Worte. Wichtige Ontologische

Elemente sind hauptsächlich Konzepte, Relationen und Axiome. So kann eine Aufstellung der zu erlernenden Elemente vorgenommen werden. Diese sind:

- Worte
- Konzepte
- Instanzen
- Konzeptionelle Relationen
- Axiome.

Eine Erklärung der prinzipiellen Entitäten einer Ontologie, wie Konzepte, Instanzen und Relationen kann dem Grundlagenkapitel 2 zu den Ontologien entnommen werden. Zu erwähnen ist, dass eine genauere Unterscheidung der konzeptionellen Relationen erfolgen kann. So gibt es 'taxonomische Relationen', die das ontologische Wissen durch verallgemeinernde oder spezifizierende Beziehungen organisieren. Die 'nicht-taxonomischen Relationen' verweisen auf Relationen zwischen den Konzepten, diese werden konzeptionelle Beziehungen genannt.

Eingabedaten des Systems Ein Ontologielernsystem extrahiert entsprechendes Wissen, welches für die Ontologie von Interesse sein kann, aus seiner Eingabe. Dieser Input des Systems unterscheidet sich primär in den Quellen, aus denen Informationen bezogen werden. Die Ressourcen sind demnach unterschiedlichsten Typs und Herkunft, sowie in ihrer Sprache verschieden. Die Typen des Inputs aus denen ontologisches Wissen gewonnen werden, sind folgende:

Strukturierte Daten Datenbank-Schemata, bereits existierende Ontologien, Wissensbasen und lexikalisch semantische Netze (WordNet)

Semi-strukturierte Daten Wörterbücher, HTML- / XML-Dokumente und Dokument-Typ-Definitionen

Unstrukturierte Daten Natürlichsprachliche Texte und Web-Texte

Die Eingaben, welche den unstrukturierten bzw. semi-strukturierten Typen entstammen, bestehen aus Text, der in den natürlichen Sprachen Englisch, Deutsch und anderen verfasst wurde. Die Art der Eingabe ist stark abhängig von der Anwendung die umgesetzt werden soll.

Verwendung von Hintergrundwissen Effiziente Systeme greifen auf sogenanntes Hintergrundwissen (engl.: background knowledge) zurück, um neue Wissenselemente zu akquirieren bzw. existierende aus der Eingabe upzudaten. Hintergrundwissen kann in den Prozess der Wissensgenerierung unterstützend eingebunden werden, um mit dem vorhandenem Wissen neue Lernelemente zu generieren.
So kann dieses Wissen ebenfalls aus unterschiedlichsten Typen von Ressourcen entnommen werden und in Art und Umfang variieren. Die Bezugsquellen sind somit zum einen linguistischer Art, wie z. B. bei Wörterbüchern und zum anderen ontologischer Art, in Form einer bestehenden Ontologie.
Eine Verfügbarkeit und Notwendigkeit eines Einsatzes dieses Hintergrundwissens, welches zur Unterstützung der angewandten Lernmethoden verwendet wird, muss folglich abgeschätzt werden. Es gibt Umgebungen, in denen spezielles Hintergrundwissen, wie zum Beispiel Basisontologien oder semantische Lexika nicht vorhanden sind. Der Bedarf und Typ des Hintergrundwissens, wird durch die Art der beabsichtigten Ontologie, und den Methoden, die angewandt werden, bestimmt. Zur Konstruktion allgemeinerer Ontologien wird eine andere Art von Hintergrundwissen verlangt, als bei der Erstellung einer sehr domänenspezifischen Ontologie.

Ausgabedaten des Systems Bei Betrachtung der Ausgabe, die der Lernprozess des Systems liefert, muss eine grundsätzliche Überlegung bezüglich der Repräsentationsform des resultierenden Wissens stattfinden. Das Resultat des Lernprozesses kann in unterschiedlichsten Formen vorliegen. Ein gewünschtes Resultat kann sein:

- eine aus dem System entstehende Ontologie,
- eine textuelle Wissensbasis oder
- eine beliebig weitere Datenstrukturen bzw. Repräsentationen.

Das ontologische Wissen des Resultats kann in unterschiedlichster Form repräsentiert werden. So gibt es auch Repräsentationssprachen unterschiedlichsten Typs. Klassische Sprachen sind z. B. logikbasiert Sprachen wie 'KIF', Beschreibungslogiken, 'KL-ONE', Framebasierte Sprachen wie 'OKBC', Webbasierte Sprachen wie 'XML' und hybride Sprachen wie 'RDF'. Eine genauere Betrachtung der Repräsentationssprachen kann unter [36] erfolgen. Die Sprache 'OWL' wurde in Kapitel 2 vorgestellt.
Eine resultierende Ontologie kann bestimmte ontologische Strukturen und Topologien besitzen. Strukturen sind beispielsweise strikte Hierarchien der Konzepte, pyramidale Hierarchien, sowie gerichtete Graphen und Verknüpfungen von Hierarchien und Axiomen. Eine Ausgabe kann außerdem in einer gewissen Topologie, wie beispielsweise Graphen und Axiome oder als einfache hierarchische Bäume, angegeben werden.
Sofern das Resultat eine bestimme Repräsentationssprache ist, kann diese auch zur Weiterverarbeitung für weitere Prozesse bereitgestellt werden.

Anwendungszweck Beim Entwurf muss beachtet werden, inwiefern das System eingesetzt werden soll. So gibt es Systeme, welche eine Ontologie lernen (die Struktur der Ontologie) und Systeme, die den Benutzer, Experten oder andere Systeme beim Aufbau unterstützen. Damit sind erstere Systeme autonome Lernsysteme, und weitere Systeme sind Module, die benutzt werden, um die Ontologie aufzubauen.
Daher kommt es darauf an, in welcher potentiellen Anwendung die Ontologie benutzt werden kann und für welche spezifische Domäne sie entworfen wurde. Die häufigsten Anwendungen sind dabei nach wie vor 'Informationsextraktion' und 'Retrieval'.

So kann ein Grad an Automatisierung bei der Wissensgewinnung festgelegt werden. Dieser verläuft von manuell bis komplett automatisch. Beim manuellen Lernen pflegt der User seine Daten praktisch von Hand ein, d. h., er ist vollständig für den Lernprozess verantwortlich.
In Lernprozessen der automatisierten Lernsysteme sind verschiedene Grade von Automatisierung zu benennen. Es gibt voll-automatische Systeme, halb-automatische Systeme und kooperative Systeme. In diesen Systemen übernehmen die Benutzer unterschiedlichste Rollen. Der Benutzer kann zum Beispiel eine initiale Ontologie

vorschlagen und bestimmte Validierungen beziehungsweise Veränderungen die vom System vorgeschlagen werden, akzeptieren. So werden an bestimmten Stellen die Entscheidungen des System bestätigt oder nach dem Ermessen des Benutzers verändert. Ein wesentlicher Beitrag zum Erfolg der kooperativen Systeme ist dabei die Tatsache, dass der User mit seinen Entscheidungen während des Lernprozesses, maßgeblich sein "Know-How" einbringt und damit Hintergrundwissen impliziert, welches durch maschinelle Berechnung in dieser Art vorerst nicht erreicht werden kann.

Preprocessing Eine ebenfalls wichtige Komponente eines Systems kann das sogenannte 'Preprocessing'[3] sein. In einem entsprechenden Datenvorverarbeitungsschritt wird die Möglichkeit angewandt, eine Vorverarbeitung der Eingabe vorzunehmen, um diese in eine geeignete Struktur zu bringen, aus der möglichst gut gelernt werden kann. Dabei ist das wohl bekannteste Verfahren das linguistische Preprocessing, mit seinen Textverarbeitungstechniken 'Tokenizing'[4], 'Part-of-Speech Tagging' (POS)[5] und 'Syntaktische Analyse', die essentielle Strukturen aus den Eingabetexten vorbereiten können.

Geeignete Lernmethode Zum eigentlichen Entwurf eines geeigneten Lernprozesses, wird zusätzlich ein Lernansatz gewählt, welcher dem System eine bestmögliche Verarbeitung der Daten liefert. Ein Lernverfahren kann prinzipiell mit zwei Methoden umgesetzt werden. Zum Einsatz kommen statistische und symbolische Verfahren, die auf der Eingabe arbeiten. Beide Ansätze haben unterschiedliche Eigenschaften wenn sie in Systemen des Ontologielernens eingesetzt werden.
So sollte eine Lernmethode ausgewählt werden, die mit ihrer Methodik auf bestmögliche Weise ontologiespezifisches Wissen gewinnen kann. Die Methoden zur Extraktion des Wissens, können von wissensarmen Ansätzen, wie statistisch basierten Techniken, zu wissensintensiven Ansätzen, wie logikbasierten Techniken, reichen. Diese Ansätze sind entweder überwacht (engl.: supervised) oder unüberwacht (engl.: unsupervised) und unterscheiden sich im Umgang mit Trainingsdaten.

[3] Datenvorverarbeitung
[4] Parser zur lexikalischen Analyse
[5] zur Generierung von Wortarten

Die überwachten Methoden benutzen bereits manuell in Klassen eingeteilte Trainingsdaten. Eine Klassifizierung kann durch Annotation der Daten aus dem Korpus festgehalten werden. Sogenannte unsupervised Algorithmen benötigen keine vorher klassifizierten Daten.

3.3 Lernansätze des Ontologielernens

An dieser Stelle des Kapitels wird etwas genauer auf die zuvor aufgeführten Lernansätze eingegangen. Grundlage ist ein Kapitel aus [36]. Die Lernansätze können von statistischer oder symbolischer Art sein. Ein dritter Ansatz beschreibt die 'Hybriden Techniken', welche mindestens zwei der zuvor genannten Ansätze vereinen, und dabei versuchen, Vorteile der jeweiligen Techniken zu verwenden und Einschränkungen zu eliminieren.
Innerhalb der symbolischen Techniken kann zusätzlich eine Kategorisierung vorgenommen werden. Diese sieht wie folgt aus: logische, linguistisch basierte und Template gesteuerte Ansätze.
Zudem können heuristische Methoden eingesetzt werden, um die jeweiligen Techniken zu unterstützen.
In den statistischen Ansätzen wird eine statistische Analyse der erfassten Daten aus dem Input durchgeführt. Damit kann z. B. eine Erfassung von selektionalen Präferenzen erreicht werden und eine Lokalisierung der Konzepte in der Ontologie auf einem generalisiertem Level vorgenommen werden.
Andere Ansätze verwenden die statistische Analyse von kookkurrenten[6] (engl.: co-occurrence) Daten, um konzeptionellen Relationen aus dem Text zu lernen, oder das Prinzip der Hidden-Markov Modelle (HMM) auf die explizit im nächsten Kapitel der Arbeit eingegangen wird. Auch können z. B. statistische Indizes angewandt werden, um Assoziationsregeln einzuordnen, die geeignet sind komplexe semantische Relationen zu reflektieren.

[6] Gemeinsames Auftreten sprachlicher Einheiten in einem Satz

Statistische Verfahren Ein wichtiger Aspekt bei der Anwendung von statistischen Verfahren ist die Wahl der Abstraktionsebene der untersuchten statistischen Elemente. So können die statistischen Verfahren zum einen auf isolierten Wörtern, und zum anderen auf sogenannten 'Batches of Words' basieren.
Modelle bezüglich isolierter Wörter ignorieren die Sequenz, in denen die Wörter vorkommen. Dies ist begründet in der Annahme, dass das Vorkommen jedes einzelnen Wortes in einem Dokument bedingt unabhängig von allen anderen Worten in dem Dokument, angesichts seiner Klasse, ist. Dieser Ansatz wird 'Naiv Bayes' genannt. [31]
Die Hauptidee hinter der statistischen Methode bestehend aus Wortstapeln (engl.: batches of words) ist, dass die semantische Identität eines Wortes durch seine Verteilung über unterschiedliche Kontexte reflektiert wird. Somit wird die Bedeutung eines Wortes durch Terme von Wörter mit denen es zusammen auftritt und der Häufigkeit dieser Kookkurrenzen repräsentiert. Die Kookkurrenz von zwei oder mehreren Worten in einer wohldefinierten Einheit von Informationen (Sätzen, Dokumenten) wird 'Collocation'[7] genannt. Das Lernen durch Kollocation und Kookkurrenz ist am verbreitetsten innerhalb der statistischen Verfahren.

Symbolische Verfahren Bei der Erklärung der Prinzipien von symbolischen Ansätzen können zunächst logische Methoden wie 'Induktive logische Programmierung', 'First-Order-Logic (FOL) Clustering und Rule-Learning' sowie 'Propositional Learning' genannt werden.[36] All diese Methoden entdecken neues Wissen durch Deduktion oder Induktion und repräsentieren Wissen durch Aussagen (engl.: proposition), Logik erster oder höherer Ordnung. Deduktions-basierte Lernsysteme verwenden logische Deduktion und Inferenzregeln, wie Resolution, um neues Wissen von Existierendem abzuleiten.
Die zweite Klasse symbolischer Ansätze, nämlich die der induktions-basierten Lernsysteme, erzeugt Hyothesen aus Beobachtungen (Beispielen) und stellt neues Wissen künstlich aus Erfahrungen her.
Auf Symbolische Verfahren die auf syntaktischer Analyse basieren, wird im Rahmen dieser Arbeit nicht genauer eingegangen.

[7] Linguistik: Das gehäufte gemeinsame Auftreten von Wörtern

Weitere Methoden der symbolischen Verarbeitung sind die Pattern-basierten beziehungsweise Template-gesteuerten Methoden. Diese Ansätze unterstützen das Matching von Schlüsselworten, Pattern und Templates. Bei den Template-gesteuerten Methoden wird der Input, in aller Regel ein Text nach vordefinierten Schlüsselworten, Templates oder Pattern die Relationen, wie Hyponymien[8], bezeichnen, durchforstet. Es gibt verschiedene Typen von Templates, um die unterschiedlichen Ontologieelemente zu extrahieren. Diese sind syntaktischer oder semantischer Art und für allgemeine oder spezielle Zwecke, zu gebrauchen.
Das wohl geläufigste Patternmatching Verfahren ist das von 'Hearst' [20], bei dem lexiko-syntaktische Muster in Form von regulären Ausdrücken zur Extraktion von hyponymen und hperonymen Relationen angewandt werden.

Multi-Strategie-Verfahren Zuletzt gibt es die Kategorie der Multi-Strategie-Lernsysteme. Da die meisten Systeme nicht nur einen Typ, sondern eine Vielzahl von Ontologieelementen lernen, werden kombinierte Ansätze verwendet. Um verschieden Komponenten der Ontologie zu lernen, wird Multi-Strategie-Lernen angewandt, welche verschiedene Lernalgorithmen einsetzt.
Neben der Kategorisierung der Ansätze in symbolische und statistische Prinzipien, kann auch eine allgemeine Einteilung vorgenommen werden.
Hier wird betrachtet, welche Lernaufgaben die diversen Methoden bewerkstelligen sollen, das heißt, welche Aufgabe (engl.: task) letztendlich ausgeführt wird. Die Lernmethoden können daher, basierend auf den Aufgaben welche von ihnen durchgeführt werden, kategorisiert werden.
Nach Vorwegnahme einer allgemeinen, d. h. theoretischen Beschreibung, werden die Lernaufgaben benutzt, um Wissen aus dem Input zu extrahieren oder die Ontologie zu verfeinern (engl.: refine). In der Kategorie der Lerntasks gibt es also: 'Klassifikation', 'Clustering', 'Regellernen', 'Formale Konzept Analyse' und 'Ontologiepopulation', auf welches in diesem Kapitel noch explizit eingegangen wird.
Die zuvor beschriebenen Lernansätze können innerhalb der eben angegebenen Kategorisierung bezüglich ihrer Lernaufgabe verschiedene Funktionen übernehmen.

[8] Begriff aus der Linguistik: Unterbegriff eines Begriffs

3.3.1 Bewertung der Ansätze

Nach primärer Beschreibung der einzelnen Verfahren, kann nun eine Bewertung und Einschätzung der Güte der jeweiligen Ansätze vorgenommen werden.
Ein Hauptunterscheidungskriterium der Lernansätze lässt sich folgendermaßen beschreiben: Statistische Methoden werden in aller Regel als "blind" oder auch "arm an Wissen" bezeichnet, während die symbolischen Methoden als reich an Wissen angesehen werden können.
In Systemen, in denen keine Analysen semantischer Art und 'Reasoning Prozeduren' vorgenommen werden, wie zum Beispiel in 'Information Retrieval Systemen', die Anfragen syntaktisch bearbeiten, sind statistische Methoden gut anwendbar. Die statistischen Verfahren sind im Allgemeinen berechenbarer, allgemeiner gehalten, skalierbarer und in der Praxis einfacher zu implementieren. Der statistische Ansatz ist üblicherweise aufgrund der allgemeineren Anwendbarkeit durchaus für verschiedene Domänen und auch Sprachen einsetzbar. Aufgrund der Tatsache, dass das statistische Verfahren Hintergrundwissen nicht berücksichtigt, werden für die Methode weniger initiale Ressourcen benötigt.
Das nicht vorhandene Wissen in diesem Ansatz zeigt an dieser Stelle einen großen Nachteil der statistischen Methoden auf, da die verwendeten Daten normalerweise relativ "sparse" sind. Gut zu erkennen ist dies beispielsweise bei der Erstellung von allgemeingültigen Konzepten aus sehr fachspezifischen Texten, und in umgekehrter Richtung, bei der Extraktion von fachlichen Konzepten aus sehr allgemeinen oder auch irrelevanten Textdaten.
Statistische Ansätze werden grundsätzlich in offline Ontologie Lernverfahren eingesetzt, die nicht-inkrementell ablaufen und üblicherweise den gesamten Input als Ganzes, z. B. vollständig geparste Texte im gesamten Kontext erlernen.
Die symbolischen Ontologie Lernverfahren hingegen gelten als präziser, robuster und liefern angemessenere Resultate als die Kategorie der statistischen Verfahren. Allerdings benötigen diese Systeme, seien sie denn linguistisch-basierte oder patterngestützte Methoden, eine höheren Anteil an Adaption bei der Umsetzung.[36]

Das Lernprinzip in den symbolischen Verfahren ist inkrementell und kann zudem Reasoning Techniken ausnutzen und semantisches Wissen zur Extraktion neuen Wissens verwenden. Die symbolischen Ansätze gehen bei der Verarbeitung eher in die Tiefe als in die Breite. Dieses Verhalten hat zur Folge, dass mit symbolischen Methoden eine höhere Genauigkeit erzielt werden kann als bei den statistischen Verfahren.
Innerhalb der symbolischen Ansätze sind die logik-basierten Methoden die wissensintensivsten und sind damit geeignet für Systeme, die auf tiefes Verständnis und Schlussfolgern (engl.: reasoning) abzielen.
Die Kategorie der symbolischen, am meisten verbreiteten Methoden, sind die Pattern- und Template-gestützten Ansätze, welche ein Vorwissen voraussetzen und auf natürlichsprachlicher Verarbeitung basieren. Diese haben eine äußerst gute Performanz in bestimmten Domänen, allerdings ist die Adaption auf neue Domänen und die Extraktion neuer Muster innerhalb der Domäne extrem aufwendig.

Nachdem bisher ein Überblick über den aktuellen Stand der Lernansätze gegeben werden konnte, wird in diesem Teil der Arbeit nochmals explizit auf ein spezielles System eingegangen. Wie zuvor angedeutet, gibt es die unterstützenden Tools um Ontologien zu lernen und autonomen Systeme des Ontologielernens.

3.4 Rapid Ontology Development

Um einen kurzen Einblick in den aktuellen Stand der Forschung gewähren zu können, soll ein aktuelles Lernsystem vorgestellt werden.
Das Modell des 'Rapid Ontology Development', kurz ROD, von *Zhou* [49] gibt einen Einblick, wie ein ontologisches Lernsystem in seinem Kern aussehen kann.
Eine Ontologie kann als eine Konzeptualisierung einer Domäne angesehen werden. Diese ist damit eine semantische Grundlage von maschinenverständlichen digitalen Inhalten. Der Inhalt kann zwischen den verschiedenen Anwendungen vollständig kompatibel verwendet werden. Unter dieser Annahme kann ein Modell angegeben werden.
Ontologien stellen Wissen aus einer speziellen Domäne zur Verfügung, welches sowohl vom Anwender, als auch vom Computer verwendet werden kann, darüber hinaus lie-

fern sie Domänenkonzepte und Relationen zwischen den Konzepten und definieren Eigenschaften, Funktionen, Beschränkungen und Axiome.
Ein lernorientiertes Modell der Ontologieentwicklung umfasst folgenden Aspekte: [49]

- die Ontologie Repräsentation muss für Computer und Mensch verständlich und zu verarbeiten sein,
- eine verwendete Repräsentationssprache muss Adäquanz in der Repräsenation und Effizienz für Inferenz, vorweisen
- eine standardisierte Ontologie Repräsentationssprache, z. B. OWL (siehe Unterabschnitt 2.2.3).

Ein wichtiger Punkt ist die Erschaffung von Inhalten der Ontologien, wie Konzepte und Relationen, die in der 'Ontologie-Akquirierung' erfolgt. Domänenwissen kann durch den Anwender als Expertenwissen eingebracht werden, aber auch aus anderen Quellen, wie Wörterbücher, Webdokumente oder Datenbankschemata einfließen.
Ein ebenfalls wichtiges Thema ist die Bewertung der Ontologie, welche die Qualität der Ontologie verbessern kann, Kompabilität zwischen den Systemen aufrecht erhalten kann, sowie die Adaptionsfähigkeit der Ontologien erhöhen kann. Überprüft wird vornehmlich die Vollständigkeit, Konsistenz und Korrektheit einer Ontologie.
Als letzten Aspekt gibt es die 'Ontologie Verwaltung', in der es darum geht, die existierende Ontologie zu organisieren, zu suchen und auf dem neuesten Stand zu halten. Dabei ist eine konstante Entwicklung der Umgebung der Ontologie zu erreichen, um den Veränderungen flexibel nachkommen zu können. Dies ist in der Regel mit automatisierten Lösungen umsetzbar.
Mit diesen Annahmen kann ein lernorientiertes Modell zur Ontologieentwicklung erstellt werden.
Wie in der Abbildung 3.2 zu sehen ist, besteht das 'Rapid Ontologie Development' aus drei Phasen: der Designphase, der Lernphase und der Validierungsphase.
Die Designphase enthält die Identifikation und eine detaillierte Analyse der Domäne, Anforderungen und relevanten Ressourcen mit Hilfe eines Benutzers oder Domänenexpertens. Als Erkenntnis beziehungsweise Ausgabe dieser Phase, wird eine Spezifikation der Domäne, vorgesehene Anwendungsformen der Ontologie und die

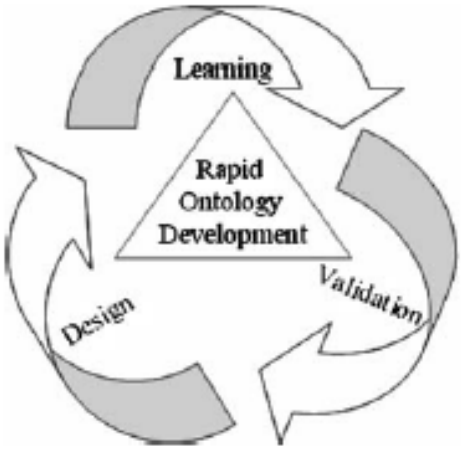


Abbildung 3.2: Rapid Ontology Development[49]

maßgeblichen Domänenquellen geliefert. Die Quellen einer Domäne liegen in vielerlei Formaten vor. Sie können strukturierten Typs sein, wie bei Thesaurus[9] und Glossar[10], semi-strukturierten Typs, wie bei Bücherindizes und Webseiten, oder in Form von Expertenwissen. Ein guter Ansatz kann gefunden werden, wenn die Anforderungen des Benutzers und die zum Ziel gesetzten Anwendungen miteinbezogen werden. In der Lernphase werden geeignete Lerntechniken ausgewählt, implementiert und schließlich angewandt, um Ontologien aus den Domänenquellen zu finden.
Die Lernergebnisse werden im Lauf der Validierungsphase ausgewertet und präzisiert. Dabei werden die aufgefundenen Ontologien auf Redundanz, Konflikt und auf fehlende Informationen untersucht. In dieser Phase ist der Eingriff des Domänenexperten ausdrücklich erwünscht, da Probleme identifiziert, Gründe analysiert und Lösungen erstellt und angewandt werden.
'Rapid Ontology Development' beschreibt einen inkrementellen Prozess, das bedeutet, der beschriebene Zyklus wiederholt sich solange, bis das Ergebnis für Anwender und Wissensentwickler annehmbar ist.
Damit ermöglicht ROD das Management von ontologiebasierten Businessprozessen und die dynamische Integration von Anwendungen. Es liefert zudem wiederverwendbare Ontologien, welche in neue Applikationen eingebunden werden können.[49]

[9] http://thesaurus.com/
[10] http://www.glossar.de/glossar/index.htm

Nachdem eine Grundstruktur mit dem 'Rapid Ontology Development' angegeben werden konnte, kann in einer nächsten Überlegung ein Framework zum Ontologielernen definiert werden. Das Framework beschreibt also die Hauptkomponenten des Prozesses, der dem Ontologielernen zugrunde liegt. In seiner allgemeinsten Form besteht es damit aus der Informationsextraktion, Ermittlung der Ontologie und anschließende Organisation der Ontologie.

- **Informationsextraktion**

  Ontologielernen kann eine Vielzahl von Daten unterschiedlicher Struktur und Ausprägung ausnutzen. Die Informationsextraktion verarbeitet textuelle Dokumente und strukturierte Daten. Eine weitere wichtige Aufgabe ist die Erkennung von Informationen in unterschiedlichster Repräsentationsform, und deren anschließender Konvertierung in eine Form, die vom darauffolgenden Schritt der Ontologieentdeckung benutzt werden kann.

  Dabei werden Textdokumente per Inhaltsanalyse verarbeitet.

- **Ermittlung der Ontologie**

  Aus dem zuvor extrahierten Wissen werden mit überwachten und unüberwachten Lernalgorithmen ontologische Konzepte und Relationen festgestellt. Während des Lernens der Konzepte werden unbrauchbare Wörter und Satzteile, die zum Beispiel nur grammatikalische Funktionen erfüllen, herausgefiltert. Die Kandidaten für Domänenkonzepte sollen ausgegeben werden.

  Das Lernen von Relationen unterliegt der Statistik von Kookkurrenzen, dazu gibt es verschiedene Ansätze, die sich in ihren unterschiedlichen Methoden unterscheiden.

- **Organisation der Ontologie**

  Die aus dem Lernprozess extrahierten möglichen ontologischen Konzepte und Relationen werden auf Verwendbarkeit geprüft, dies kann durch folgende Schritte gewährleistet werden:

  - Clustering von synonymen Termen und deren Relationen um Redundanzen aufzuheben

- Ableiten von inversen Relationen, die schon zuvor entdeckt wurden
- Auffindung lokaler Zentren von Konzepten, sogenannten Konzeptinseln aus Gruppen von Konzepten, die nah zusammenhängend, untereinander kaum verbunden mit Konzepten außerhalb der Gruppe, sind, da Konzepte verschieden stark in Relation zueinander stehen
- Aufbau von top-level Ontologien, wobei die lokalen Zentren als höherstufige (top-level) Ontologien in der Zieldomäne dienen

All die Komponenten des beschriebenen Frameworks zum Ontologielernen sollen möglichst automatisiert verwendet werden, einige Komponenten müssen manuell ausgeführt werden.

Kandidatenmodelle Als nächstes kann auf den Vorgang, der dem allgemeinen Ontologielernen als Basis dient, eingegangen werden. Eine Ontologielerntechnik fängt für gewöhnlich bei der Formalisierung der Daten mit einer Menge von Features, die ontologisches Wissen ausweisen, an. Danach werden interne Repräsentationen von Modellen geeigneter Kandidaten aus dem formalen Input der Daten generiert. Zuletzt wird das beste Modell unter verschiedenen Kriterien ausgewählt und auf die jeweilige Probleminstanz angewandt, um an ontologisches Wissen zu gelangen.

Lerneinheiten Beim Ontologielernen gibt es unterschiedliche Lerneinheiten. Es können dabei Einheiten, die Wörter oder ganze Terme sind, gelernt werden. Eine Einheit ist also ein einzelnes Wort oder ein zusammengesetzter Ausdruck.
Sogenannte 'Learningtargets', also Elemente, die aus dem Lernprozess zu extrahieren sind, können Konzepte, Relationen, Definitionen und Axiome sein.

Datenquellen Eine Sammlung von Dokumenten, Wörterbüchern, das Internet und Benutzerinteraktionen. Sie beschreiben die Datenquellen auf die zurückgegriffen wird und die dem Lernprozess Informationen liefern.

Lernstrategien und Lernverfahren Bekannte Entwurfsprinzipien bilden die Dimension der Lernstrategien. Es gibt daher Bottom-Up-, Top-Down- und Hybride-

Strategien. Die Bottom-Up-Strategie beginnt mit Textdokumenten und leitet schrittweise Top-Level-Ontologien ab, wohingegen beim Top-Down das Prinzip genau gegensätzlich ist.
Eine wichtige Rolle kommt den eingesetzten Lerntechniken zu, welche statistisch-basierter, regel-basierter oder hybrider Art sind.

Statistische Modelle Ein statistisches Modell wird in aller Regel als ein probabilistisches Netzwerk repräsentiert, welches probabilistische Abhängigkeiten zwischen zufälligen Variablen wiedergibt. So werden statistische Informationen durch beobachtete Häufigkeiten oder gemeinsame Verteilung[11] der Terme berechnet, um Konzepte und deren Relationen zu bestimmen. Dabei variieren die Ansätze, wie das probabilistische Netzwerk generiert wird und welche Methoden angewandt werden um individuelle Verteilungen zu kombinieren; die Bekannteste ist hier 'Maximum-Likelihood'.

Regelbasierte Modelle Bei regelbasierten Ansätzen wird Matching von vordefinierten Regeln oder heuristischen Mustern eingesetzt, um Terme und Relationen zu gewinnen. Modelle innerhalb dieses Ansatzes werden prinzipiell als Menge von Regeln repräsentiert, die bestimmte Bedingungen testen und Aktionen ausführen.

Da in den meisten Fällen annotierte Trainingsdaten nicht vorhanden sind, sind die meisten Lerntechniken unüberwacht.
Für manche Probleme kann das 'Assoziationslernen' angewandt werden. Dabei werden Beziehungen zwischen Objekten untersucht, die anhand von Ähnlichkeitsmaßen abgeschätzt werden können. Objekte sind dabei einmal ähnlich aufgrund ihres gemeinsamen inhaltlichen Auftretens in bestimmten Kontexten. Diese werden als extrinsische Features bezeichnet. Zum anderen können Objekte gleiche Eigenschaften und Beziehungen haben, in diesem Fall liegen intrinsische Features vor.
Einige Verfahren sind stärker abhängig von externem Wissen als andere, eine Unterstützung durch Wissen ist daher entweder "wissensreich" oder "wissensarm".

[11] auch multivariante Verteilung: Wahrscheinlichkeitsverteilung einer mehrdimensionalen Zufallsvariable

Nachdem eine allgemeine Prozessstruktur zum Ontologielernen erarbeitet werden konnte, wird im Nachfolgenden eine generische Struktur für unterstützende Entwicklungstools angegeben.

3.5 Generische Architektur

Dieser Abschnitt liefert eine generische Architektur zum Lernen von ontologischen Entitäten und bezieht sich inhaltlich auf [29].
Die Ontologien liefern eine formale Konzeptualisierung einer bestimmten Domäne des Interesses und stellen diesen Wissensbereich für eine ganze Gruppe von Leuten zur Verfügung.
Die Idee hinter dem Ontologielernen unterliegt dem sogenannten 'Balanced Cooperative Modeling Paradigma' [30], welches eine koordinierte Interaktion zwischen menschlichen Entwickler und Lernalgorithmus zur Konstruktion von Ontologien beschreibt. Der Ansatz kombiniert die Entwicklung der Ontologien mit dem maschinellen Lernen, wie es in den kooperativen Systemen zum Lernen und zur Konstruktion von Ontologien eingesetzt wird.
In der Thematik des Ontologielernens geht es um Wissensbeschaffung aus unterschiedlich beschaffenen Daten und deren anschließender Repräsentation durch eine ontologische Struktur. Zur Verdeutlichung wurde eine generische Architektur für das Lernen aus Ontologien aufgestellt. Diese besteht aus vier relevanten Hauptkomponenten. Nach dem Ansatz von *Maedche und Staab* [30] sind es folgende Komponente:

- **Ontologie Management Component**
  Der Anwender kann hier manuell in Ontologie eingreifen, erlaubt wird:
  - Integration bestehender Ontologien
  - Durchforsten, Validierung, Modifikation, Versionierung und das Weiterentwickeln von Ontologien
- **Ressource Processing Component**
  Beinhaltet eine Vielzahl von Techniken um relevante Eingbedaten zu finden,

importieren, analysieren und transformieren. Hier soll eine Menge von vorverarbeiteten Daten generiert werden, die als Input für die weiteren Komponenten dienen.
Wichtige Unterkomponente ist Verarbeitungssystem für natürliche Sprache.

- **Algorithm Library Component**
  Eine Bibliothek, bestehend aus einer Menge unterschiedlicher Algorithmen, welche die Aufgabe der Extraktion und Verwaltung von ontologischen Teilen des Ontologiemodels übernehmen. Ziel ist, einen standardisierten Output mit einer allgemeinen Ergebnisstruktur aller Algorithmen zu finden.

- **Coordination Component**
  Hier kann der Anwender mit dem System interagieren durch:
  - Einfluss auf Verarbeitung der Ressourcen und Algorithmenlibrary
  - Angebot von umfassenden User-Interfaces zur Unterstützung der Selektion von relevanten Daten
  - Anwendung von Verarbeitungs- und Transformationsalgorithmen
  - Start von bestimmten Extraktionsmechanismen

Bei der Durchführung des Ontologielernens kann zwischen zwei fundamentalen Aspekten unterschieden werden. Zum einen gibt es die Möglichkeit der Verwendung einer früheren Wissensbasis. Hier muss der Lernprozess nicht von Grund auf durchgeführt werden, da von einer Basis bestehenden Wissens begonnen wird. Dies ermöglicht, eine erste schnelle Version der Ontologie mit geringen Aufwand zu erstellen, und diese dann durch einen automatischen Lernprozess oder durch manuelle Betreuung zu erweitern.
Zum anderen gibt es die Vielzahl unterschiedlich strukturierter Inputdaten, die ebenfalls auf jeweils eine bestimmte Weise verarbeitet werden.
Unter Berücksichtigung der zuvor definierten generischen Architektur kann im weiteren Verlauf ein spezielleres Framework beschrieben werden, welches einem zu entwickelndem System eine Grundlage in seinem Arbeitsprozess liefert. Es beschreibt

eine konzeptionelle Konkretisierung des 'Rapid Ontologie Development' . In dem Paper von *Maedche und Staab* bezüglich 'Ontologie Learning for the Semantik Web' wird eine Architektur angegeben, die Wissensextraktion mit maschinellen Lernverfahren kombiniert, um eine Anwendung für das Semantik Web zu entwerfen.
Die Module in dem angegebenen Framework durchlaufen verschiedene Schritte im sogenannten 'Engineering Circle'.

1. **Import& Reuse**

   Existierende Ontologien werden importiert und wiederverwendet durch Vereinigung existierender Strukturen oder unter Definition von Abbildungsregeln zwischen existierende Strukturen und der zu erstellenden Ontologie.

2. **Extract**

   In der Extraktionsphase werden Hauptelemente der Zielontologie (engl.: target) modelliert mit Lernsupport der verwendeten Ressourcen.

3. **Prune**

   Der gewonnene Entwurf der Zielontologie wird zurechtgestutzt, um an den eigentlichen Zweck angepasst zu werden.

4. **Refine**

   Eine Verfeinerung wird durchgeführt, die von der gegebenen Domänenontologie profitiert und eine hohe Granularität aufweist.

5. **Validation**

   Validation der resultierenden Ontologie.

Der Entwurfszyklus ist in Abbildung 3.3 graphisch dargestellt.

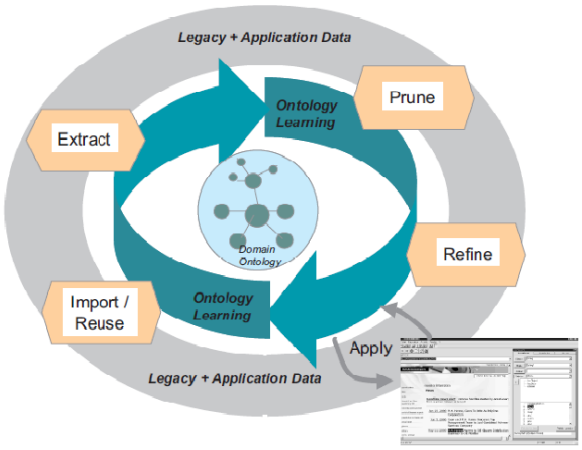


Abbildung 3.3: The Engineering Circle[30]

Dieser Zyklus kann erneut durchlaufen werden, um neue Domänen in die konstruierte Domäne einzubinden oder um den aktuellen Bereich aufrechtzuerhalten und upzudaten.

Einige Grundlagen in der Anwendung des Ontologielernens konnten vorgestellt werden.

3.6 Ein allgemeiner Ontologie Lernalgorithmus

Der hier beschriebene Lernalgorithmus von *Staab* [37] aus der Arbeit *'Ontology Learning'* wird als Anschauung für eine relativ einfache Implementierung eines Algorithmus zum Lernen von Entitäten einer Ontologie sein. Der Algorithmus besteht aus einer mehrteiligen Implementierung, welche die verschiedenen Teile einer Ontologiedefinition abdeckt und damit wichtige Komponenten des Ontologielernens aufgreift. Wichtige Komponenten sind Konzept- und Relationsextraktion der Ontologie. Diese Komponenten sind äquivalent zu den einzelnen Phasen im Prozess des behandelten 'Engineering-Circle'. Jeder Teil der Ontologie kann voneinander isoliert ausgewertet werden, so dass beispielsweise Konzepte und Relationen jeweils für sich extrahiert werden können.

3.6.1 Lexikalische Einträge und Konzept Extraktion

Die einfachste Technik, die eine recht intuitive Idee zur Extraktion relevanter lexikalischer Einträge, die auf ein Konzept hindeuten, umsetzt, ist das Zählen der Vorkommen von Termen in einer gegebenen Menge von linguistisch vorverarbeiteten Dokumente, dem Korpus $\mathcal{D}$. Es ist das sogenannte Prinzip der Termfrequenzen [12] (engl.: termfrequency).
Der Ansatz hinter der verwendeten Annahme ist, dass das häufige Vorkommen eines Terms in einer Menge von domänenspezifischen Texten auf das Auftreten eines relevanten Konzeptes hinweist. Das einfache Zählen des Vorkommens eines Terms ist der standardisierte 'Information Retrieval' Ansatz, der auf diese Art Terme wichtet. Der vorgestellte Algorithmus basiert auf folgenden Definitionen: [29]

- Die **Lexikalische Eintragshäufigkeit** $\boldsymbol{lef}_{l,d}$ [13] ist die Häufigkeit des Auftretens eines lexikalischen Eintrages $l \in \mathcal{L}$ in einem Dokument $d \in \mathcal{D}$
- Die **Dokument Häufigkeit** df_l ist die Anzahl der Dokumente in dem Korpus $\mathcal{D}$ in denen l vorkommt
- Die **Korpushäufigkeit** cf_l ist die totale Anzahl der Auftreten von l in den gesamten Korpus $\mathcal{D}$

Anzumerken ist, dass $df_l \leq cf_f$ und stets gilt $\sum_d lef_{l,d} \neq cf_l$. Somit wird die Relevanz eines Terms gemessen anhand des Information-Retrieval-Maßes **tfidf** der **Termhäufigkeit invertierter Dokumenthäufigkeit**.

***Definition**: Sei $lef_{d,l}$ also die Termfrequenz eines lexikalischen Eintrags l in einem Dokument d. Sei df_l die gesamte Dokumentenhäufigkeit des lexikalischen Eintrags l. Demnach ist $tfidf_{l,d}$ des lexikalischen Eintrags l für das Dokument d gegeben durch:*
Die Formel: Termfrequenz eines lexikalischen Eintrags

$$tfidf_{l,d} = lef_{l,d} * \log\left(\frac{|D|}{df_l}\right)$$

[12] Begriff aus der Information Retrieval: Steht für die relative Häufigkeit eines Terms oder Wortes in einem gesamten Dokument
[13] engl.: lexical entry frequency

Damit gewichtet 'tfidf' die Termfrequenz eines lexikalischen Eintrages in einem Dokument mit einem Faktor, der seine Wichtigkeit vermindert, wenn der Term in nahezu jedem Dokument vorkommt.
Terme, die zu selten oder zu oft vorkommen, werden niedriger eingestuft als Terme, die eine gewisse Balance erhalten.
Um die Wichtigkeit eines Terms für einen gesamten Korpus einzustufen, werden durch 'tfidf' die Werte für lexikalische Einträge l wie folgt berechnet:
Definition:

$$tfidf_l = \sum_{d \in D} tfidf_{l,d} \ , \ tfidf_l \in \mathbb{R}$$

Der Anwender kann einen **Schwellwert** $k \in \mathbb{R}^+$, den $tfidf_l$ überschreiten muss, definieren, wobei der Schwellwert k variabel sein kann. Der Schwellwert wird verwendet, um Terme im Korpus zu untersuchen.

3.6.2 Extraktion von Taxonomischen Relationen

Eine einfe Variante der Extraktion von taxonomischen Relationen kann in unterschiedlichsten Arten erfolgen. Es können dabei zwei wichtige Ansätze [37] verfolgt werden, die hierfür grundsätzlich eingesetzt werden:

- Statistisch-basierte Extraktion unter Verwendung von Clustering oder Klassifikation
- Lexiko-syntaktische Patternextraktion

Clustering Um Clustering anzuwenden, können distributionale[14] Daten verwendet werden, die Konzepte in Hierarchien einbetten. Die so erhaltene distributionale Repräsentation beschreibt einen Term durch die gewichtete Häufigkeit der Terme, die in einem eingegrenzten Inhalt vorkommen. Die Eingegrenztheit ist definiert durch die sequentielle Nachbarschaft der Terme, welches die Anzahl der Terme beschreibt, die zwischen dem repräsentierten und dem repräsentierendem Term liegen.

Beim Clustering werden in einem Prozess Objekte in Gruppen organisiert, deren

[14] entspricht dem Begriff extensional

Mitglieder in gewisser Weise ähnlich sind. Diese Ähnlichkeit kann unterschiedlichster Definition sein, sie beruht beispielsweise auf einem festgelegten Ähnlichkeitsmaß, oder kann definiert werden durch andere distributionalen Repräsentationen.
Es gibt drei Hauptarten des Clustering, deren Prinzipien hierarchische Beschreibungen der Terme produzieren [37]. Es entstehen Hierarchien von 'Termclustern', die dem Anwender zur Weiterverarbeitung geliefert werden. Im Folgenden werden die Methoden zur Konstruktion kurz beschrieben:

1. Agglomerativ [15]: Zu Anfang in der Phase der Initialisierung bildet jeder Term ein eigenes Cluster, in der nächsten Phase werden iterativ größere Cluster gebildet, indem ähnliche beziehungsweise weniger unähnliche initiale Cluster nach dem Bottom-Up Prinzip vereint werden. Dies wird solange weitergeführt, bis ein Abbruchkriterium erreicht wird.

2. Partitional: Hier ist die Menge aller Terme in der initialen Phase ein Cluster. In der Verfeinerungsphase werden iterativ kleinere Cluster generiert, indem das größte Cluster oder auch am wenigsten homogene Cluster in verschiedene Subcluster geteilt wird.

3. Konzeptual: Es werden die exakten Überdeckungen der repräsentierenden Terme zwischen jeweils zwei repräsentierten Termen untersucht und Cluster gebildet, die konzeptional zueinander passen.

Klassifikation Mit der Klassifikation werden Taxonomien genauer ausgearbeitet, so können neue relevante Terme in einer gegebenen Konzepthierarchie klassifiziert werden unter Verwendung von Quellen wie WordNet, die schon substantielle Hierarchien besitzen. Danach können Klassifier aus einem Trainingskorpus und der Menge vordefinierter Konzepte mit ihren lexikalischen Einträgen gelernt werden. Zusätzlich kann eine Konstruktion von distributionalen Repräsenationen erfolgen, die für relevante, noch nicht klassifizierte Terme stehen. Der gelernte Klassifier gibt einen Knoten vor, zu dem der neue Term eine Unterklasse ist.

[15] auch: anhäufend

Lexiko-syntaktische Pattern Zur Extraktion von bestimmten taxonomischen Beziehungen kann eine lexiko-syntaktische Methode gewählt werden. Der hier angewandte pattern-basierte Ansatz, welcher in aller Regel heuristischer Art ist, scannt den Text der untersucht werden soll nach Instanzen von ausgezeichneten lexiko-syntaktischen Mustern, die eine interessante Relation, zum Beispiel Taxonomie, auszeichnen.

Grundlegende Idee ist es, einen regulären Ausdruck zu definieren, der die wieder auftretenden Ausdrücke festhält und die Resultate des Matchingausdrucks auf eine semantische Struktur abbildet. Diese sind taxonomische Relationen zwischen Konzepten.

Hierfür kann ein anschauliches Beispiel gegeben werden, wie lexiko-syntaktische Pattern eingesetzt werden. Es wird folgendes Muster zur Anwendung auf einen Satz definiert:

$$\ldots \text{NP}\{,\text{NP}\}^*\{,\} \text{ or other NP} \ldots$$

Wobei NP für Nominalpronomen steht, und das Patternmatching damit Substantive verarbeiten kann. Aus diesem Muster ist rückzuschließen, dass Nominalpronomen auf Konzepte hinweisen, die links von *or other* stehen und damit Unterkonzepte von dem Konzept rechts von *or other* sind. Der Beispielsatz *"Prellungen, Wunden, Knochenbrüche oder andere Verletzungen sind bekannt"* liefert damit folgende taxonomische Relationen: (Prellung, Verletzung), (Wunde, Verletzung) und (Knochenbruch, Verletzung).

3.6.3 Extraktion von allgemeinen binären Beziehungen

Für das Entwurfsziel des Ontologielernens kann ein Data-Mining Algorithmus auf syntaktische Strukturen und statistische Kookkurenzen angewandt werden. Der Algorithmus arbeitet auf Assoziationsregeln, um interessante Gemeinschaftsbeziehungen zwischen den Daten zu finden.

Ein Textkorpus stellt Informationen zu einer bestimmten Domäne bereit, indem er Objekte beschreibt. Diesbezüglich kann folgendes Beispiel betrachtet werden:

1. *"Frankfurts" schönstes "Hotel" liegt in Sachsenhausen.*
2. *Alle "Zimmer" sind mit "TV", Telefon, Modem und Minibar ausgestattet.*

Bei Verarbeitung des ersten Satzes kann eine Abhängigkeitsrelation zwischen den Termen extrahiert werden, und so können Paare von lexikalischen Einträgen gebildet werden. Im zweitem Satz wird die Heuristik zum präpositionalen Zusammenfügen von Satzteilen, die einen präpositionalen Ausdruck mit seiner am nächsten liegenden Nominalphrase zusammenfügt, angewendet. Beide Möglichkeiten können damit Konzeptpaare bilden, die mit Hilfe einer bestehenden, als Hintergrundwissen agierenden Konzepthierarchie erstellt werden. Damit kann ein Algorithmus zum Lernen von verallgemeinerten Assoziationsregeln vorhandene Taxonomien einbinden, um eigene allgemeine binäre Konzeptbeziehungen auszubilden.
Andere Algorithmen verwenden Verben als potentielle Indikatoren für allgemeine binäre Beziehungen.

3.7 Ontologiepopulation

Die Methodiken der Population einer Ontologie werden zur automatischen Extraktion von Instanzen, welches Schwerpunkt dieser Arbeit ist, eingesetzt. An dieser Stelle des Kapitels wird daher explizit eine klassische Anwendung der Population vorgestellt.

Gemäß der allgemeinen Definition, ist die Ontologiepopulation eine Anwendung aus der Klasse des Ontologielernens.
In einer Vielzahl von literarischen Arbeiten ist für den Begriff 'Ontologiepopulation' keine allgemeingültige Definition ableitbar. Intuitiv ist die Population von Ontologien am besten zu beschreiben als: "Vorgang des automatischen, beziehungsweise semi-automatischen Befüllens einer bestehenden Ontologie mit Instanzen" [43].
Ein entscheidender formaler Unterschied zum Lernen einer Ontologie ist, dass bei der Population, die eigentliche Definition der Ontologie nicht verändert wird. Beim Lernen werden Konzepte und Beziehungen gefunden, beziehungsweise bestehende verändert, wohingegen bei der Population die Informationen aus dem Text extrahiert und diese als Instanzen von bestehenden Konzepten und Relationen hinzugefügt werden.

3.7.1 Paradigmen

Im Wesentlichen besteht die Population von Ontologien aus zwei Hauptparadigmen. Der erste Ansatz beruht auf der Struktur der Terme und benutzt Pattern, die häufigste Anwendung ist die Verwendung der 'Hearst-Pattern' [20]. Das bedeutet, hier wird das Auftreten von textuellen Mustern (engl.: pattern) untersucht, um bestimmte Beziehungen, im englischen Sprachgebrauch zum Beispiel "is-a-Beziehungen", zwischen Wörtern zu identifizieren. Ein Muster kann außerdem eine "Head-Matching Heuristik" [45] sein, die versucht zu testen, ob ein Term in einem zweiten Term vorkommt, in diesem Fall die vorangestellte Position, also der "Kopf" des zusammengesetzten Terms. Hier wird die Struktur der Terme genauer untersucht.
Beim zweiten Ansatz werden sogenannte 'Kontext-Features' [9] betrachtet. Der Korpus wird benutzt, um Features zu extrahieren, die in Kontext zur semantischen Klasse der gesuchten Instanzen stehen.
Der 'Kontext-Feature Ansatz' wird in der später durchgeführten konkreten Anwendung auf Daten, die durch die Informationsextraktion gewonnen wurden, eingesetzt. Eine Beschreibung der Methode erfolgt daher in Unterabschnitt 11.3.4.

Befüllen durch Instanzen Der Vorgang des Befüllens von Instanzen ist somit gleichbedeutend mit einer allgemeinen Anreicherung des ontologischen Wissens. In dem Paper von *Ciminano* [44] wird die Erhöhung des Wissens durch Ontologiepopulation und Anreicherungsmethoden (engl.: enrichment) beschrieben. Es wird eine inkrementelle Methodik zur Instandhaltung der Ontologie beschrieben, die eben die Möglichkeiten der Ontologiepopulation und Methoden zur Anreicherung ausnutzt, um das Wissen zu erweitern, welches durch Instanzen der Ontologie und deren verschiedener Lexikalisierungen (engl.: lexicalization) festgehalten wird. Einsatz finden hier ebenfalls Techniken des Ontologielernens, um die Intervention des Anwenders so gering wie möglich zu halten.
Der Bedarf an der Population von Ontologien ist anschaulich am Beispiel des World Wide Web zu erkennen, das die zur Zeit wohl reichhaltigste Quelle (engl.: repository) an Informationen darstellt. Gerade bei dem aktuellen Übergang in das künftige Semantik Web, ist der Vorgang der Anreicherung von Metadaten, welche semantisch

annotiert sind, von fundamentaler Bedeutung. Diese semantischen Annotationen liefern bestimmten Entitäten der betrachteten Domäne zusätzliche Informationen, die damit eine semantische Interpretation des Inhalts erleichtern, indem formale Modelle der Interpretation beschränkt werden. Die domänenspezifischen Entitäten werden als Instanzen der Konzepte in der Ontologie repräsentiert. Damit erfasst eine Domänenontologie Wissen in einer statischen Form, und ist damit in einem bestimmten Zeitfenster eine Art Momentaufnahme des Wissens eines bestimmten Bereichs, welcher aus einer bestimmten Perspektive mit Hilfe seiner Konzeptualisierung festgehalten wird. Das Wissen kann formal für Maschine und Mensch organisiert und zentralisiert werden. Eine Ontologie ist der zentrale Punkt in der Wissensdomäne, der wissensintensiven Service leistet.

Ein wesentlicher Aspekt im Umgang mit der Domäne ist die Aufrechterhaltung und Pflege des festgehaltenen Wissens. Diese Anforderung an das System kann durch Population und Anreicherung eingehalten werden. Der damit verbundene Vorgang ist jedoch, wie auch schon das allgemeine Ontologielernen, zeitraubend, arbeitsintensiv und fehleranfällig, so dass wiederum der Einsatz von maschinellen Lernverfahren zur Beschaffung der notwendigen Daten sinnvoll ist. Infolge der Erhaltung einer bestehenden Ontologie, im Sinne von Verwaltung oder Wartung, kann das festgehaltene ontologische Wissen der Domäne angereichert werden. Bei der "Instandhaltung" geht es dabei vorrangig um Instanzen und deren verschiedenen Lexikalisierungen, so dass neue Instanzen für Konzepte einer Domäne identifiziert werden, und diese hinzugefügt werden. Dieser Vorgang entspricht der Methode der Ontologiepopulation. Werden hingegen nicht-taxonomische Beziehungen zwischen Instanzen erlangt, die ebenfalls ihren verschiedenen Lexikalisierungen gerecht werden, wird die Ontologie durch 'Enrichment-Methoden' verbessert. Beide Methoden werden inkrementell durch Maschinenlernen durchgeführt.

3.7.2 Wissenserweiterungsmethodik

Die hier vorgestellte Methodik, wie in [44] beschrieben, ist eine Möglichkeit, den Prozess der Population einer Ontologie durchzuführen. Im Zuge der Aufrechterhaltung

von Ontologien, kann das ontologische Wissen, welches in Form von Instanzen und deren lexikalischen Synonyme (verschiedenen Lexikalisierungen) der Domäne vorliegt, erweitert werden. Um dies umsetzen zu können, wird zunächst die Ontologie mit neuen Instanzen "besiedelt", danach lexikalisch synonyme Beziehungen zwischen den verschiedenen Lexikalisierungen der Instanzen hergestellt.
Die Methodik basiert auf der Idee, dass Instanzen und deren lexikalische Vorkommen semi-automatisch auf dem neuesten Stand gehalten werden, indem periodisch das System zur Informationsextraktion unter Benutzung eines Korpus, der das erwünschte Wissen beinhaltet und welches partiell mit dem schon bekannten, in der Ontologie enthaltenen Instanzen annotiert, auf ein Neues trainiert wird.
Dieser Trainingskorpus wird schrittweise, in Abhängigkeit zu der Rate, in der Domäneninstanzen erneuert werden, erstellt, um die Existenz von neuen Instanzen und deren lexikalischen Synonymen zu sichern. Dazu muss das Verhältnis der neuen Instanzen zu den, schon in einem bestimmten Zeitintervall, bekannten und damit in dem Korpus vorkommenden, betrachtet werden. Instanzen, sowie deren Synonyme, die schon in der Ontologie sind, werden zur Annotation der Dokumente im Korpus verwendet, um eine domänenspezifische Begriffserklärung, auch im Sinne der Auflösung von Mehrdeutigkeiten, zu erreichen. Schließlich kann eine semantisch konsistente Annotation erreicht werden, und aus dem Korpus stammende Begriffe können eindeutig Entitäten aus der Ontologie zugeordnet werden. Mit diesen Annotationen wird die Trainingsmenge gebildet, die zum Training des Systems zur Informationsextraktion eingesetzt wird.
Ähnlich wie auch schon in den grundliegenden Entwurfsprozessen zum Lernen einer Ontologie (siehe Abbildung 3.2), kann ein inkrementelles Verfahren zur Ontologiepopulation und der Anreicherung angegeben werden.
Die jeweiligen methodischen Komponenten der Population [44] können im Einzelnen beschrieben werden.

- **Ontology-based Semantic Annotation**[16]

  Hier wird der Korpus mit Instanzen des Konzeptes aus der Domänenontologie annotiert. Es entsteht ein domänenspezifischer Korpus, welcher semantisch

[16] Semantische Annotation auf Basis der Ontologie

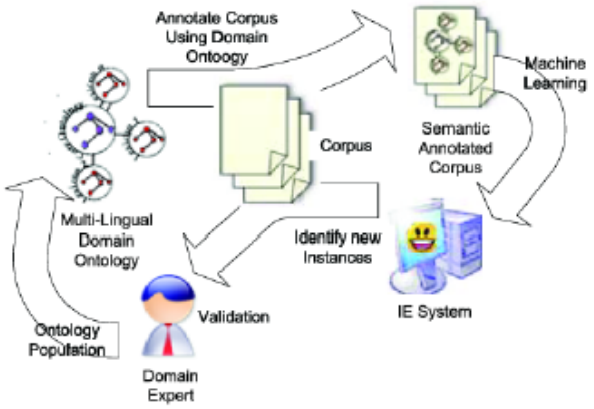


Abbildung 3.4: Methodik zur Ontologiepopulation[44]

annotiert wurde. Durch eine instanzbasierte Methode werden automatisiert Metadaten aus der Ontologie entnommen, und diese in zu lernende Dokumente eingebracht. Um das gesamte Wissen, welches in der Ontologie festgehalten werden kann auszunutzen, werden Disambiguierungs[17] Techniken eingesetzt. Dieser Schritt erzeugt Trainingsbeispiele für die nächste Phase.

- **Knowledge Discovery**

  In dieser Phase werden neue ontologische Instanzen, welche sich noch nicht in der Ontologie befinden, identifiziert. Als Modul zum Maschinenlernen dient dabei z. B. das 'Hidden-Markov-Modell' (siehe Abschnitt 4.1), das auf dem annotierten Korpus zur Informationsextraktion trainiert wird und neue Instanzen lokalisiert. Ein eigenes Markov-Modell wird für jede Menge an ontologischen Instanzen, die einem bestimmten Konzept zugeordnet werden, erstellt und trainiert. So werden Token ausgewertet, um den Kontext in denen die Instanzen der bestimmten Konzepte vorkommen, zu erschließen.

  Strukturen der Modelle werden manuell erstellt und zugehörige Parameter werden über die Daten aus der Trainingsmenge durch Berechnung der Anteile von Counts nach der Maximum-Likelihood Estimination[18] abgeschätzt.

[17] Auflösung von Mehrdeutigkeit von Begriffen

[18] Aus Statistik: parametrisches Schätzverfahren, welches Parameter als Schätzung auswählt welche gemäß dessen Verteilung die Realisierung der beobachteten Daten am plausibelsten erscheint

Diese Vorgehensweise entspricht nicht unbedingt der klassisch überwachten, durch den Anwender unterstützte, Methode, da die Domänenontologie die Beispiele zum Trainieren in Form von annotierten Instanzen liefert.
Die auf bestimmte Konzepte trainierten Markov-Modelle erkennen von nun an neue ontologische Instanzen. Extrahierte Instanzen bilden Instanzmengen von Kandidaten, welche durch den Domänenexperten bestätigt werden, um in die Ontologie einzugehen.

- **Knowledge Refinement**

  Hier soll der Anteil am Gesamtprozess, die ein Domänenexperte bei der Population von Ontologien hat, verringert werden. In erster Linie geht es dabei darum, die verschiedenen Lexikalisierungen einer gleichen ontologischen Instanz zusammenzubringen. Die Identifikation verschiedener Lexikalisierungen der existierenden Instanzen, geschieht beispielsweise durch Clusteralgorithmen, die auf der Annahme arbeiten, dass verschiedene Lexikalisierungen einer Instanz eine gemeinsame Menge von Kerncharakteren haben. So sind, der Menge nahestehende Instanzen eine potentielle alternative Beschreibung der eigentlichen Instanz.

- **Validation und Insertion**

  Im letzten Schritt werden die Kandidaten, die als ontologische Instanzen in Frage kommen, sowie deren Synonyme, die in die Ontologie aufgenommen wurden durch einen Experten validiert. Danach startet die gesamte Methode beim ersten Schritt neu, bis keine Veränderungen mehr möglich sind.

KAPITEL

VIER

Lernverfahren: Hidden-Markov-Modell

Nachdem ein Überblick über den Stand der Forschung und einigen dazugehörigen grundlegende Prinzipien zum Lernen von Ontologien gegeben wurde, soll in diesem Teil der Ausarbeitung nochmal vorgreifend eine theoretische Grundlage zu den in der Implementierung angewandten maschinellen Lernverfahren zur automatischen Extraktion von Informationen gegeben werden. Dieses Kapitel liefert theoretische Definitionen der Verfahren und zeigt angewandte Prinzipien auf.
Damit besteht dieser Teil der Arbeit aus zwei prinzipiell unterschiedlichen Methoden, die zur Aufgabe der Informationsgewinnung eingesetzt werden. Zum Einen handelt es sich hierbei um ein maschinelles Lernverfahren aus der Klasse der unüberwachten Lernansätze, das Hidden-Markov-Modell, kurz HMM genannt.

4.1 Hidden-Markov-Modell

Aus diversen vorangegangenen Arbeiten und damit verbundenen Anwendungen, beispielsweise in der Computer Vision kann die Erkenntnis gewonnen werden, dass die Hidden-Markov-Modelle Anwendung in vielerlei Gebieten finden. Besonders im Bereich der Linguistik können mit aussichtsreichen Erfolgsraten vorliegende linguistische Phänomene zunächst modelliert um daraufhin approximiert werden. Somit ist durch den Einsatz des Markov-Modells die Extraktion von Wissen im Sinne von Instanzen einer Ontologie möglich.

Das stochastische Modell der Hidden-Markov-Modelle kann auch auf Texte bzw. Textdokumente angewandt werden.
Zur Veranschaulichung der Grundidee soll ein einfaches Beispiel dienen, sodass die formale Definition der Markov-Modelle geliefert werden, bevor explizit auf die Extraktion von Informationen mit dem Modell eingegangen wird.

Ein gegebenes beliebiges Textdokument kann als eine Folge von "zufälligen" Ereignissen betrachtet werden. Dies ist linguistisch so zu interpretieren, dass auf eine Wortart, mit einer bestimmten Wahrscheinlichkeit eine andere Wortart folgt und für diese diversen Wortarten wird mit einer bestimmten Wahrscheinlichkeit wiederum ein konkretes Wort generiert. In der deutschen Sprache ist dies gut zu erkennen, da diese bestimmten grammatikalischen Regeln unterliegt. So können Stellungen innerhalb des Satzes zurückgeführt werden und ein Wort einer syntaktischen Kategorie zugeordnet werden. Im Deutschen stehen beispielsweise Adjektive vor Substantiven und nach Artikeln. Diese Tatsache kann beim Part-of-Speech Taggen ausgenutzt, und so Sätze in ihre Teile zerlegt werden, um von einem Wort auf das Folgende zu schließen. Nach diesem Prinzip kann somit ein Satz oder auch Text mit Zuständen und Übergängen modelliert werden.
Wird der Text nun als Ergebnis des gesamten Zufallsexperiments angesehen, so kann zwar eine Abfolge von Wörtern erkannt werden, jedoch nicht mehr die Folge der Wortarten die diese Wörter generierte.
Damit wird eine verarbeitete Datei auf eine Anzahl von beobachtbaren und auch verborgenen Zuständen abgebildet.
Unter dieser Annahme kann formal ein Textdokument als ein Zufallsexperiment mit einer Menge von Zufallsvariablen $\{O_1, ..., O_T, X_1, ..., X_T\}$ beschrieben werden. Demnach sind sind Werte O_i die Folge der benachbarten Daten, auch Realisierung oder Observation genannt. Die Werte entsprechen den einzelnen Wörtern die im Text beobachtet werden. Wobei die Werte der Zufallsvariablen X_i, äquivalent mit denen der "hidden" Daten sind. Damit sind sie in diesem Sinne gleichbedeutend mit den Wörtern ihrer zugehörigen Wortarten. Bei der beschriebenen Wahrscheinlichkeitsverteilung handelt es sich um eine sogenannte "gemeinsame Verteilung" (engl.: joint-

distribution) der zugrundeliegenden Zufallsvariablen und ein Hidden-Markov-Modell kann genau diese modellieren.[47]

4.1.1 Formale Definition [47]

Definition: Sei ein Hidden-Markov-Modell eine gemeinsame Verteilung der Menge von Zufallsvariablen $\{O_1, \ldots, O_T, X_1, \ldots, X_T\}$, dann ist ein Hidden-Markov-Modell ein 5-Tupel ‹S,K,π,A,B› , wobei:

- S = $\{s_{1,\ldots}s_n\}$: ist die Menge aller Zustände
- K: ist das Ausgabealphabet
- $\prod$: ist die Menge aller Anfangswahrscheinlichkeiten der Startzustände
- A: ist die Menge der Übergangswahrscheinlichkeiten
- B: ist die Menge der Ausgabewahrscheinlichkeiten

Die Zufallsvariablen aus der gemeinsamen Verteilung nehmen folgende Werte an:

> Die beobachteten Daten aus der Realisierung O_t liegen demnach in K, so gilt $O_t : \Omega \to K$
>
> Werte der zugrundeliegenden Zustände, die durchlaufen werden X_t sind also $X_t : \Omega \to S$

Der stochastische Prozess des definierten Markov-Modells unterliegt den Grundannahmen des "Begrenztem Horizont [1]" und der "Zeitinvarianz [2]".
Die Modellparameter $\prod$, A und B werden an dieser Stelle genauer definiert, wie folgt:

> $\pi = \{\pi_1, \ldots, \pi_n\}$, und damit die Menge der Anfangswahrscheinlichkeiten für jeden Zustand, $\pi_1 = P(X_1 = s_i)$

[1] Die Wahrscheinlichkeit, dass ein bestimmter Zustand s_i erreicht wird, hängt nur vom vorherigen Zustand ab

[2] Die Wahrscheinlichkeit, dass ein bestimmter Zustand s_i erreicht wird, ist immer gleich, unabhängig des Zeitpunktes

Sei A eine Matrix, die Übergangswahrscheinlichkeitsmatrix, mit Einträgen der Wahrscheinlichkeiten für Übergänge zwischen zwei Zuständen:

$$a_{ij} = P\big(X_{t+1} = s_j \mid X_t = s_i\big), \quad a_{ij} \geq 0, \quad \sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$$

Sei B Menge der Wahrscheinlichkeiten für die Ausgabe eines bestimmen Symbols $k \in K$ in einem Zustand s_i; $b_i(k) = P(o_t \mid X_t = s_i)$

Die hier beschriebenen Parameter sind die modellspezifischen Komponenten, die ein Hidden-Markov-Modell beschreiben. Sie müssen möglichst exakt gesetzt werden, damit das stochastische Modell den gegebenen Sachverhalt optimal abbilden kann und die gewünschte Aufgabe möglichst gut bewerkstelligt. Dazu müssen Markov-Modelle trainiert werden, indem die beschriebenen Parameter iterativ angepasst werden.
Das Trainieren der Parameter wird durch vorheriges Annotieren, auch als 'Taggen' des Textdokuments bekannt, durchgeführt. Dieser Vorgang liefert die Trainingsdaten, die dem Modell zum Anpassen der Parametern dienen sollen. Im Fall der natürlichsprachlichen Textdokumente, welche aus grammatikalischen Konstrukten bestehen, soll der vorgetaggte Text die Parameter so anpassen, dass das Modell für die Wortfolgen des Textes, die entsprechenden Wortartfolgen mit möglichst geringer Fehlerrate schätzt.
Das Textdokument wird nun mit den Daten der Parametern aus dem Modell repräsentiert. Demzufolge steht eine Menge von sichtbaren Daten, dem eigentlichen Text, den nicht zu sehenden Daten, nämlich einer Folge von aus dem Text generierten Wortarten, gegenüber.

Die Parameter werden mit dem nach Maximum-Likelihood-Methode bestimmt. Um die optimalen Modellparameter zu bestimmen wird der Baum-Welch-Algorithmus verwendet.
Da bestimmte linguistische Probleme gut mit einer bestimmten statistischen Verteilung beschrieben werden können, ist es möglich diese Probleme mit statistischen Methoden in Natural-Language-Processing zu verarbeiten.
Zur Beschreibung kann folgendes Szenario betrachtet werden: Die Ergebnisse eines Zufallsexperiments liegen als Menge von Daten vor, dabei sind die Parameter der

zugrundeliegenden, angenommenen Verteilung nicht bekannt. Mit einer geeigneten Methode können diese Parameter abgeschätzt werden. Ein solches Verfahren ist die bereits erwähnte Maximum-Likelihood-Methode, die für eine Menge beobachteter Daten, der Realisierung, die unbekannten Parameter der zugrundeliegenden Verteilung bestimmt, dass diese Parameter den Realisierungsdaten maximale Wahrscheinlichkeiten zuordnet.
Der angewandte Algorithmus arbeitet damit auf dem Prinzip der Expectation-Maximation[3]-Methode (EM-Methode). Die Klasse der EM-Algorithmen sind allgemeine Verfahren zur Bestimmung dieser Maximum-Likelihood-Schätzwerte von probabilistischen Modellen. Für spezielle Anwendungsgebiete muss ein entsprechender Algorithmus gefunden werden, die hier beschriebene Anwendung des Hidden-Markov-Modells verwendet wie zuvor beschrieben den Baum-Welch-Algorithmus. In seiner allgemeinsten Form ermittelt der Algorithmus damit iterativ einen Maximum-Likelihood-Schätzwert. Dies ist in erster Linie dann von Bedeutung, wenn die gegebenen Daten unvollständig sind und damit nicht alle Parameter der Daten in eine direkte analytische Berechnung des Schätzwertes eingehen können.

Baum-Welch-Algorithmus

Ein wesentlicher Bestandteil eines jeden HMM-Systems ist der so genannte "Baum-Welch-Algorithmus". Er wird benutzt, um die unbekannten Parameter eines Hidden-Markov-Models (HMM) zu finden. Der Algorithmus zählt zu den erwartungsmaximierenden Algorithmen. Er berechnet die Maximalwahrscheinlichkeitsschätzungen und die Übergangs- und Emissionswahrscheinlichkeit eines HMMs, wenn ein Trainingsdokument Emissionen enthält. Das heißt wenn ein Trainingsdokument vorher mit so genannten "Taggs" versehen worden ist. Diese "Taggs" signalisieren z. B. dem Baum-Welch-Algorithmus in einem Dokument welche Wörter zu emittieren sind.
Der Baum-Welch-Algorithmus besteht aus 2 wichtigen Schritten, die wiederholt werden bis sich die Parameter eines HMM-Modells eine nur minimale Veränderung aufweisen. Diese zwei Schritte sind:

1. Der Algorithmus berechnet die Vorwärtswahrscheinlichkeit (engl.: forward pro-

[3] Erwartungswert Maximierung

bability) und die Rückwärtswahrscheinlichkeit (engl.: backward probability) für jeden Zustand des HMMs. Während der Berechnung wird festgehalten, wie oft Übergänge und Ausgabesymbole verwendet worden sind.

2. Es wird die Frequenz der Übergangs-Emissions-Paar-Werte berechnet und durch die Wahrscheinlichkeit des gesamten Strings(Zeichenkette) dividiert. Die so berechneten Parameter, sorgen dafür das häufiger benutzte Übergänge und Ausgabesymbole eine höhere Wahrscheinlichkeit erhält als weniger häufig benutzte. Nach dieser Anpassung wird das Modell dann eine höhere Wahrscheinlichkeit für die gegebene Realisierung besitzen. Es wird also besser angepaßt sein.

Der Algorithmus sei formal wie folgt definiert: [41]

- Eingabe: Menge von Tokens $O^{(i)}$ aus Dokumenten der Trainingsmenge, Anzahl N der benötigten Ziel-Zustände

1. initialisiere die Parameter zufällig
2. benutze backward-forward, um Zustandswahrscheinlichkeit $\gamma_t(i)$ mit den aktuellen Parameter zu bestimmen $\rightarrow$ erhalten eine Zustandsfolge
3. zähle die Häufigkeit der Übergänge von S_i zu $S_j \rightarrow$ berechne daraus die neue Übergangswahrscheinlichkeit $a_{ij} = \frac{|Transisitonen_{S_i \rightarrow S_j}|}{|alleTransisitonen|}$
4. für jeden Zustand S_i zähle, wie oft eine Beobachtung O_t in diesem Zustand gemacht wird $\rightarrow$ berechne daraus die neue Ausgabewahrscheinlichkeit $b_i(O_t) = \frac{|Ausgabe_{O_t in S_i}|}{|alleAusgaben|}$
5. beginne wieder in Schritt 2 mit den neuen Parametern bis λ über eine Iteration (fast) konstant bleibt

- Ausgabe: Parameter $\lambda = (A, B, \pi)$

Abbildung 4.1: Formaler Baum-Welch-Algorithmus[41]

Zusätzlich wird ein weiterer Algorithmus in das System zur Extraktion mit dem Hidden-Markov-Modell eingebettet. Dieser findet einen optimalen Pfad, das bedeutet ein Pfad mit maximalen Wahrscheinlichkeiten von Übergängen zwischen den Zuständen. Der Algorithmus, der diese Aufgabe übernimmt, ist in aller Regel der Viterbi-Algorithmus, welcher eine Zustandssequenz in dem Modell ausmachen kann.

Viterbi-Algorithmus zur Bestimmung des optimale Pfades

Der vorgestellte Algorithmus [15] liefert eine effiziente Berechnungsmethode der optimalen Zustandsfolge, die in diesem speziellen Fall mit s^* ausgewiesen werden kann. Ein induktives Verfahren, welches der Forward-Prozedur zur Berechnung der Parameter mit Hilfe des Baum-Welch-Algorithmus ähnlich ist und sich ebenfalls die Markov-Eigenschaften zunutze macht, kommt hierbei zum Einsatz.
Im Sinne der dynamischen Programmierung kann, mit gegebenen Hidden-Markov-Modell und der beobachteten Sequenz von Symbolen, eine Sequenz von versteckten Zuständen die am wahrscheinlichsten ist, ermittelt werden. Eine so erstellte Sequenz der Zustände entspricht einem sogenannten 'Viterbi-Pfad'.
Die Prozedur lässt sich folgendermaßen beschreiben. Zunächst wird eine Wahrscheinlichkeit definiert, die partiell optimale Pfade beschreibt. Diese Pfade $\delta_t(i)$ erzeugen Anfangssegmente von der Observationsfolge bis O_t mit maximaler Wahrscheinlichkeit und enden im Zustand i. Demnach ist nach Definition:

$$\delta_t(i) = \max_{s_1,s_2,...s_{t-1}} P(O_1, O_2, ...O_t, s_1, s_2, ...s_{t-1}, s_t = i \mid \lambda)$$

Dabei sind s_i Zustandsfolgen und der Parameter i ist abhängig von einem gegebenen Markov-Modell λ.
Analog zur Bestimmung der Vorwärtsvariablen $\alpha_t(i)$ wird der optimale Pfad $\delta_t(i)$ berechnet. Einziger Unterschied ist die hier durchgeführte Maximierung der in den Vorgängerzuständen vorliegenden Wahrscheinlichkeitsanteile, anstatt einer Summation bei der Forwardprozedur.
So kann ein resultierender Algorithmus der eine Berechnung eines optimalen Pfades vornimmt, um eine optimale Zustandsfolge s^*, die somit die Produktionswahrscheinlichkeit einer Observationsfolge bei einem gegebenen Modell maximiert, angegeben werden. Der folgende Algorithmus in Pseudocode, nach der Vorlage von Fink [15], zeigt anschaulich die Prozedur des Viterbi-Algorithmus, hierbei entsprechen die Parameter b_i der Emissionswahrscheinlichkeit und π_i den Zustandsstartwahrscheinlichkeiten, bekannt aus der formalen Definition. Weitere Parameter des Algorithmus sind bekannt aus der bisher gelieferten Funktionsbeschreibung der Viterbi-Prozedur.

Viterbi-Algorithmus

Definiere: $\delta_t(i) = \max_{s_1,s_2,...s_{t-1}} P(O_1, O_2, ...O_t, s_1, s_2, ...s_{t-1}, s_t = i \mid \lambda)$
1. Initialisierung
   $\delta_1(i) := \pi_I b_i(O_1)$ und
   $\psi_1(i) := 0$
2. Rekursion
   fuer alle Zeitpunkte t, t=1...T−1:
   $\delta_{t+1}(j) := \max_i\{\delta_t(i)a_{ij}\}b_j(O_{t+1})$ und
   $\psi_{t+1} := \arg\max_i\{\delta_t(i)a_{ij}\}$
3. Rekursionsabschluss
   $P^*(O \mid \lambda) = P(O, s^* \mid \lambda) = max_i\delta_T(i)$
   $s_T^* := \arg\max_j \delta_T(j)$
4. Rueckverfolgung des optimalen Pfades
   fuer alle Zeitpunkte t, t=T−1...1:
   $s_t^* = \psi_{t+1}(s_{t+1}^*)$

Eine wichtige Erkenntnis bei Umgang des Viterbi-Algorithmus ist, dass jeweils jede Entscheidung bei der Berechnung der $\delta_t(i)$ nur lokal optimal ist. Eine global optimale Wahrscheinlichkeit ist erst nach Betrachtung der Observationsfolge auf ihrer gesamten Länge bekannt. So gilt dementsprechend Gleiches für die insgesamt optimale Zustandsfolge s^*, welche erst nach Auswertung von $\delta_T(i)$, also den partiell optimalen Pfaden ermittelt werden kann.

Prinzipiell bildet der Zustand den Abschluss der optimalen Zustandsfolge, welcher $\delta_T(i)$ maximiert. Übrige Folgeglieder werden nach Prinzip des Backtracking von Ende der Folge anhand der für die Berechnung der $\delta_t(i)$ getroffenen Entscheidungen errechnet werden. Dies wird mit Hilfe von "Rückwärtszeigern" $\psi_t(j)$ umgesetzt, welche entlang der partiellen Pfade definiert sind. Diese Zeiger speichern für die entsprechenden $\delta_t(j)$ den jeweils optimalen Vorgängerzustand, und ist wie folgt definiert:

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i)a_{ij}$$

Innerhalb der Prozedur wird ein optimaler Pfad dann folglich mit umgekehrter Richtung des Berechnungsablaufs berechnet. So wird mit der letzten partiellen optimalen Zustandsfolge

$$s_T^* = \arg\max_i \delta_T(i)$$

begonnen, um rekursiv gemäß

$$s_t^* = \psi_{t+1}(s_{t+1}^*)$$

und in umgekehrter zeitlicher Reihenfolge den optimalen Pfad aufzubauen.
Eine nicht zu vernachlässigende Einschränkung, die sich im praktischen Umgang eines solchen inkrementellen Dekodierungsverfahren ergibt, ist die Tatsache, dass eine optimale Zustandsfolge erst nach Vorlage der Gesamtheit der zu analysierenden Daten, nach Erreichen des Endes der Observationsfolge. So kann es ebenfalls sein, dass das hier beschriebene Verfahren suboptimale Lösungen liefert.

Zur Verbesserung der Performanz können Algorithmen hinzugezogen werden, die die passende Struktur des Markov-Modells suchen. Hierbei werden insbesondere die Anzahl der Zustände im Modell bestimmt, die die repräsentativsten Ergebnisse liefern. Diese Algorithmen benutzen 'Shrinkage Verfahren' [16] und statistische Optimierungen [17, 48] zur Auffindung einer bestmöglichen Struktur von erstellten Modellen.
Um eine Verbesserung im Umgang mit den vorliegenden Trainingsdaten, welche oftmals in keinen angemessenen Verhältnis zu dem Emissionsvokabular vorliegen können, zu erreichen, wird ein Smoothing-Verfahren [16] eingebunden. Diese Optimierungstechniken werden in diesem Kapitel noch ausführlicher behandelt, da diese ebenfalls Einsatz im Rahmen der Implementierung eingesetzt werden.

Die bisher beschriebene Modellierung ist die formale Grundannahme zur Verarbeitung von beispielsweise Texten die, aufgrund von Auftreten und Verwendung der möglichen Wortarten eines Satzes, natürlichsprachlicher Natur sind.
Zunächst wurde damit eine rein formale Beschreibung des Hidden-Markov-Modells gegeben, darauf aufbauend wird auf die eigentliche Anwendung zur Informationsextraktion und deren Prinzipien zum Extrahieren von Instanzen aus Dokumenten eingegangen.
Im Folgenden wird das Ausgabealphabet auch als Emission bezeichnet.

4.1.2 Modelle zur Informationsextraktion

Es kann zunächst folgende Extraktionsaufgabe betrachtet werden.
Aus einer Sammlung von Artikeln, die einen bestimmten Sachverhalt beschreiben, soll aus jedem Dokument eine bestimmte Informationsangabe extrahiert werden. So kann angenommen werden, dass für jedes Dokument in einem Textkorpus, der aus mehreren Dokumenten besteht, ein relationaler Eintrag, ein sogenanntes Template (Vorlage)besteht, der freie oder mit Text gefüllte Slots aus dem Dokument besitzt. Ein konkretes Dokument besitzt demnach verschiedene relevante Informationen zu dem Sachverhalt, die unterschiedlichen Typen angehören. Diese Typen können zum Beipiel Titel, Namen, Orts- oder Zeitangaben, Beschreibungen und anderen Entitäten entsprechen. Sie liefern damit jeweils getypte relevante Fragmente des Dokuments, welche im folgenden auch als *Felder* bezeichnet werden. Für jedes dieser Felder, falls es eine relevante Information liefert, kann ein separates Hidden-Markov-Modell trainiert werden.
Ein Textdokument aus der formalen Annahme, kann als Resultat eines stochastischen Prozesses angesehen werden. Demzufolge liefert dieser Prozess bestimmte Sprachmodelle. In vereinfachter Weise sind dies zwei Modelle, nämlich das Hintergrundmodell, welches Token (Ausgabesymbole) emittiert, die nicht zur gesuchten Ausgabe zählen, und dem Modell welches spezifische zur extrahierenden Information gehörende typische Token emittiert. Um ein Dokument mit solche einem Modell zu generieren, emittiert der Prozess im Hintergrundmodell, um dann an einem gewissen Punkt zum spezifischen Modell überzugehen und dort ebenfalls Token zu emittieren. Zuletzt kann der Prozess zurückwechseln um das gesamte Dokument zu vervollständigen. Realistischerweise kann der beschriebenen Prozess aus spezielleren Modellen in Struktur und Komplexität bestehen, die Dokumente und deren Kontext exakter nachbilden.
Das Hidden-Markov-Modell kann damit als eine Finite-State-Maschine beschrieben werden. Das Modell ist im Prinzip ein endlicher Automat, bestehend aus Knoten und Kanten, die mit den Wahrscheinlichkeiten eines Übergangs zwischen beliebigen Zuständen beschriftet sind. Dieser Automat beinhaltet stochastische Zustandsübergänge und symbolische Emissionen, welche den betrachteten Kontext und die dar-

aus entstandenen Beobachtungen aufzeigen. Eine detaillierte Betrachtung kann aus den Arbeiten von Freitag und McCallum entnommen werden [17, 16]. Diese Arbeiten begründen die Grundlage zur eigentlichen Umsetzung und Implementierung des Hidden-Markov-Modells dieser Diplomarbeit. Formal beschreibt das theoretische Modell einen zweistufigen stochastischen Prozess. Der Zufallsprozess der ersten Stufe ist von diskret stochastischer Art und betrachtet die Zustandsübergänge. Die Zustände haben Übergangswahrschcheinlichkeiten, die die Wahrscheinlichkeit eines Übergangs vom aktuellen Zustand in einen der Folgenden beschreibt.
Zusätzlich erzeugt in der zweiten Stufe der Zufallsprozess zu jedem Zeitpunkt t eine beobachtbare Ausgabe beziehungsweise Emission nach einer Wahrscheinlichkeitsverteilung, die nur jeweils vom aktuell eingenomme&nen Zustand abhängig ist.
So ist eine Folge von Emissionen beobachtbar und diese charakterisieren damit das Verhalten des Modells. Zustandsfolgen, welche innerhalb des Modells eingenommen werden, bleiben hingegen im Verborgenen (engl.: hidden).
Wie auch hier zu erkennen, wird aus dem probabilistisch generativen Prozess, der durch das Automatenmodell beschrieben wird, eine Sequenz von Symbolen (Emissionsfolge) erstellt. Dabei wird in einem Zustand gestartet, worauf ein Übergang in einen nächsten neuen Zustand erfolgt, welcher ein nach der Wahrscheinlichkeit gewähltes Symbol emittiert um danach einen weiteren Übergang erfolgen zu lassen. Dies wird abgeschlossen mit Emission eines weiteren Symbols. Dieser Prozess geht solange bis ein ausgezeichneter Endzustand erreicht wird. Das Modell umfasst eine Anzahl von Zuständen, welcher sich formalisieren lässt.

Notation zur Aufgabe der Informationsextraktion

Demnach ist mit jeder der Mengen von Zuständen $S = \{s_1, ..., s_n\}$ eine Wahrscheinlichkeitsverteilung über den Symbolen aus dem Emissionsvokabular $V = \{w_1, w_2, ..., w_k\}$ zugeordnet. Die Wahrscheinlichkeit das ein Zustand s_j ein Symbol aus dem Vokabular der Emissionen emittiert wird als $P(w \mid s_j)$ bezeichnet. Zu jedem Zustand ist die Verteilung über seine ausgehenden Transitionen angegeben, so dass die Wahrscheinlichkeit einer Bewegung von s_i nach s_j beschrieben wird mit $P(s_j \mid s_i)$. Die Zustandsverteilung des ersten Zustands wird angegeben mit $P_0(s)$.

Die wichtige Komponente der Trainingsdaten bestehen aus verschiedenen Sequenzen der beobachteten Emissionen, die mit $\{o_1, o_2, ..., o_m\}$ definiert sind.
Zur Informationsextraktion liegt also ein gegebenes Modell vor mit dessen zugehörigen Parametern. Durch Bestimmung der Sequenz von Zuständen, die am ehesten das gesamte Dokument generiert hat, wird die eigentliche Extraktion der Informationen durchgeführt. Dabei wird das Symbol extrahiert, welches verbunden ist mit dem bestimmten ausgewiesenen "Targetzustand". Diese sind sozusagen Zielzustände, die in der Trainingsphase beim "Taggen" des Korpus festgelegt werden und die dem Zustand entsprechen, an wessen Stelle die Information vorzufinden wird, welche extrahiert werden soll.
Durch das Prinzip des Taggens, also das Auszeichnen (engl.: label) von ausgewiesenen Zuständen den sogenannten Targetfeldern, entstehen zum Beispiel binäre Sequenzen, aus denen zu erkennen ist, welche Felder wichtige "Ziele" repräsentieren. Werden die Trainingssequenzen eindeutig gelabelt, so dass diese einen eindeutigen Pfad der Zustände ausweist, können Wahrscheinlichkeiten mit Verhältnis des Vorkommens (engl.: ratio of counts) in den meisten Fällen nach dem Maximum-Likelihood Prinzip berechnet werden. Durch Angabe und Annotation der Trainingsdaten lernt das Modell die Transitions- und Emissionswahrscheinlichkeiten.

Ein Modell zur Informationsextraktion mit dem Hidden-Markov-Modell lässt sich mit folgenden grundlegenden Eigenschaften beschreiben.[17, 16]

Eigenschaften des Modells zur Informationsextraktion

Jedes erstellte Modell extrahiert genau einen Typ von Feld. Ein separates Markov-Modell wird für jedes Feld konstruiert und trainiert. Damit wird jedem definierten Targetzustand genau ein Oberbegriff zugeordnet, damit Instanzen zu diesem Konzept zukünftig automatisch gefunden werden.
Modelliert wird jeweils ein gesamtes Dokument aus dem Korpus, welches nicht in einzelne Sequenzen oder Teile unterteilt werden muss, daher fällt Preprocessing dieser Art nicht an. Übergangswahrscheinlichkeiten und Emissionswahrscheinlichkeiten werden aus dem vollständigen Text des Trainingsdokuments gelernt.

Generierte Modelle haben zwei Arten von Zuständen. Zum Einen die Targetzustände und zum Anderen Nicht-Targetzustände. Die Zustände, welche keine expliziten Targets angeben, werden in einigen wissenschaftlichen Werken auch als Hintergrundzustände (engl.: background states) bezeichnet. Alle Targetzustände werden verwendet, um diejenigen Token zu extrahieren, für die das Modell trainiert wird.
Es kann eine Unterscheidung innerhalb der Menge von Nicht-Targetzuständen vorgenommen werden. Dabei ist der Zustand, welcher explizit nicht als Hintergrundzustand ausgewiesen ist und sich sequenziell vor dem Targetzustand befindet als Prefixzustand definiert werden. Implikativ zu dieser Annahme ist der Zustand, direkt dem Targetzustand folgend ein Suffixzustand, beziehungsweise beschreibt die Menge der Suffixzustände.
Ein ausgezeichneter Targetzustand, kann in speziellen Fällen aus mehreren Zuständen bestehen. Diese Art von Targetzustand wird auch als Multitargetfeld bezeichnet, die Zustände innerhalb dieses Feldes können je nach Kontext beliebig miteinander verbunden sein.
Ein Modell ist in aller Regel nicht voll verbunden (engl.: fully connected), das bedeutet, dass nicht alle Zustände jeweils untereinander erreichbar sind. Mit dieser Beschränkung der Übergangsstruktur kann ein Kontext besser festgehalten und damit die Extraktionsgenauigkeit verbessert werden.
Zusätzlich zu den klassischen Parametern des Markov-Modells, werden Label ausgezeichnet, die angeben, ob es sich um Target-Zustände oder um Observationen handelt. Die einfachste Methodik ist dabei der Umgang mit binären Werten. Ein solches Label *L(s)* kennzeichnet damit, ob ein Zustand s ein Target-Zustand ist.
Trainingsinstanzen werden ebenfalls gelabelt, damit erkannt werden kann, welche Beobachtungen aus den Target-Beobachtungsfolgen sind, die für jeweilige Extraktionsaufgaben, also bestimmten Entitäten in Frage kommen. Diese werden repräsentiert durch Sequenzen binärer Label für jede Beobachtungssequenz $\{l_1, l_2, \ldots l_m\}$.
Ziel ist es also um die Extraktion durchführen zu können, einen Algorithmus anzuwenden, der bei einem gegebenen Hidden-Markov-Modell und einer Sequenz von Emissionssymbolen, die höchstwahrscheinlichste Zustandssequenz entdeckt. Dieser Vorgang wird in folgendem Abschnitt zur Abschätzung der Modellparameter beschrieben.

Abschätzung der Parameter des Modells

Nach den Prinzipien der Modellierung zur Extraktion der Informationen, kann sich den eigentlichen Berechnungen innerhalb des Modells gewidmet werden.
Idealerweise müsste ein solches Modell die statistischen Eigenschaften der angewandten Daten möglichst gut nachbilden. Da kein Verfahren im einfachsten Fall, also bei einer gegebenen Stichprobe, zu Anfang schon ein optimales Modell liefert, muss im besten Fall mit Hilfe von Expertenwissen eine geeignete Modellstruktur in Anzahl der Zustände und Art der Emissionsverteilung vorgegeben werden. Die freien Parameter werden mit sinnvollen initialen Werten belegt, um eine iterative Optimierung in Bezug auf die betrachteten Daten vorzunehmen. Die Modellparameter werden so schrittweise verbessert, bis eine ausreichende Qualität erreicht wird. Das Modell wird auf diese Wiese trainiert.
Um ein trainiertes Modell zu erhalten, müssen die Parameter des zugrundeliegenden Modells berechnet werden. Sofern eine Struktur der Zustandsübergänge bestimmt worden ist, resultieren daraus die übrigen Parameter, welche die Wahrscheinlichkeiten für Übergänge und Emissionen sind. Diese Wahrscheinlichkeiten werden zur Informationsextraktion durch die gelabelten Trainingsdaten, die dann aus Sequenzen von Wörtern mit den ausgewiesenen Targets bestehen, abgeschätzt.
Im besten Fall bestimmen die Label einen eindeutigen Pfad durch die Markov-Struktur. Ist der Pfad der Zustände im Modell nicht eindeutig, müssen mit Hilfe der Expectation-Maximation Algorithmen in einem iterativen Prozess die Parameter geschätzt werden und der unvollständige Pfad komplettiert werden. Diese Aufgabe wird wie schon auf Seite 62 beschrieben von dem "Baum-Welch-Algorithmus", welcher ein Spezialfall der EM-Algorithmen ist, durchgeführt.
Der Algorithmus bestimmt auch hier die Parameter in zwei Phasen. In der ersten Phase, dem Estimation-Schritt werden Erwartungswerte für die Anzahl der Übergänge pro Kante berechnet, bevor im Maximation-Schritt die Modellparameter anhand der Erwartungswerte neu berechnet werden. Ein erwarteter Pfad kann so unter Beachtung der gelabelten Targetfelder exakt generiert werden. So kann die unter dem Baum-Welch-Algorithmus bekannte Forward-Prozedur zur Bestimmung der Parameter in einem Iterationsschritt für die Anwendung zur Informationsextraktion folgen-

dermaßen angegeben werden:

$$\alpha_{t+1}(s) = \begin{cases} 0, & \text{if } l_{t+1} \neq L(s) \\ \sum_{s'} \alpha_t(s) P(s \mid s') P(o_{t+1} \mid s), & \text{sonst} \end{cases}$$

Der hier definierte Parameter $\alpha_t + 1$ ist die aus der allgemeinen Definition des Baum-Welch-Algorithmus bekannten Forward-Variable und analog zur Bestimmung der Variablen, gibt es in modifizierter Form die Backward-Prozedur mit entsprechender Backward-Variable.

Die Wahrscheinlichkeiten der zu berechnenden Parametern unterliegen der Multinomialverteilung. Übergangswahrscheinlichkeiten werden durch Maximum-Likelihood mit dem Verhältnis der Zählung von Übergängen der Zustände berechnet.

Die Schätzung der Parameter, respektive der Emissionsdaten, ist etwas anspruchsvoller, da eine oft auftretende Problematik zusätzlich zu berücksichtigen ist. So sind in vielen Fällen die Trainingsdaten, die zur Verfügung stehen, extrem 'sparse'[4] bezüglich der Anzahl von Parametern des Modells. Demzufolge ist das Emissionsvokabular größer als die Anzahl von Trainingsbeispielen und eine Maximum-Likelihood Schätzung der Emissionswahrscheinlichkeit führt zu sogenannten 'Poor Estimates'. Diese haben zur Folge, dass einige Worte die Wahrscheinlichkeit Null haben.

Um diese Problematik zu verbessern aber auch die Anwendung insgesamt zu optimieren, können eine Auswahl von Techniken, die oftmals auch in Kombination miteinander Verwendung finden, eingesetzt werden. Eine solche Technik ist unter dem Begriff *'Smoothing'* bekannt, und wird im folgenden Abschnitt erklärt.

4.1.3 Optimierung der Modellstruktur

Smoothing

Die Limitiertheit der Trainingsdaten erschwert das Lernen von Modellen mit robusten Emissionswahrscheinlichkeiten. Limitierte Daten, sind in der Regel ausgezeichnete Targets die nicht ausreichend in den Trainingsdaten vorkommen und somit nicht gelernt werden können.

Um diesen Problem entgegenzuwirken, kommen bei den Hidden-Markov-Modellen

[4] auch spärlich

zur Informationsextraktion Smoothing Techniken zum Einsatz. Dabei kann unter dem englischen Begriff in der Übersetzung ein Verfahren angesehen werden, welches eine sogenannte "Glättung" der Trainingsdaten vornimmt. Wichtigeste Verfahren sind dabei unter anderem das 'Laplace Smoothing' auch 'Additives Smoothing' genannt, oder das sogenannte 'Absolute Discounting'. Die eigentlichen Ansätze hinter den Prinzipien sind aus dem Bereich der textuellen natürlichsprachlichen Verarbeitung, in denen sie erfolgreich angewandt werden. Beide Verfahren berechnen die Wortverteilung in einem Zustand nur unter Verwendung der Trainingsdaten im eigentlichen Zustand.
Alternativ zum Smoothing, welches sich zu einer uniformen Verteilung der Daten entwickelt, wie es in den hier kurz erwähnten Verfahren der Fall ist, kann der Smoothingansatz des *'Shrinkage'* eingesetzt werden.

Shrinkage und statistische Optimierung

Das hier beschriebene 'Shrinkage' kann grundsätzlich als eine weitere Smoothingmethode angesehen werden. Es lassen sich weitere Ansätze zur Optimierung, das bedeutet in diesem Fall zur Auffindung einer geeigneten Struktur des Modells, ist besagtes Shrinkage, und das zughörige Erlernen einer optimalen Transitionsstruktur, auch Topologie genannt, unter Verwendung von stochastischer Optimierung, definieren. Diese beiden Prinzipien, sind ebenfalls aus den Arbeiten von Freitag & McCallum zu entnehmen. [17, 16]
Ganz allgemein beschrieben, ist das Shrinkage eine statistische Technik, die in den Prozess der Parameterbestimmung integriert wird und welche damit die Parameterschätzungen der Emissionswahrscheinlichkeiten signifikant verbessert. Ziel ist es hierbei ein entsprechende Übereinkunft zu finden, zwischen Modellen einer einfachen Struktur, die sehr wenige bis zur minimalen Anzahl von Zuständen enthalten, und komplexeren Strukturen, die viele Zustände beinhalten. Abhängig von der Aufgabe, die der maschinelle Ansatz zum Lernen bewältigen soll, gibt es eine Differenz zwischen der Konstruktion komplexer Modelle und simpler Modelle. Ein komplexeres Modell ist in der Lage aufwendige und schwierige Strukturen der Informationsextraktionsaufgabe zu repräsentieren, jedoch liefern diese Strukturen gelegentlich schwache

Parameterabschätzungen, da Trainingsdaten höchst zerstückelt vorliegen. Vorteil der simplen Modelle ist, dass diese in sehr robuste Parameterabschätzungen führen, aber auch gleichzeitig eine schlechtere Performanz aufweisen, da zu einfache Strukturen nicht genug Ausdruckskraft besitzen um die Daten zu modellieren.
Ziel ist es also mit Variieren des Levels der Komplexität eine geeignete Zustandskonfiguration zu finden, die den verschieden Typen von Extraktionsproblemen gerecht wird.
Der zunächst behandelte Ansatz, um eine verbesserte Struktur, respektive der Parameterabschätzungen zu finden, ist durch stochastische Optimierung herbeizuführen. Es kann ein Algorithmus präsentiert werden, der automatisch gute aufgabenspezifische Strukturen findet. Damit wird das Problem gelöst, eine Selektion einer geeigneten Zustandsübergangsstruktur vorzunehmen. Die Genauigkeit einer Extraktion hängt sehr stark von der Wahl einer geeigneten Stuktur ab, die bestimmte beobachtete Phänomene besser in einer Sequenzen aus Prefixen, Targets und Suffixen festhalten kann. Der hier gelieferte Ansatz extrahiert die spezifische Datei, so dass das zugehörige Modell alle Token des Dokuments erfasst. Es gilt so genannte Targetzustände im Modell zu trainieren, um nur Symbole (engl.: token), die ein wichtiger Teil eines Satzes darstellen zu emittieren. Die Backgroundzustände geben Nicht-Targetsymbole aus. Es können hierbei Modelle geliefert werden, die sich in der Anzahl der Zustände und Verbindungen unterscheiden, so dass diese ein weites Gebiet an Textmustern abdecken können.
Der hier präsentierte Algorithmus beginnt dabei mit der minimalen Anzahl von Zuständen, was einem simplen Modell entspricht und führt auf der Menge der vorhandenen Zustände eine mögliche Operation aus. Dazu stehen verschiedene Operationen der Zustandsaufspaltung (engl.: state-splitting) zur Verfügung. Im Sinne des Hill-Climbing Prinzips im Raum der möglichen Strukturen, wird die Operation durchgeführt, die die beste Performanz liefert, um danach rekursiv nach weiteren Aufspaltungsoperationen zu forschen. Jeder Operationsschritt optimiert in unterschiedlicher Weise die Performanz, der zu bewältigenden Aufgabe.
Auf folgendem Bild sind zunächst zwei der beschriebenen möglichen Strukturen aufgeführt. Die erste dargestellte Struktur entspricht dem einfachsten Modell, welcher

in den meisten Fällen der Startkonfiguration entspricht. Die zweite Struktur ist eine komplexere Struktur, die zusätzlich aus einem Multitargetfeld besteht. In der Abbildung 4.2 entspricht ein kreisförmiger Zustand einem Nicht-Target Feld und die eckigen Zustände sind Targetfelder. An dieser Stelle kann auf die Operationen einge-

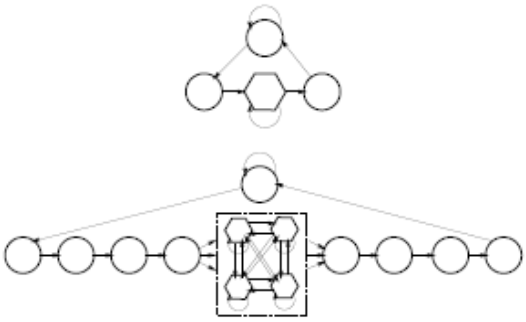


Abbildung 4.2: Strukturen des Hidden-Markov-Modells[17]

gangen werden, welche die Strukturen der Modelle verändern, das heißt die Zustände erweitern. Um das Lernen der Zustandsübergangsstrukturen durch stochastische Optimierung voranzutreiben, werden die Operationen zum Splitten eines Zustands genauer beschrieben. Dazu soll zunächst eine genauere Betrachtung der einzelnen Zustandstypen erfolgen:

- **Target** Modellieren den Inhalt der Satzteile von Targets
- **Prefix** Eine Menge aus einem oder mehreren Zuständen verbunden als String. Übergänge finden nur zum nächsten Zustand im String statt, oder sind letzter Zustand im String und damit eine Transition zu einem oder mehreren Targetzuständen. Die Modelle implizieren, dass auf dem Weg zu einem Targetzustand, mindestens ein Prefixzustand passiert werden muss.
- **Suffix** ist ähnlich zur Struktur eines Prefix. Jegliche Zustandssequenzen passieren einen Suffix beim Verlassen der Menge von Targetzustände.
- **Background** modelliert alle Texte, die nicht von anderen Typen von Zuständen modelliert werden. Die Zustände haben ausgehende Transitionen nur zu sich selbst und zum Anfang aller Prefixe. Sie haben eingehende Transitionen nur von sich selbst und dem Ende aller Suffixe.

Diese Zustände beschreiben die Klasse von Strukturen, die zur Informationsextraktion genutzt werden kann. Mit den generierten Strukturen wird der typische Inhalt eines Feldes und sein Kontext zu beiden Seiten, durch Prefix- und Suffixzustände modelliert.

Das Hill-Climbing wird im Raum der möglichen Strukturen durchgeführt, dabei wird bei jedem Schritt eine Menge von Operationen, um das aktuelle Modell zu erweitern, angewandt. Aus der Menge von möglichen Operationen, wird eine als resultierende Struktur für das nächste Modell gewählt. Jeder Zustandstyp (Prefix, Target, Suffix, Background) kann jeweils um zwei Arten von Operationen erweitert werden. Diese Operationen sind entweder das **Verlängern** (engl.: lengthen) oder **Aufspalten** (engl.: split) des Zustandes. Bei einer Verlängerungsoperation wird ein einzelner Zustand dem aktuellen Einzelzustand oder der Menge von Zuständen des jeweiligen Zustandtyps hinzugefügt, dabei müssen Übergänge sinngemäß beibehalten werden. Hingegen wird bei der Splitoperation, ein Duplikat des zu erweiterndem Zustands gemacht und Transitionen ebenfalls dupliziert und entsprechend angepasst, so dass die ersten und letzten Zustände des neuen durch Splitten entstandenen Zustandes, die selben Verbindungen zum Rest des Netzwerks hat, wie der alte Zustand. Zu beachten ist, dass die Targetzustände Transitionen auf sich selbst haben können. Nach Einsatz der Operationen kann aus jeweiligen Zustandstyp, wenn dieser zum Beispiel ein Singlezustand war, ein String aus Zuständen generiert werden.

Da die Operationen in verschiedenster Weise und Reihenfolge angewendet werden können, werden abhängig von den Modellen ganz unterschiedliche Strukturen erstellt und dies impliziert, das parallele Zustandssequenzen erlaubt sind.

Jeder Schritt, dem eine hier definierte Operation zu Grunde liegt, wird als Operation im Zustandsraum angesehen. Sobald ein Zustand der Struktur hinzugefügt wird, kann dieser nicht zurückgenommen werden. Da keine "Backtracking Operation" gegeben ist und somit jede Operation das aktuelle Modell zu einem Komplizierteren verändert, kann in diversen Fällen das Problem eines lokalen Maximums auftreten, in der die Suche nach einer optimalen Struktur im Prinzip des Hill-Climbings feststecken kann. Dem Problem des lokalen Maximums kann eine einfache Strategie entgegengesetzt werden. Der in der Arbeit von Zhang [48] präsentierte Algorithmus behält

sich nicht nur die lokal beste Operation, sondern die k besten Operationen. Demnach hat jeder Schritt des Algorithmus k aktuelle Modelle, auf denen Operationen angewandt werden können. So enthält das Verfahren Kandidatenmodelle aus denen beste Operationen ausgeführt werden.

Methode zur Struktursuche

Die Methode die in dem von Freitag & McCallum [16] vorgestellten Verfahren zur Informationsextraktion kann in Form eines Algorithmus in vereinfachter Form angegeben werden. Nachfolgend ist die Prozedur zur Optimierung einer Hidden-Markov-Struktur angegeben.

Optimierungsprozedur

```
1  Procedure LearnStructure (LabeledSet, Ops)
2      ValidSet ← 1/3 of LabeledSet
3      TrainSet ← LabeledSet−ValidSet
4      CurModel ← the simple model
5      Keepers ← {CurModel}
6      I ← 0
7      while I < 20 and CurModel has fewer than 25 states
8          Candidates ← {M | M ∈ op(CurModel) ∧ op ∈ Ops}
9          for M ∈ Candidates
10             score(M) ← average of 3 runs trained on
11                 TrainSet and scored for F1 on ValidSet
12         CurModel ← M ∈ candidates with highest score
13         Keepers ← Keepers ⋃ {CurModel}
14         I ← I+1
15     for M ∈ Keepers
16         score(M) ← average F1 from
17             3−fold cross−validation of LabeledSet
18     return M ∈ Keepers with highest score
```

Beschriebene Prozedur ist die zur Wahl einer geeigneten Struktur
Nach Anwendung des Verfahrens, kann eine gelernte Struktur zur Extraktion von definierten Targetfeldern angegeben werden. In nachfolgender Abbildung ist eine solche Struktur zu sehen. Hierbei sind ausschließlich Strukturen dargestellt, die Transitionen mit einer Wahrscheinlichkeit größer als 0.1 aufweisen, die Topologien sind als

Teilmodelle von Zuständen mit entsprechenden Wahrscheinlichkeitsangaben anzusehen.

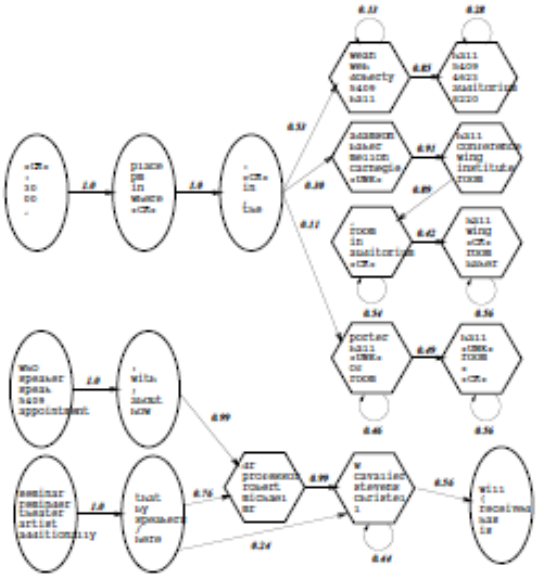


Abbildung 4.3: Gelernte Struktur mit Feldern zur Extraktion[16]

Die Abbildung 4.3 zeigt zwei Teile von resultierenden gelernten Strukturen auf, die in diesem Fall auf die Extraktion von Feldern im Kontext zu den Sachverhalten "Vorlesungsräumlichkeit", oberer Graph, und "Dozent", entsprechend unterer Graph, entworfen sind.

Jeder Zustand des Sachverhaltes Vorlesungsräumlichkeit der einem Knoten im Graphen entspricht, gibt die von oben nach unten in abnehmender Wahrscheinlichkeit fünf wahrscheinlichsten Token die durch einen jeweiligen Zustand emittiert werden an. Im Kontext bezüglich der Dozenten, sind die fünf wahrscheinlichsten Token entsprechend einem Odds-Ratio Verhältnis[5] aufgetragen.

[5]auch Quotenverhältnis: ist die statistische Maßzahl, die eine Angabe über die Stärke zweier Merkmale die in Zusammenhang stehen machen kann. Zwei Odds werden dabei miteinander verglichen und es entsteht ein Assoziationsmaß.

Sofern die ausgegebenen Token in den Zuständen nicht unmittelbar bestimmten Worten, Satzzeichen oder auch einem einzelnen Buchstaben entsprechen, können in der Knoten auch "unbekannte" Token aufgezeigt werden. Diese Art von Token, werde als 'UNK' gekennzeichnet und stehen sinngemäß für Emissionstoken, die weniger als drei Mal in der Trainingsmenge vorkommen.
Die beiden abgebildeten Modelle sind sind für die jeweilige Extraktionsaufgaben optimal bezüglich ihrer Aufteilung. So konnte ein Modell zur Extraktion der Vorlesungsräumlichkeiten in insgesamt acht Schritten der Operationen auf Zuständen gefunden werden, beziehungsweise in drei Zustandsteilungen für das Modell zu der Auffindung von Feldern zu Dozenten. Wie aus der Abbildung zu entnehmen ist, ist in diesem Fall ein einzelner relativ kurzer Kontext der Prefixzustände ausreichend, um eindeutig zu einer Räumlichkeit zu führen. Im Gegensatz dazu sind die Targetzustände in drei parallele Pfade partitioniert, bei denen der erste Pfad allein eine Transitionswahrscheinlichkeit von 53 Prozent aller Satzteile für Vorlesungsräumlichkeiten besitzt. Dieser Pfad hält viele allgemeingebräuchliche Satzteile im Kontext Vorlesungsräumlichkeiten fest, die aus zwei oder drei Token bestehen. Der letzte Pfad bildet die Phrasen ab, die nicht durch die anderen beiden Pfade so einfach modelliert werden können, insbesondere kann hier eine hohe Häufigkeit von "unbekannten" Token nachgewiesen werden.
Im ersten Targetzustand des Modells zur Extraktion der Dozenten werden zunächst Titel, Anreden und Vornamen emittiert. Der zweite Targetzustand emittiert Abkürzungen der Vornamen beziehungsweise der Nachnamen und hat zusätzlich einen Zustandsübergang auf sich selbst, um eine vollständige Modellierung eines mehrteiligen Namens abzubilden. Bei genauerer Betrachtung ist zu erkennen, dass in dem dargestellten Modell, zwei Prefixsequenzen verwendet werden und die Dozenten damit in zwei Arten von Kontexten auftreten können. Der erste Pfad modelliert die initiale formale Bezeichnung der Dozenten der Form "Speaker:" oder auch "Who:", wobei die zweite Präsentationsart der Ankündigung von Dozenten in einem weniger formalen Kontext, wie "reminder that" oder auch "seminar by", ist. Ebenfalls geht ein signifikanter Anteil der Kontexte direkt zum zweiten Targetzustand über, in diesem Kontext wird ein Titel oder Vorname des Dozenten ausgelassen.

4.1.4 Shrinkage durch hierarchische Strukturen

Für das Prinzip des Shrinkage, welches durch das Erstellen von hierarchischen Strukturen vorgenommen werden kann, können zunächst exemplarisch drei unterschiedliche Topologien, die sich in der Anzahl an Knoten und zugehörigen Kanten unterscheiden, aufgezeigt werden. Diese sind auf folgender Abbildung 4.4 zu sehen. Zu

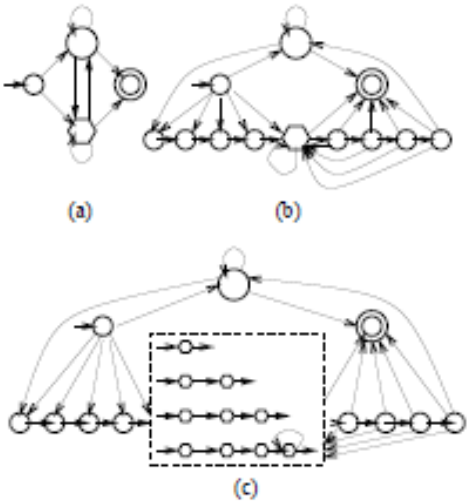


Abbildung 4.4: Topologien[16]

sehen sind mögliche Topologien mit ihren unterschiedlichen Transitionsstrukturen. Diese unterscheiden sich hier im wesentlichen in zwei topologischen Parametern, welche zum Einen Kontext-Fenstergröße (engl.: context window size) und zum Anderen Targetpfad-Count wären. Die Fenstergröße entspricht der Anzahl von Prefix- und Suffixzuständen in der jeweiligen Struktur, wobei die Anzahl der Pre- und Suffixe nicht immer gleich sein müssen, lediglich aus Gründen der einfacheren Handhabung kann diese Eigenschaft gewählt werden.
Der Pfadcount wird definiert als die Anzahl paralleler, in der Länge unterschiedlicher, sequenzieller Targetpfade. Er kann als P angegeben werden und gibt damit an, dass ein Modell P Targetpfade besitzt, die in ihrer Länge von eins bis P variieren.
In der Darstellung beschreibt das Modell (a) die einfachste mögliche Topologie. Es kann dabei angenommen werden, dass sich bei der Suche des Textes aus dem Target der zu erschließende Kontext im Bereich um das Targetfeld befindet, da nur eine

kleine Anzahl von Zuständen abgearbeitet werden kann. Andernfalls können weitere Prefix- und Suffixzustände in das Modell integriert werden, wie dies in Struktur (b) zu erkennen ist. Diese Topologie weist eine Fenstergröße von vier auf.
Eine weitere wichtige Eigenschaft der Topologien ist, dass die Target-Fragmente sich in ihren Längen unterscheiden können. Bestimmte Token die aus den Targetzuständen emittiert werden, befinden sich dabei in der Regel immer wieder zum Beispiel am Anfang oder am Ende des Fragments. Diese Tatsache lässt sich damit festhalten, indem diese Strukturteile expandiert werden. Sodass einzelne single Targetzuständen zu Arrays von parallelen Pfaden von variierender Länge werden. Zusätzlich kann der finale Zustand des längsten Pfades einen Übergang auf sich selbst beinhalten, um damit unüblich längere Targetfragmente ausdrücken zu können. In Modell (c) ist eine Struktur mit mehreren Targetpfaden aufgeführt.

Um das eigentliche Shrinkage durchzuführen, wird mit statistischen Techniken eine optimale Balance dieser Gegensätze zwischen eben den beschriebenen Strukturen gefunden. Eine Unterscheidung von Datenhäufigkeiten der Zustände kann auf die Annahme zurückgeführt werden, dass entsprechend einfache und kleine Strukturen in hoher Anzahl Übergänge in ihren wenigen Zuständen die zur Verfügung stehen aufweisen. Bei großen Topologien können eine Vielzahl von möglichen Pfaden benutzt werden, wobei insgesamt weniger Transitionen über diese Fragmente erfolgen. Eine Kombination der Abschätzungen ist unter entsprechenden Bedingungen nachweislich optimal.
So wird eine Methode angewendet, die die Abschätzungen mit einem gewichteten Mittelwert kombiniert und diese Gewichte durch Expectation-Maximation lernt. Das eigentliche Shrinkage kann dann im Sinne einer Hierarchie angesehen werden, wobei diese die erwartete Ähnlichkeit zwischen den Parameterschätzungen repräsentiert und die Abschätzungen an den Blättern eingetragen sind. Um aus einem Hidden-Markov-Modell eine Hierarchie dieser Art aufzubauen, werden Teilmengen von Zuständen definiert, die ähnlich anzunehmende Verteilungen der Wortemissionen haben, diese werden zusammengefasst und zu einem gemeinsamen Elternknoten höherer Stufe deklariert. Es ergibt sich somit eine Hierarchie von Wortverteilungen,

eine schematische Konfiguration ist in folgendem Bild in Abbildung 4.5 zu sehen. Die Konfiguration des Shrinkage nimmt sich der Datenspärlichkeit in kontextuellen Zuständen an und betrachtet explizit Shrinkage auf Nicht-Target Zuständen.

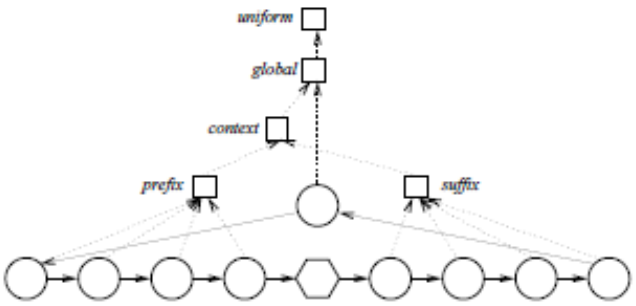


Abbildung 4.5: Hierarchie beim Shrinkage[16]

Es können folgende Annahmen für Konfigurationen des Shrinkage gemacht werden. Alle Prefixzustände haben abhängige Wortverteilungen. Dieser Sachverhalt ist an der Tatsache zu erkennen, dass alle Prefixzustände auch alternativ durch einen einzigen Knoten repräsentiert werden können, der bis zu vier Transitionen auf sich selbst erlaubt.

Die erweiterten internen Knoten der Hierarchie, können ebenfalls zu Elternknoten zusammengefasst werden. So können Emissionsverteilungen des Markov-Modells repräsentiert werden, die wieder sukzessiv einfacher sind. Auf oberster Stufe der Hierarchie kann so die anspruchsloseste aller Wortverteilungen stehen, welche der Uniformen Verteilung entspricht. Bei der Uniformen Verteilung werden entsprechend allen Wörtern des Vokabulars gleiche Wahrscheinlichkeiten zugeordnet. Aufgrund der Einbindung der Uniformen Verteilung wird eine Anwendung des Smoothings nach beispielsweise Laplace hinfällig.

Unter Abgrenzung der verschiedenen Klassen von Zuständen: Target, Nicht-Target, Prefix und Suffix, können vier Arten von Shrinkagekonfiguration angegeben werden. Das hier beschriebene Shrinkage basiert auf der Idee von Freitag und McCallum und ihrer Arbeit zur 'Informationsextraktion mit Hidden-Markov-Modellen und Shrinkage' [16]. Der Ansatz wird ebenfalls in dem Framework zum Hidden-Markov-Modell

der Universität Stanford integriert, welches in dieser Arbeit aufgegriffen wird.

- **Keine.** Kein Shrinkage durchgeführt; es wird nur die Smoothingtechnik Absolute Discounting eingesetzt.

- **Uniform.** Anstelle des Smoothing, werden alle Verteilungen von Einzelzuständen zur Uniformen Verteilung "geschrumpft".

- **Global.** Die Verteilungen von allen Targetzuständen werden zu einem gemeinsamen Elternknoten, gleiches gilt für die Uniforme Verteilung; ebenso die Nicht-Target Zustände mit einem unterschiedlichen Elternknoten.

- **Hierarchisch.** Targetzustände werden behandelt wie in dem *globalen* Verfahren. Die Klassen der übrigen Zustände werden zu einem separaten, klassenspezifischen Elternknoten geschrumpft. Die Prefix- und Suffixzustände werden desweiteren zu einem gemeinsamen "Kontext" Großvater zusammengeschrumpft. Zuletzt werden alle Nicht-Zustände, Prefix und Suffixzustände zu einem einzelnen Vorfahren geschrumpft und kann gemeinsam benutzt werden über allen Zuständen, die nicht Targetzustände sind. Ferner werden alle Zustände nochmals zu der Uniformen Verteilung geschrumpft.

Nach Einsatz der Konfigurationen werden die Parameterberechnungen shrinkagebasierte Wortwahrscheinlichkeiten liefern. So ist die neue, shrinkage-basierte Berechnung in einem Blatt der Hierarchie eine lineare Interpolation der Berechnungen in allen Verteilungen von den Blättern zur Wurzel. Die üblichen lokalen Abschätzungen werden nach bekanntem Prinzip aus den Trainingsdaten durch Maximum-Likelihood, das bedeutet nach einfachem Anteil der Vorkommen beziehungsweise Zählungen dieser. Damit sind die Trainingsdaten für einen internen Knoten der Hierarchie die Vereiningung der Daten aus seinen Blättern.

Die Schätzungen der Wortwahrscheinlichkeiten für die Knoten auf dem Pfad kann ebenfalls angegeben werden. Dieser Pfad startet in Zustand s_j und lässt sich mit $\{P(w \mid s_j^0), P(w \mid s_j^1), ...P(w \mid s_j^k)\}$ beschreiben, wobei $P(w \mid s_j^0)$ die Schätzung an einem Blatt angibt, und $P(w \mid s_j^k)$ die Uniforme Verteilung an der Wurzel. Hierbei

bezeichnet w ein Wort und s_j einen Zustand.

Die *Interpolations Gewichte* aus diesen Schätzungen sind demnach

$\{\lambda j^0, \lambda j^1, \ldots, \lambda j^k\}$, wobei $\sum i{=}1^k \; \lambda j^i = 1$.

Unter den gegebenen Voraussetzungen kann eine neue shrinkage-basierte Schätzung für die Wahrscheinlichkeit von Wort w in einem Zutand s_j mit $\hat{P}{=}(w|\ s_j)$ definiert werden. Damit entspricht dies dem gewichteten Mittel:

$$\hat{P}(w \mid s_j) = \sum_{i=1}^{k} \lambda_j^i P(w \mid s_j^i). \tag{4.1}$$

In einem weiteren Schritt kann nun die Bestimmung der sogenannten 'Mixture-Gewichte' beschrieben werden. So können die optimalen Gewichte λ_j^i zwischen den Vorfahren von Zustand s_j empirisch abgeleitet werden. Indem die Gewichte gefunden werden, die den 'Likelihood' von einigen bisher noch nicht gesehenen "Held-out Daten"[6] die in diesem speziellen Fall mit $\mathcal{H}$ deklariert werden können, maximieren. Durch eine einfache Art des Expectation-Maximation kann dabei das Maximum gefunden werden. Bei dieser Vereinfachung wird angenommen, das jedes Wort generiert wird, zunächst durch Wahl eines Knotens, beispielsweise s_j^i mit der Wahrscheinlichkeit λ_j^i, aus der Hierarchie vom Pfad zur Wurzel um danach die Wortverteilung dieses Knotens zu verwenden, um das Wort zu generieren. Wenn die Wahl der Knoten für die bestimmten Wörter unbekannt ist, wird durch Expectation-Maximation der gesamte Likelihood maximiert. Die EM Prozedur startet durch Initialisierung der λ_j Gewichte mit initialen Werten, und besagt, dass $\lambda_j^i = \frac{1}{2}$ ist , so werden die folgenden beiden EM-Schritte solange iteriert bis sich die λ_j nicht mehr verändern.

E-Step Berechnet den Grad bis zu welchem jeder Knoten die Wörter in der von Zustand s_j eigenen 'Held-out Menge' $\mathcal{H}_j$ vorhersagt:

$$\beta_j{}^i = \sum_{w_i \epsilon H_j} \frac{\lambda_j{}^i P(w_t \mid s_j{}^i)}{\sum_m \lambda_j{}^m P(w_t \mid s_j{}^m)}$$

.

[6]Das Prinzip unterliegt einer additionalen Technik zur Abschätzung von Wahrscheinlichkeitsverteilungen. Dabei werden in der Trainingsphase einige Daten zurückgehalten um ein besseres Modell zu finden. Die Held-out Daten werden dann als zusätzlicher Trainingskorpus eingesetzt.[3]

M-step Leitet neue und vor allem nachweislich verbesserte Gewichte durch Normalisierung der β Parameter her:

$$\lambda_j{}^i = \frac{\beta_j{}^i}{\sum_m \beta_j{}^m}$$

.

Die beschriebene Methode weist durch das Abstecken der 'Held-out' Menge aus den verfügbaren Trainingsdaten eine gewisse Ineffizienz auf, welche wiederum durch Schätzung des E-Step mit jeder individuellen Worthäufigkeit ausgeglichen wird.
Mit den beschriebenen Shrinkage Verfahren zur Optimierung der Strukturen eines Modells welches den Eigenschaften des Hidden-Markov-Modells unterliegt, lässt sich die Performanz der Aufgaben zur Informationsextraktion deutlich verbessern. Dabei verbessert das Verfahren nicht nur die Leistung der Systeme, sondern kann ebenfalls zu einer Senkung der Fehler(rate) für die Erkennung bestimmter Felder bei der Extraktion führen.

Performanzmerkmale zur Strukturoptimierung

Performanzmerkmale sind die schon bekannten Maße zur Beschreibung der Güte der gelieferten Ergebnisse. So kann bei der Bewertung der mit Optimierung durch Shrinkage gelieferten eben diese Angaben gemacht werden.
Dabei wird mit C die Anzahl der Testdokumente für welche die Voraussage eine Instanz des Targetfelds richtig voraussagt, angegeben. Sodann entspricht die *Precision* (P), dieses C geteilt durch die Anzahl der Dokumente für welche überhaupt eine Voraussage gemacht werden kann. Der *Recall* (R) ist definiert als, C geteilt durch die Anzahl der Dokumente, welche eine Instanz des Targetfelds tatsächlich beinhalten. Eine weitere Angabe zum Resultat kann außerdem die *F1-Measure* sein. Es beschreibt das harmonische Mittel von P und R, welches mit $\frac{2}{\frac{1}{P}+\frac{1}{R}}$ angegeben werden kann.
Aus den Auswertungen [16] ist zu erkennen, dass Shrinkage die Fähigkeit besitzt, die Performanz von HMM zur Informationsextraktion zu verbessern. Es nimmt sich auf beschriebene Weise der Problematik an, Strukturen mit komplexen Modellen zu verwenden und gleichzeitig das Fehlen von Trainingsdaten auszugleichen.

Eine "goldene Regel" kann beim Umgang dieser Art von stochastischen Modellen beachtet werden: Statistische Parameterschätzungen, wie sie beim Hidden-Markov-Modell eingesetzt werden, liefern dann gute Ergebnisse, wenn ausreichend viele Trainingsbeispiele vorliegen.

KAPITEL

FÜNF

Lernverfahren: Case-Based Reasoning

Das zweite, hier beschriebene maschinelle Lernverfahren ist das 'Case-Based Reasoning', in deutscher Literatur auch "Fallbasiertes Schließen" genannt. In diesem themenimanenten Abschnitt wird das Case-Based Reasoning verkürzt auch als 'CBR' bezeichnet.

Das Schließen aus Fällen findet ursprünglich in klassischen Anwendungsgebieten, wie Klassifikations-, Diagnose- und Help-Desk Systemen Verwendung.[38]

Anfänglich wurde die Theorie des Fallbasierten Schließens durch die Kognitionswissenschaft und ihrer Forschung des menschlichen Gedächtnisses inspiriert. Wegweisend hierzu war die Arbeit von *Schank, 1982* zur Theorie der 'Dynamic Memory'[34], welche an einer anderen Stelle des Kapitels detaillierter behandelt wird.

Ein CBR-System wird ein Modell von *Schlussfolgerungen* angeben, welches Problemlösung, Problemverständnis und Lernen berücksichtigt. Das 'Lernen' in einem solchen System wird als natürliche Konsequenz des Schlussfolgerns erfolgen. Als ein maßgebliches Charakteristikum des 'Fallbasierten-Schließens' kann die *Fähigkeit aus Erfahrungen zu Lernen* definiert werden.[22]

Um die Methodik des Case-Based Reasoning deutlich zu machen, kann ein einfaches Beispiel aus dem Alltag beschrieben werden. Dabei ist folgende Situation im Ansatz dem klassischen Szenario aus der Arbeit von *Kolodner, 1993*[22] nachempfunden.

Beispiel: *Ein Arzt stellt bei einem Patienten eine ungewöhnliche Kombination von Symptomen einer Krankheit fest, die in dieser Art zuvor noch nie aufgetreten ist. Wenn der Arzt nun einen Patienten vorher mit ähnlichen Symptomen erfolgreich behandelt hat, wird er sich an diesen alten Fall einer Behandlung erinnern und die alte Diagnose als eine Lösung für das aktuelle neue Krankheitsbild vorschlagen.*

Diese Vorgehensweise ist zunächst nicht nur sehr zeitsparend, sondern entspricht dem natürlichen Vorgehen des Menschen bei der Lösung einer noch unbekannten Problemsituation. Im Allgemeinen ist folgende Tatsache zu bemerken: Ein zweiter Problemlösungsansatz oder eine zweite Aufgabe, die bewältigt werden soll, stellt sich einfacher als eine erste heraus, da durch Erinnerung einer vorherigen Lösung ein Verhalten wiederholt werden kann. Eine neue Situation ist damit kompetenter zu beurteilen und so können z. B. Fehler auf Grund der Erinnerungsfähigkeit vermieden werden. Dieser erläuterte Sachverhalt wird 'Erfahrung' (engl.: experience) genannt.[22] Für dieses Beispiel gilt allerdings zusätzlich zu beachten, dass eine alte Lösung nicht ausnahmslos als korrekt angesehen werden sollte, sondern diese im Zusammenhang mit einer neuen Problemsituation, validiert werden muss, um sie erfolgreich als neue Erfahrung abzulegen. Erst wenn dies geschehen ist, kann von einem Lernprozess gesprochen werden, wie er beim "menschlichen Problemlösen und Lernen"[1] erfolgt. Dieser (Lern-)Prozess durch CBR soll in diesem Abschnitt der Arbeit erklärt werden.

Zunächst kann festgehalten werden, dass ein Ansatz zur Modellierung des menschlichen Lernens in Form der Fähigkeit der Problemlösung vorgenommen wird. Das CBR orientiert sich damit, wie auch im Beispiel angedeutet, an der Möglichkeit des menschlichen Denkvermögens, vorherige Probleme und deren Lösung zu erinnern. Die Erinnerung wird als eine Art Ausgangspunkt zur Lösung einer gegenwärtig, neu auftretenden Problemsituation verwendet. Fest verankert in dieser Auffassung ist dazu die Annahme, dass "ähnliche Probleme eine ähnliche Lösung haben", *Leake und Wilson, 1999*[25].

Erfahrungswissen Grundsätzlich kann angenommen werden, dass beim 'Fallbasierten Schließen' die Vorgehensweise darin besteht, dass aus den Situationen implizierte Erfahrungen in Form von Fällen (engl.: case) abgelegt und diese zum Lösen eines neuen Problems herangezogen werden. Um dies zu ermöglichen, werden für die neue Problemstellung ähnliche Erfahrungen aus dem Speicher abgerufen, um diese im jeweiligen Kontext der neuen Situation anzuwenden. Diese, aus dem Vorgang "neu gewonnene Erfahrung" wird hinzugefügt, das bedeutet: im Speicher abgelegt[39]. So entsteht eine Ansammlung von Fällen, die einer speziellen Datenbank entspricht, der sogenannten 'Fallbasis' (engl.: case base), in der die Fälle abgelegt werden. Die Fallbasis besteht aus einer Menge von Fällen.

Formalisierung Die Fälle bestehen daher im Allgemeinen zunächst aus einer (formalen) Problembeschreibung und einer entsprechenden Lösung des beschriebenen Problems. So repräsentiert ein Fall einen erfassten Vorfall, in dem ein Problem oder eine Problemsituation vollständig oder partiell gelöst wurde. Ein solcher Eintrag eines Falles entspricht in seiner einfachsten Form dem geordneten Paar: [26]

$$(Problem, Lösung)$$

.

Diese Zuordnung ist die Grundannahme zur Repräsenation eines Falles. Eine Problembeschreibung liefert demzufolge eine zugeordnete Lösung. Das hier definierte Problem-Lösungspaar ist somit der Formalismus, der nahezu in jeder Anwendung eines CBR-Systems vorzufinden ist. Einzig in der Problemlösungsmethode des 'Case Completion'[26] wird von dieser Formalisierung abgewichen.

Der prinzipielle Vorgang zum Aufbau der Fallbasis und dem damit verbundenen Erschließen neuer Fälle ist in folgender schematischen Vereinfachung [39] (Abbildung 5.1) aufgeführt.
Die grundlegende Idee [38] hinter dem Verfahren des Fallbasierten-Schließens ist also die Gewinnung von situations-spezifischem Erfahrungswissen und die anschließen-

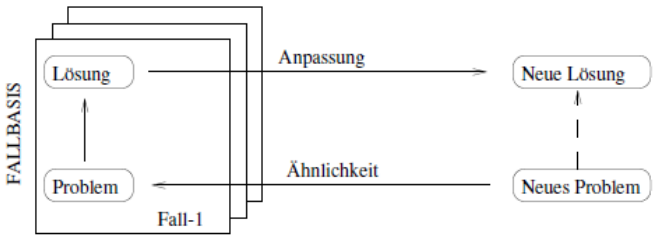


Abbildung 5.1: Fallbasiertes Schließen (prinzipielle Vorgehensweise) [39]

den Hinterlegung dieses Wissens. Eine Repräsentationsform dieses Erfahrungswissens kann zunächst in vereinfachender Annahme, eine für CBR typisches, strukturiertes dokumentiertes Problem-Lösungs-Paar sein, welches einen sogenannten Fall repräsentiert. Außerdem kann eine formale Problembeschreibung eine Attribut-Wert basierte Repräsentation sein.

Nach der Speicherung der neuen Fälle kann ein Rückgriff auf dieses Erfahrungswissen erfolgen, so dass eine effiziente Lösung der aktuell gegebenen Problemstellung vorgenommen werden kann. Dies geschieht, indem eine Problemlösung im Allgemeinen, durch die Grundannahme eines Analogieschlusses, durchgeführt wird. Ein Analogieschluss beschreibt dabei eine Schlussfolgerung, die durch Analogie[1] zwischen zwei Objekten herbeigeführt wird.

Um auf das hinterlegte Wissen zurückgreifen zu können, damit dieses erfolgreich für die Lösung des gegebenen Problems eingesetzt werden kann, muss in einem nächsten Schritt eine Auswahl von mindestens einem geeigneten Fall erfolgen.[38] Dieses sollte der aussichtsreichste Kandidat sein, der zur aktuellen Problemlösung beitragen wird.

Problemlösung Da eine neue Situation in aller Regel nicht exakt mit der im Fall vorgefundenen Problembeschreibung und der damit auch zugehörigen Lösung übereinstimmt, muss der Lösungsvorschlag in einem weiteren Schritt eventuell an die neuen Anforderungen angepasst werden. Schließlich sollte, wie auch in dem zuvor beschriebenen Beispiel, die adaptierte Lösung validiert werden, um im Erfolgsfall als neues Erfahrungswissen in der Fallbasis abgelegt werden zu können.

Die Prozedur des CBR ist ein Problemlösungsparadigma, welches sich fundamental

[1] Bsp. Muster A ähnlich B, B hat Eigenschaft C, also hat auch A die Eigenschaft C

von den Hauptansätzen der künstlichen Intelligenz unterscheidet. Anders als in den meisten Ansätzen, die ausschließlich auf allgemeinem Wissen einer Problemdomäne basieren, oder Verbindungen zwischen allgemeinen Beziehungen durch Problembeschreibungen und (Schluss-)Folgerungen erstellen, kann das Fallbasierte-Schließen *spezifisches Wissen* aus vorher erfahrenen konkreten Problemsituationen liefern.[1] Dieses Wissen ist in einem Fall gespeichert, welches Erfahrungswissen aus der Lösung eines neuen Problems durch Auffinden eines ähnlichen alten Falles und damit verbundener Wiederverwendung in der neuen Problemsituation gewonnen wurde. Fallbasierte Systeme besitzen, im Vergleich zu allgemeinen Expertensystemen, somit spezifisches Wissen in Form von Erfahrungen, welches auf die jeweilige Problemdomäne ausgelegt ist.

Des Weiteren charakterisieren *Aamodt* und *Plaza* [1] das 'Cased-Based Reasoning' als einen Ansatz, der inkrementelles, fortwährendes Lernen durchführt. Nachdem ein Problem gelöst wurde, kann eine neue Erfahrung hinzu gespeichert werden. Dieser neue "Erfahrungswert" steht umgehend für zukünftige Probleme bereit. Dadurch wird eine stetige Wissensakquisition erreicht, die zu einer Verbesserung des allgemeinen Wissens des gesamten wissensbasierten Systems beiträgt. Die Fallbasis erfährt einen kontinuierlichen "Updatevorgang" und die Kompetenz des CBR-Systems verbessert sich theoretisch mit jedem neuen Fall.
Dieses spezifische Erfahrungswissen, welches durch CBR gewonnen wird, soll in diesem Kapitel genauer untersucht werden. Dazu kann ein dynamisches Modell angegeben werden, welches wichtige Unterprozesse des Fallbasierten-Schließens aufgreift.
Die bisher beschriebene Methodik des CBR kann in Form des klassischen 'Case-Based Reasoning Cycle', der erstmalig in Arbeit von *Aamondt und Plaza, 1994* [1] definiert wurde, erweiternd beschrieben werden. Das klassische Prozessmodell, mit seinen Begrifflichkeiten, ist somit die Grundlage vieler weiterführender Ausarbeitungen.[12, 39, 26]

5.1 Der klassische CBR-Zyklus

In diesem Abschnitt wird die allgemeine Funktionsweise des Fallbasierten-Schließen erklärt. Dazu wird das klassische Prozeßmodell [1] mit seinen Phasen, die den jeweiligen Unterprozessen des Schließen aus Fällen entsprechen, herangezogen. Der CBR-Zyklus (engl.: CBR-cycle) wird dabei in einer höchsten Stufe der Verallgemeinerung beschrieben, in diesem Sinne gibt es vier grundlegende Prozesse [39]:

1. **Retrieve**: *Ermittlung bzw. Bereitstellung* mindestens eines geeigneten Falles aus der Fallbasis, dieser sollte möglichst dem aktuellen Fall am ähnlichste sein. (Abschnitt 5.5)

2. **Reuse**: *Wiederverwendung* von gespeicherten Problemlösungswissen, d. h., Benutzung der Informationen und des Wissens des gefundenen Falles zur Lösung des aktuellen Problems.

3. **Revise**: *Überprüfung* und gegebenenfalls *Modifikation* der gefundenen Lösung.

4. **Retain**: *Speichern* der neu gewonnenen Erfahrungen in der Fallbasis, die Teile sind nun benutzbar für zukünftige Problemlösungen.

Durch die aufgeführten Prozesse kann die Lösung eines Problems durch CBR und seinen aufeinanderfolgenden Phasen angegeben werden. Diese involviert zunächst das Einholen einer Problembeschreibung, anhand der Ähnlichkeit des gegenwärtigen Problems zu vorherigen Problemen, die in der Fallbasis mit ihren bekannten Lösungen gespeichert vorliegen, bemessen wird. Dadurch können ein oder mehrere ähnliche Fälle ermittelt werden (RETRIEVE). Diese auf Erfahrung basierenden Fälle können nun wiederverwendet werden (REUSE). Dabei wird die Lösung einer der bereitgestellten Fälle benutzt, diese werden entweder übernommen oder möglicherweise adaptiert, um Abweichungen in der Problembeschreibung gerecht zu werden. So entsteht ein Ausgangspunkt für eine neue Lösung. Dieser vorgeschlagene Lösungsvorschlag wird ausgewertet, um zu überprüfen (REVISE), ob das aktuelle Problem genauso zu lösen ist wie der vorher gelöste Fall, oder um die Lösung in einem Modifikationsschritt an die konkreten Bedingungen anzupassen. Die überarbeitete Problembeschreibung und

zugehörige Lösung kann schließlich in der Fallbasis als neuer Fall gespeichert werden (RETAIN), und der Fall steht künftig für Anfragen zur Verfügung. Das System hat mit diesem Prozess gelernt, ein Problem zu lösen.[1, 39]
Der hier beschriebene Vorgang ist in Abbildung 5.2 aufgezeigt.

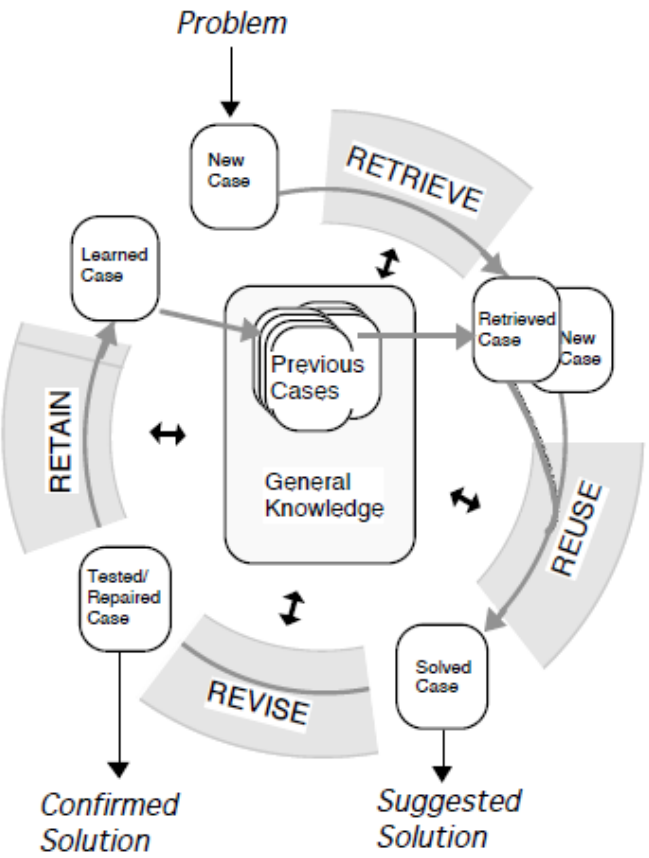


Abbildung 5.2: Case-Based-Reasoning-Zyklus [1]

Prozessablauf Der Prozess zur Erstellung eines neuen Falles durchläuft die vier Phasen des Modells wie folgt: Eine initiale Beschreibung des Problems definiert einen *neuen Fall*. Dieser neue Fall wird verwendet, um im Retrievalschritt einen Fall aus der Sammlung der *bisherigen Fälle* wieder aufzufinden. So wird der *abgerufene Fall* mit dem neuen Fall kombiniert, um im Reuseschritt einen *gelösten Fall* zu erhalten, der eine vorgeschlagene Lösung des initialen Problems ist. Im Reviseprozess wird diese Lösung auf Erfolg getestet, diese kann zum Beispiel in einer realen Umgebung an-

gewandt oder durch einen Experten (engl.: teacher) ausgewertet werden. Sofern der Test negativ ausfällt, also keine Aussichten auf Erfolg bestehen, kann eine "Reparatur" vorgenommen werden, mit der der neue Fall modifiziert wird. Wird eine Lösung bestätigt, kann diese in der Retainphase als nun nutzbare Erfahrung zur zukünftigen Wiederverwendung gespeichert werden. Die Fallbasis wurde durch einen neuen *gelernten Fall* oder durch Modifikation eines existierenden Falles aktualisiert.[1]
Wie in Abbildung 4.5 zu erkennen ist, kann die Fallbasis sinnbildlich für das allgemeine Wissen des Systems angesehen werden, welches die Prozesse des CBR unterstützt. Es lässt sich beschreiben als allgemeines domänen-abhängiges Wissen, das dem spezifischen Wissen, enthalten in den Fällen, entgegensteht.[1]
Zu erwähnen ist, dass das klassische Modell in der Form wie es hier nach *Aamodt und Plaza* angegeben wurde, mittlerweile eine Erweiterung erfahren hat. *Reinartz et al.* erweiterten das Prozessmodell durch zwei weitere Phasen: 'Restore' und 'Review'. Zur weiteren Recherche hierzu kann die Arbeit [32] verwendet werden.
Da der Retrivalphase im 'CBR' eine besondere Bedeutung zukommt, wird eine genauere Beschreibung dieses Prozesses in einem folgenden Kapitel (Abschnitt 5.5) erfolgen.

5.2 Knowledge Containers

Eine genaueren Betrachtung des eingebetteten Wissens innerhalb eines CBR-Systems kann ebenfalls aus einer anderen Perspektive vorgenommen werden. Hierbei geht es weniger darum die Module, welche einzelne Unteraufgaben (im Sinne von Subroutinen) des Systems beschreiben, wie im modulbasierten Konzept des inkrementellen CBR-Zyklus zur Struktur des Lösungsansatzes, aufzuzeigen. Hinter der Beschreibung steht die Idee eines 'Wissenscontainers' (engl.: knowledge container) nach *M. Richter 1995*, einem Strukturelement, welches dem Wissen im System eine Struktur zu geben vermag und damit die Methoden, die zur Wissensrepräsentation verwendet werden angeben.[26, 27]
Das Case-Based Reasoning kann als eine wissensbasierte Technik beschrieben werden. Ein wesentlicher Vorteil ist dabei die Integration von weiterem semantischen Wissen.

Dieses semantische Wissen kann z. B. in der Wissensakquisitation eingebracht werden, indem ein Domänenmodell erstellt wird, welches essentielle Eigenschaften der Domäne, die Entitäten die verwendet werden und deren Beziehungen im Sinne einer Ähnlichkeit beschreibt. Jede "Wissenseinheit" kann demnach durch mindestens eine der folgenden Kategorien [26, 27] repräsentiert werden:

- Der Definition eines Indexvokabulars der Fälle
- Der Konstruktion eines Ähnlichkeitsmaßes
- Der Fallbasis des Systems
- Der Lösungsadaption des Wissens

Der Container des Indexvokabulars umfasst so, z. B. Attribute und Prädikate zur Beschreibung der Fälle auf verschiedenster Abstraktionsstufe. Das Indexvokabular beschreibt also die Terme, die in der Domäne benutzt werden können.
Das Ähnlichkeitsmaß ordnet diese verschiedenen Terme, sowie Anfragen und Dokumente einander zu. Der Container beinhaltet damit die Indexierung, Ähnlichkeitsfunktionen, Sammlung von Ähnlichkeiten und die Selektion der Fälle.
Der Wissenscontainer der Fallbasis besteht aus der Erhaltung der Fallbasis und Aufnahme von Fällen und deren Struktur, die durch die Repräsentation bestimmt werden.
In der Lösungsadaption wird Wissen zum Einsatz in den Reuse- und Revisephasen des CBR-Zyklus verlangt, um eine spezifizierte Lösung zu transformieren.[26, 27]
Anhand dieses Modells ist also auch schematisch aufgezeigt, wie Wissen im Prozess des Fallbasierten Schließens aufgenommen werden kann, dies geschieht in den jeweiligen Containern. Ein Lernvorgang kann in den Fällen bzw. in der Fallbasis stattfinden, weshalb diese Komponenten einen wesentlichen Anteil am Erfolg des gesamten Wissensakquisitationsvorgang haben.
Im nächsten Abschnitt kann nun geklärt werden, wie das Wissen in geeigneter Weise in den zur Verfügung stehenden Fällen repräsentiert und in weiterer Verwendung in einer Fallbasis organisiert wird.

5.3 Fallrepräsentation und Organisation der Fälle

Um eine geeignete Form der Repräsentation der Fälle in einem CBR-System zu finden, sollte nochmal die Frage "*Was ist ein Fall?*" vergegenwärtigt werden. Damit verbunden ist die Überlegung, was inhaltlich in einen Fall gespeichert werden soll, und wie dies am geeignetsten durch einen Fall repräsentiert werden kann. Dies ist vor allem in der Phase der Fallerstellung (engl.: case authoring) von fundamentaler Bedeutung, da hier Überlegungen bezüglich der internen Repräsentation eines Problems in den dann erstellten Fällen vorgenommen werden.
Aus einer klassischen Definition der Kognitionswissenschaft bezüglich des 'Episodischen Gedächtnisses'[2][42] kann abgeleitet werden, dass ein Fall innerhalb eines CBR-Systems, welcher also mit dem Erfahrungswissen aus Erinnerungen eines Menschen annähernd gleich zu setzten ist, eine Abstraktion von Ereignissen ist, die in Zeit und Raum begrenzt sind. So konnte ein Fall ursprünglich als 'Problem Solving Episode' bezeichnet und sinnvollerweise mit einer "Begebenheit zur Problemlösung" übersetzt werden.
Kolodner beschrieb in ihrer klassischen Auffassung einen Fall im CBR als: "ein kontextualisiertes (Teil-)Stück von Wissen, welches eine Erfahrung repräsentiert, die eine fundamentale Lektion um die Ziele eines Reasoners[3] zu erreichen, lehrt"[22]. Diese Betrachtung wurde von *Bergmann und Kolodner*[6] spezifiziert, indem angenommen wurde, dass dieser kontextualisierte Teil von Erfahrung, in verschiedenster Form repräsentiert werden kann.

Wie anfänglich in diesem Kapitel beschrieben, wird diese Erfahrung, die ein Fall repräsentiert, oftmals nur unterteilt in eine Problem- und Lösungsbeschreibung. So sind diese beiden Informationen ein Minimum an Bestandteilen die einen Fall repräsentieren. Damit entspricht ein Problem einer Beschreibung der Umgebung, in der ein Fall aufgetreten ist. Die zugehörige Lösung entspricht dann der Beschreibung des Lösungswegs für das auftretende Problem.

[2]Def.: Ist die Erinnerung an autobiographische Ereignisse, wie Zeiten, Orte, assoziierte Gefühle und anderes kontextuelles Wissen, welches explizit angegeben werden kann.

[3]System, welches die Fähigkeit besitzt logische Konsequenzen zu erschließen

Abhängig von der beabsichtigten Wiederverwendung ist es häufig sinnvoll zusätzliches Wissen anzugeben. Mögliche weitere Informationen entsprechen beispielsweise Angaben zur Güte, Beschreibungen zur Qualität der Ausgabe oder einer Kostenbewertung des Falles. Eine explizite Angabe für eine umfangreiche Fallstruktur kann aus folgenden fünf Angaben bestehen: [22]

- Einer Situation und deren Ziele.
- Einer Lösung und gegebenenfalls deren Mittel um diese abzuleiten.
- Einer Menge von Resultaten des Ausführens.
- Erklärungen der Resultate.
- Lehren, die aus den Erfahrungen gelernt werden können.

So erfährt der einfachste Formalismus in Form eines Problem-Lösungspaares eine Erweiterung, die einen Fall mit weiteren Informationen unterstützt.
In der Fallrepräsentation werden grundlegende Formalismen durch *Bergmann et al.* festgelegt. Damit ergeben sich drei Grundtypen von Repräsentationsformalismen der Fälle[6] in einem System. Diese können folgendermaßen unterschieden werden:

1. Die *Feature-Vektor-Repräsentation*, hier liegen Fälle propositional[4] vor.
2. Der Typ, der *strukturierten* oder auch relationale Fälle liefert
3. Den Formalismus *textueller* Fälle, welche auch als semi-strukturiert zu beschreiben sind.

Feature-Vektor-Ansatz Der klassische *Feature-Vektor-Ansatz* repräsentiert einen Fall als Vektor eines Attribut-Wert-Paares und kann mit Verfahren, die 'k-nearest Neighbor Matching' und 'Instanzbasiertes Lernen' unterstützen, durchgeführt werden. So werden sogenannte Kategorien (engl.: category), in welche die Vektoren mit ihren Featurewerten eingestuft werden, extensional repräsentiert als eine Ansammlung von Fällen, die *Exemplare* genannt werden[6]. Damit können neue Fälle

[4] auch: verhältnismäßig

in Kategorien klassifiziert werden, sobald ein Match zwischen einem Exemplar und einem neuen Fall gefunden wird. Dieser Matching-Prozess liefert dementsprechend eine Begründung, welche die Merkmale (engl.: features) des neuen Falles mit den Exemplaren in Verbindung bringt. Genauer betrachtet, ist es der Prozess des Retrieval, der eine Erklärung der (Ähnlichkeits-) Relation zwischen neuen und wiedergefundenen Fall aus der Menge der Exemplare liefert. Hier kann von einem wissens-intensiven Ähnlichkeitsmaß gesprochen werden.

Strukturierter Ansatz Im strukturierten Ansatz der Fallrepräsentation ist die Idee der 'Episodic Memory'[42] von Fällen aufgegriffen. Hierbei wird die Struktur um einen frame-basierten Formalismus entwickelt. Die Repräsentationen werden (partiell) durch Beschreibungslogik formalisiert. Damit können Fälle als Frames oder frame-ähnliche Strukturen repräsentiert werden. Die so erhaltenen strukturierten Fälle, werden nun als Terme, die in der sogenannten Feature-Logik, einer Spezialisierung der Beschreibungslogik stehen, angesehen. Diese Art von relationalen Repräsentationen entsteht, indem Cluster von Relationen, die in elementaren Objekten zusammen vorkommen, zusammengefasst werden.

Domänenwissen kann unter Benutzung einer Sortierhierarchie integriert werden. Zusammengesetzte Fälle, also Fälle die andere Objekte oder Unterfälle zusammen gruppieren, können durch die Tatsache, dass Unterterme auch Terme sind, gesehen werden. Ähnlichkeit zwischen zwei Fällen wird mit dem Konzepten von Subsumtion[5] und Anti-Unifikation[6] der Terme umgesetzt.
Ein in der Ausdrucksfähigkeit zum frame-basierten Ansatz ähnlicher Formalismus, ist eine objekt-orientierte Repräsentation. Der Ansatz wird daher häufig in der Klasse der strukturiert relationalen Formalismen geführt. Die Fälle werden als eine Sammlung von Objekten repräsentiert, wobei die Objekte durch eine Menge von Attribut-Wert-Paaren beschrieben werden. Die Modellierung der Daten erfolgt nach typischen

[5]Zur Berechnung ob ein Konzept eine Teilmenge eines anderen Konzeptes ist

[6]Operation aus der Logik: Methode zur Vereinheitlichung prädikatenlogischer Ausdrücke, findet in zwei Termen oder Strukturen Features die übereinstimmen und bildet eine allgemeinste Gemeinsamkeit

objekt-orientierten Paradigmen, in Form von "is a" bzw. "part of" Relationen, die durch Vererbungsprinzip in Verbindung stehen. Eine Objektklassse beschreibt die Struktur eines Objektes. Weitere Information bezüglich objekt-orientierter Ansätze liefert die Arbeit von *Bergmann, 2002* [5].

5.3.1 Textuelle Repräsentation

Zuletzt wird bei der textuellen Fallrepräsentation von schwachen Strukturen der Fälle ausgegangen. So werden Textfelder üblicherweise als Mengen von linguistischen Einheiten, die durch 'Stemming-Algorithmen'[7] gewonnen wurden, repräsentiert. Eine in dieser Form schwache Struktur der Texte ermöglicht eine bessere Interpretation, damit eine einfache Wiederverwendung der Fälle garantiert werden kann. Ebenfalls kann deshalb die Nutzung einer Vielzahl unterschiedlichster textueller Quellen erfolgen.

Prinzipiell wird ein Text in **Entitäten von Informationen** (engl.: information entities) zerlegt, die einen Fall erzeugen. Diese Entitäten sind relevante Wörter oder Ausdrücke im Text, die damit eine Wiederverwendbarkeit des episodischen Wissens, dass im Fall festgehalten wird, festlegen. Durch ein gegebenes Vokabular an relevanten Ausdrücken oder Wörtern, können Fälle aus Text nach **Informationsentitäten** (IE) abgearbeitet und eine automatische Akquisition von Fällen vorgenommen werden.

Durch CBR kann folgende Organisation der Fälle, die die Fallbasis bilden, vorgenommen werden: In Form eines gerichteten Graphen mit Knoten, welcher die Fälle mit ihren Informationsentitäten repräsentieren, wird ein sogenanntes 'Case Retrieval Net (CNR)' aufgebaut. Die Knoten sind gemäß ihrer Ähnlichkeit verbunden und entsprechend gewichtet.

Anmerkend gibt es weitere fortgeschrittene Ansätze[6], wie: hierarchische Fallrepräsentationen, generalisierte Fälle, Fälle in fallbasiertem Design, sowie Fälle in fallbasiertem Planen, die in dieser Arbeit nicht weiter behandelt werden.

Nach Betrachtung der einzelnen Repräsentationsformen, kann festgehalten werden,

[7] Aus linguistischen Morphologie: Reduziert Wörter auf ihren Wortstamm oder Ursprung

mit welchen Methoden die einzelnen Formalismen im Retrieval ihre Ähnlichkeit bestimmen, bevor im nächsten Abschnitt auf einen weiteren Teilaspekt, ein geeignetes Ähnlichkeitsmaß zum 'Retrieval' der Fälle zu finden, eingegangen wird.
So wird z. B. eine distanz-basierte Metrik angewandt, um Ähnlicheiten von Feature-Vektoren oder wissensintensive Indizierung in Verbindung von Matching-Algorithmen für frame-basierte Repräsentationen zu finden. Bei textuellen Repräsentationen kommen Techniken aus dem Information Retrieval zum Einsatz.
Eine wichtige Erkenntnis ist, dass eine enge Verknüpfung zwischen Repräsentation der Fälle und der Weise, mit der die Ähnlichkeit während des Retrievals gefunden wird, besteht[6, 1]. Diese beiden Probleme des CBR sollen an dieser Stelle genauer betrachtet werden.

Datenorganisation Die Performanz eines CBR-Systems ist eng gebunden an die Struktur und den Inhalt seines Fallspeichers[8]. Da eine neue Problemlösung durch Wiederaufruf eines zur Lösung geeigneten vorherigen Falles durchgeführt wird, muß zum einen die *Fallsuche* und zum anderen der *Matching-Prozess*, sowohl effektiv als auch angemessen effizient, in der Zeit sein. Zudem muss eine entsprechende Anforderung erfüllt werden, damit die Erfahrung eines gelösten Problems abgelegt, das heißt mit einer geeigneten Methode in den Speicher aufgenommen wird. Diese Aspekte sind bekannt unter dem "*Repräsentationsproblem*", bei dem es zu beantworten gilt: Was muss in einem Fall gespeichert werden? Welche geeignete Struktur zur Beschreibung der Inhalte eines Falles kann gefunden werden? Außerdem kann an dieser Stelle die Entscheidung getroffen werden, wie der Fallspeicher organisiert, also im einfachsten Fall indexiert wird, um effektives 'Retrieval' und 'Reuse' zu ermöglichen. Ein zusätzliches Problem besteht darin, die Struktur des Fallspeichers in das Modell des allgemeinen Domänenwissens zu integrieren. Ein grundlegendes Fallspeichermodell ist das 'Dynamic Memory Model' nach *Schank und Kolodner*[1]. Die Organisation der Fallbasis innerhalb eines System wird mit Betrachtung dieses klassischen Modells aufgezeigt. Es geht dabei in erster Linie darum, durch die geeignete Organisation eine gute Methodik für Zugriffe auf die Fallbasis zu ermöglichen. Im hier betrachteten

[8] Fallbasis

Ansatz wird dies die Möglichkiet sein, eine Indexierung der Fälle innerhalb der erzeugten Fallbasis zu bewirken.

5.4 Indizierung: Dynamic-Memory-Ansatz

Die hier gelieferte Organisationsart ist das klassische Modell und damit Vorläufer für viele folgende Weiterentwicklungen. Anhand des hier gelieferten Ansatzes kann sehr anschaulich gezeigt werden, wie das zu speichernde Wissen in einem System modelliert und organisiert wird.
Der 'Dynamic-Memory-Ansatz' orientiert sich an kognitionswissenschaftlichen Hintergründen und ist damit "ein plausibles und vereinfachtes Modell, welches das Prinzip des menschlichen logischen Denkens und Lernens abbildet"[1]. Für die praktische Umsetzung lieferte das 'Dynamic Memory Model' nach *Schank* [34] erste Erkenntnisse, bevor ein auf diesen Annahmen basierendes erstes Case-Based Reasoningsystem unter dem Namen 'CYRUS' durch *Kolodner* [22] entwickelt wurde.

'Script Theorie' Als innovative Idee erwies sich dabei die Entwicklung der 'Script Theorie'[39], in der ein 'Script' als Wissensrepräsentationsformalismus aufgefasst werden kann. Der Theorie nach stellt ein Script eine Art Gedächtnisschema dar, welches Gedächtnisinhalte strukturiert und so zur Analyse von Ereignissen und Handlungen beiträgt. So können jeweils spezifische Aktionen vorhergesagt werden, die allgemein in bestimmten Situationen auftreten. In einfachster Weise ist ein Script damit eine sequenzielle Auflistungen von aufeinanderfolgenden Ereignissen bzw. Aktionen. Hier vorliegendes Wissen ist zunächst nur in allgemeiner Form vorhanden, auf spezifischeres Wissen, für Aktionen, die mit Hilfe des Scripts vorhergesagt werden können, kann nicht zurückgegriffen werden. Dieses konnte in einer Erweiterung mit der Dynamic-Memory-Theorie erschlossen werden. Dieser Ansatz führte zur Entwicklung von dynamischen Gedächtnisstrukturen, welche "Vorgänge des Verstehens, Erinnerns und Lernens eng miteinander verknüpfen [39]", da sie auf gleichen Gedächtnisschemata basieren. Es werden erlebte Episoden gespeichert, welche ebenfalls für das Auslösen eines darauffolgenden Lernprozesses verantwortlich sein können.

Dynamisches Gedächtnis Wichtigste Komponente der Theorie des dynamischen Gedächtnisses ist die Verwendung von speziellen Strukturen, um Speicherinhalte gezielt miteinander zu verbinden. Eine solche Speicherstruktur wird als 'MOP' bezeichnet und steht für Memory Organization Packet. Diese Pakete enthalten Informationen darüber, wie Elemente des Speichers zueinander in Verbindung stehen. Ein einzelnes Paket (MOP) kann dabei als ein zuvor beschriebenes Script angesehen werden, welches durch Indizes auf spezielleres Wissen zugreifen kann. Dieses spezielle Wissen lässt sich in dieser Modellierung als eine Ausnahme beschreiben, welche sich von den Paketen allgemeiner Beschreibung unterscheidet. Die Abbildung 5.3 zeigt eine graphische Darstellung eines 'Memory-Organization-Pakets' aus der allgemeinen MOP-Theorie [34] auf. In dem Ansatz werden die MOPs in einer Spezialisierungs-

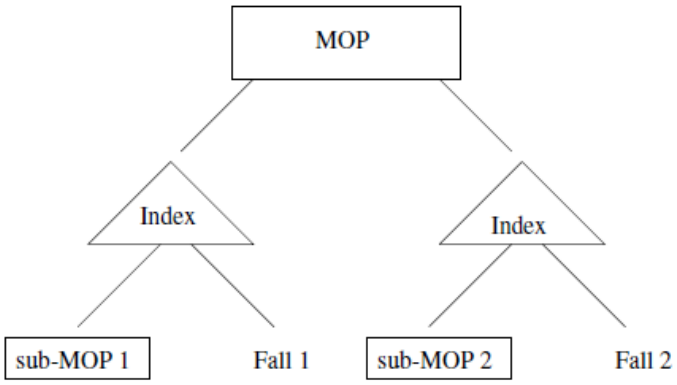


Abbildung 5.3: Memory Organization Packet (MOP)[39]

hierarchie angeordnet. Dabei liegt ein höherer Spezialisierungsgrad vor, je tiefer die Stufe des MOPs in der Struktur ist. Eine Spezialisierung ist in der Graphik mit der Bezeichnung sub-MOP angegeben. So kann der Dynamic-Memory-Ansatz sowohl allgemeines Wissen, als auch spezielles Wissen in seiner Struktur organisieren. Hierzu besitzt ein MOP eine sogenannte Norm, welche Merkmalen, die das allgemeine Verhalten des MOPs beschreibt, entsprechen.

Beispiel Anhand eines Beispiels einer Speicherstruktur zur Verwaltung von Dinosauriern kann dies aufgezeigt werden. In der Annahme besitzt ein MOP die Norm: Nahrung = Fleisch, ein in der Struktur tiefer liegendes sub-MOP hat die Norm: Nahrung = Aas.

Die Norm Nahrung = Fleisch wurde somit überschrieben, und alle Fälle, im Beispiel entsprechen diese Dinosaurierarten, die von sub-MOP verwaltet werden, diesem also nachfolgen, sind Aasfresser. Die Dynamic-Memory Methode kann damit allgemeines und spezielles Wissen organisieren.

Das dynamische Verhalten des Speichers lässt sich anhand der Reorganisation der Memory erklären. Folgende Vorgänge in der dynamischen Speicherstruktur, die als Erfahrungen bezeichnet werden, verändern die 'Dynamic Memory': Speichern neuer Fälle, Einfügen neuer und das Ändern bestehender Indizes und Erzeugung neuer Generalisierungen (MOPs) aus Fällen. Außerdem ermöglicht die Indizierung der Struktur das Auffinden des Speichereintrags, der einer aktuellen Situation am nächsten kommt. Der beschriebene Vorgang entspricht dem Erinnern.

Um neue Fälle in der Struktur abzuspeichern, muss eine entsprechende Position im Netz aufgefunden werden. Die Indizes, die im neuen Fall angegeben werden, liefern dabei eine Menge von möglichen Pfaden durch die Struktur. An den aufgefundenen potentiellen Positionen kann sich ein schon gespeicherter Fall oder ein MOP befinden. Bei erster Möglichkeit (Abbildung 5.4) wird der vorliegende Fall mit dem neuen Fall verglichen. Ähnlichkeiten werden in einem neuen MOP untergebracht und in einer sich unter diesem MOP befindender Stufe mittels ihrer Unterschiede über einen Index indiziert.

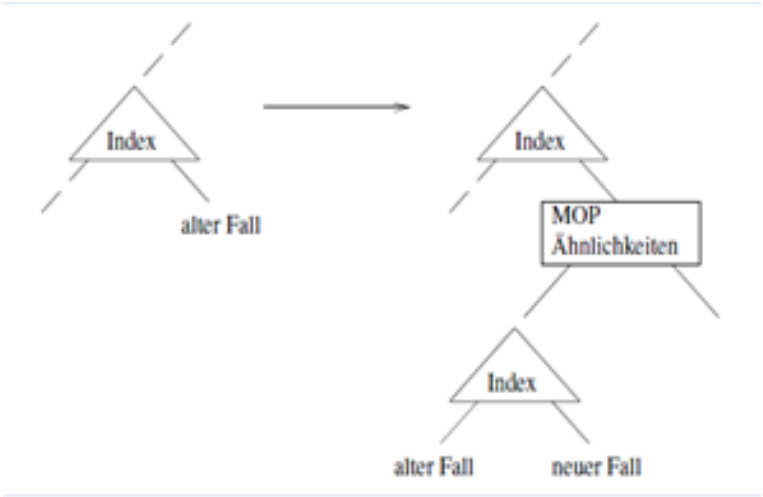


Abbildung 5.4: Dynamic Memory vor und nach dem Einfügen des neuen Falles [39]

Sofern sich an der potentiellen Position zum Einfügen ein MOP befindet, entsprechend der zweiten Möglichkeit (Abbildung 5.5), wird ein neuer Fall unterhalb dieses MOPs indiziert.

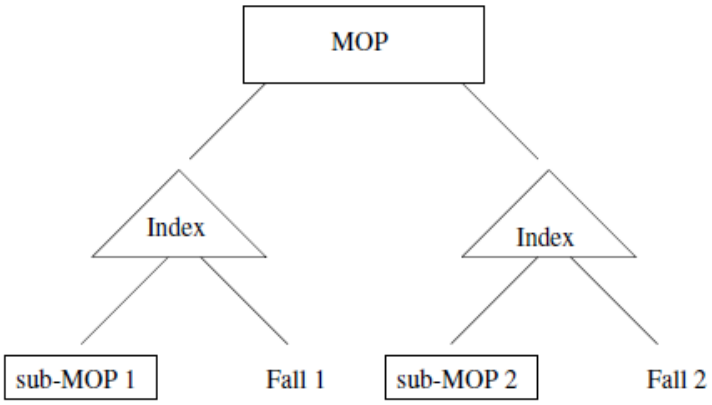


Abbildung 5.5: Dynamic Memory nach Einfügen des neuen Falles [39]

Anhand des Beispiels der Reorganisation des Netzes ist die dynamische Eigenschaft der Speichers gut zu erkennen.

In der dynamischen Speicherstruktur ist es möglich Schlußfolgerungen durchzuführen. Eine Schlußfolgerung zu ziehen entspricht in diesem Modell der Aufgabe, bei Eingang eines neuen Falles, genau diese Fälle in der Struktur zu finden, die dem neuen Fall am ähnlichsten sind. Dies erfolgt durch Definition einer Menge von Pfaden, durch die Speicherstruktur aus einer Menge von Merkmalen des neuen Falles. Eine erneute Reorganisation wird in Folge einer erfolgreichen Folgerung durchgeführt, um das neue Fallwissen in die Struktur einzubetten.[39]
Ein Fallspeicher kann nach Dynamic-Memory-Ansatz mit einer hierarchischen Struktur aufgebaut werden. In einer Erweiterung der allgemeinen MOP Theorie nach *Schank*, dem 'Episodic Memory Organization Packets'[23], kurz E-MOPs, werden bestimmte Fälle, die ähnliche Eigenschaften teilen unter einer allgemeineren Struktur organisiert. Diese Struktur wird *'generalized episode'* (GE) genannt und beinhaltet drei verschiedene Arten von Objekten. Diese sind: Normen, Fälle und Indizes. Wie zuvor sind Normen Merkmale (engl.: features), die allen Fälle, die unter einer GE ge-

meinsam indiziert werden. Indizes sind Merkmale, welche zwischen den Fällen eines GE eine Unterscheidung herbeiführen. Damit kann ein Index zu einer spezifischeren *Generalized Episode* oder zu einem Fall zeigen. Ein Index ist zusammengesetzt aus zwei Termen: einem Indexnamen und einem Indexwert.[39]
Die Struktur bildet ein Netzwerk auf Basis von Unterscheidungen (engl.: discrimination network). Eine komplexe Struktur einer Generalized Episode ist in Abbildung 5.6 dargestellt.

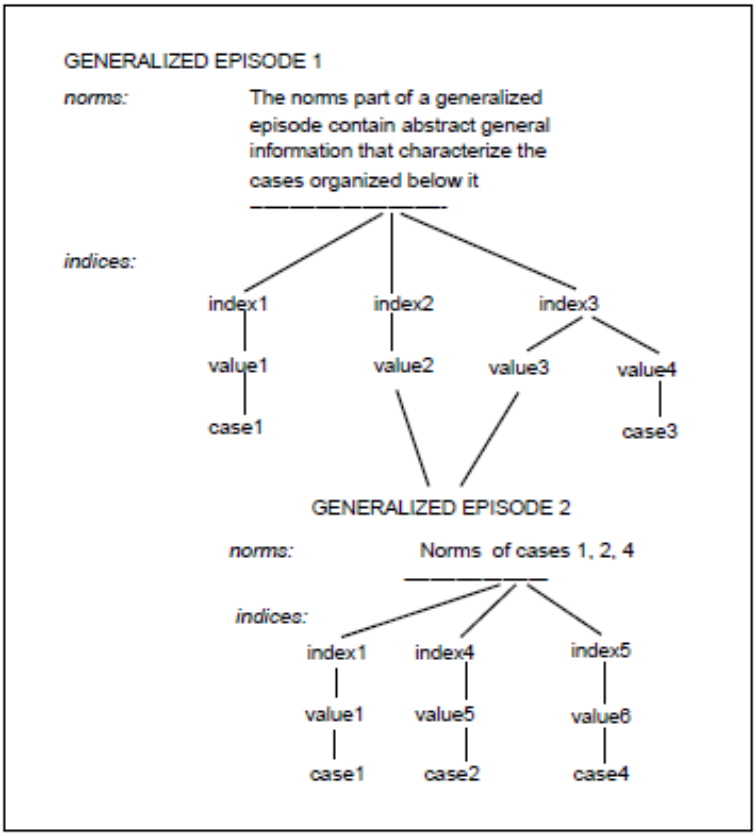


Abbildung 5.6: Generalized Episodes (GE) [1]

Die Knoten des Netzwerkes sind dabei entweder eine GE, welche die Normen enthält, ein Indexname, Indexwert oder ein Fall. Die Index-Wertpaare zeigen von einer GE zu einer andere GE oder zu einem Fall. Ein Indexwert hat entweder einen Zeiger auf einen einzelnen Fall, oder eine einzelne GE. Da es eine Vielzahl von Pfaden zu einem bestimmten Fall oder GE geben kann, ist dieses Schema der Indexierung redundant.

Der hier erstellte Fallspeicher kann bei Vorlage einer neuen Fallbeschreibung nach dem 'Best-Match' durchsucht werden. Dafür wird die neu eingegebene Fallstruktur von dem Wurzelknoten ausgehend die Netzwerkstruktur hinunter propagiert. Sofern ein oder mehrere Merkmale des Falles mit ein oder mehreren Merkmalen einer GE "matchen" kann, kann eine weitere Unterscheidung des Falles anhand der verbleibenden Merkmale vorgenommen werden. So kann der Fall mit den meisten gemeinsamen Merkmalen zum Fall der Eingabe aufgefunden werden.

Die beschriebene Suchprozedur wird sowohl zum Retrieval, als auch zur Speicherung der Fälle benutzt. Bei der Speicherung (engl.: storing) eines neuen Falles wird eine Unterscheidung durch Indexierung gemäß der unterschiedlichen Indizes unterhalb der GE vorgenommen. Aufgrund der dynamischen Eigenschaft der Speicherstruktur erfolgt nach dem Speichern die Reorganisation, wie sie ebenfalls aus dem allgemeinem Modell (siehe Abbildung 5.4) bekannt ist.
Nach der hier vorgestellten Methodik liegt die Indexstruktur als sogenanntes *'Discrimination Network'* vor, wobei ein Fall, genauer ein Zeiger zu einem Fall, unter dem jeweiligen Index, welcher ihn von anderen Fällen unterscheidet, gespeichert wird.[1] So kann eine Fallbasis organisiert und das Indizierungsschema von einer Vielzahl von Systemen, wie z. B. dem ersten Reasoner 'CYRUS'[22] verwendet werden. Dieses CBR-System basiert auf einer Erweiterung des Dynamic-Memory-Ansatzes und dient zur Speicherung und Erinnerung von Ereignissen. Eine Instanziierung des Systems ist in Abbildung 5.7 anhand des Beispiels "Cyrus" über die beruflichen Tätigkeiten des ehemaligen US-Außenministers Cyrus Vance zu sehen.

Ein Aufgreifen der allgemeinen Prinzipien des Dynamic-Memory-Ansatzes sind wie folgt abzuleiten. Das MOP in Abbildung 5.3 enthält die Normen: Handelnder, Teil-

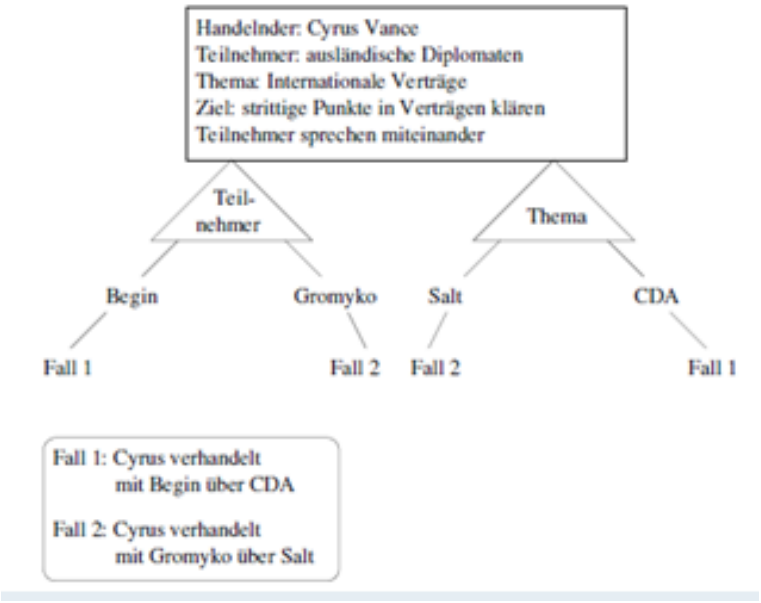


Abbildung 5.7: CYRUS-Speicherung eines Ereignisses "Diplomatisches Treffen" [1]

nehmer, Thema etc. Diese Merkmale werden durch Indizes bezüglich Teilnehmer und Thema spezialisiert. In Folge dessen können konkrete Indexwerte der Teilnehmer, wie z. B. "Begin" oder "Gromyko" angegeben werden. Diese Indexwerte zeigen, wie auch zuvor beschrieben, zu einem einzelnen Fall. Eine so erstellte Ausgangssituation liefert die in dem unterstem Kasten eingetragenen Fälle. Die dynamische Speicherstruktur kann mit weiteren Anfragen, die an das System gestellt werden, erweitert werden. Hierbei werden weitere Fälle und MOPs eigefügt.[39]
Das erste CBR-System CYRUS wird nicht in die Klasse der klassischen Systeme des Problemlösen durch Fallbasiertes Schließen klassifiziert, da eine Gliederung der Falldefinition in eine Problembeschreibung und dazu gehörige Problemlösung nicht erfolgt.

5.5 Retrieval durch Ähnlichkeitsbestimmung

Das Retrieval innerhalb des Prozesses des Fallbasierten Schließens ist der fundamentale Vorgang zum Erschließen einer Lösung in einem System. In dieser Phase des CBR-Zyklus geht es speziell um die Auswahl von geeigneten Fällen, so dass mindestens ein Fall zur Lösung des Problems beitragen wird. Das Prinzip ist dann

erfolgreich, wenn ein spezieller oder mehrere Fälle gefunden werden, die dem zu lösenden Fall am ähnlichsten sind.
Das Auffinden vorheriger Fälle kann anhand von oberfächlichen syntaktischen Ähnlichkeiten aus der Problembeschreibung erfolgen oder wird basierend auf Merkmalen (engl.: features) tiefgründiger und damit semantischer Ähnlichkeiten durchgeführt. Syntaktische Ähnlichkeitsabschätzungen werden auch als 'knowledge-poor', zu deutsch "wissensarme" Ansätze bezeichnet. Semantisch orientierte Ansätze sind bekannt als 'knowledge-intensive', gleichbedeutend mit "wissensintensiven" Ansätzen.
Zweiter Ansatz kann eine Verwendung von kontextuellen Bedeutungen der Problembeschreibung im Matching vornehmen, sofern allgemeines Domänenwisssen vorhanden ist.[1]
Die Ähnlichkeitsbestimmung ist eine wesentliche Aufgabe innerhalb des Retrievalschritts. Sie wird dazu eingesetzt, um in effizienter Weise angemessene Fälle zu finden. Beim Retrieval, dem "Abfragen", wird zu einem konkreten Problemfall die Menge der ähnlichsten Fälle aus der Fallbasis gesucht. Dieser Vorgang kann genauer beschrieben werden.

5.5.1 Retrieval Task

Der 'Retrieval Task' startet in aller Regel mit einer (partiellen) Problembeschreibung und schließt mit einem 'Best-Match' aus der Fallbasis. *Aaomodt* und *Plaza* nehmen bei ihrer Beschreibung des Retrievalprozesses[1] eine genauere Differenzierung vor. Demzufolge besteht das Retrieval aus mehreren Unteraufgaben (engl.: subtasks).

- **Identifikation der Features**

  Um ein gegenwärtiges Problem für das System verständlich zu machen, muss das Problem zuerst erkannt werden. In einfachster Weise kann dies durch Feststellung der 'Deskriptoren'[9] (engl.: descriptor) aus der Eingabe, die das Problem beschreiben, auch Indexterme[10] genannt, erfolgen.

[9] Auch "Schlagworte": die Sachverhalte zur inhaltlichen Erschließung festhalten.

[10] Aus dem IR: Term der Essenz des Inhalts eines Textes festhält und damit ein kontrolliertes Vokabular erstellt.

Ein aufwendigerer Ansatz versucht das Problem innerhalb seines Kontextes zu "verstehen". Dazu können Deskriptoren des Problems, die zu einem Verrauschen beitragen, gefiltert, weitere relevante Features des Problems erschlossen und Werte der Features auf den Sinn des Kontextes geprüft werden. Ebenfalls kann eine Erschließung von weiteren Deskriptoren, die nicht aus der Eingabe stammen, ermöglicht werden.
Am Ende der Identifikation liegt besten Falls eine Menge von relevanten Indextermen des Problems vor.

- **Initiales Matching**
  Die Durchführung der Aufgabe des Auffindens eines guten Match erfolgt in zwei Schritten. Der initiale Matchingprozess ruft eine Menge von plausiblen Kandidaten ab, bevor in einem komplexeren Schritt die Auswahl des Besten aus dieser Menge erfolgt. Die Auswahl findet im darauffolgenden Select Task statt.

Matching Um eine Menge von Fällen die "matchen" aufzufinden, werden die Deskriptoren des Problems, also die Features der Eingabe, als Indizes benutzt, um die Fallbasis in einer direkten, bzw. indirekten Weise zu referenzieren. Hierfür gibt es grundsätzlich drei prinzipielle Strategien, einen Fall oder eine Menge von Fällen aufzufinden. Ein Ermitteln kann erfolgen, durch Folgen direkter Indexzeigern der Problemmerkmale, durch Suchen in einer Indexstruktur (wie z. B. in der Dynamic Memory) oder in einem Modell von allgemeinen Domänenwissen.
Die Fälle werden also einzig aus den Eingabefeatures oder auch von Features die aus der Eingabe erschlossenen wurden, aufgefunden. Kandidaten für die Menge von Fällen, werden aufgrund von Übereinstimmungen mit den Features erstellt. Diese können auch anhand von Teilmengen der Problemmerkmale die eine Übereinstimmung aufweisen gebildet werden. Gefundene Fälle dieser Art können einem Test auf Relevanz unterzogen werden. Der Test benötigt eine Bewertung des Grades der Ähnlichkeit. Dazu gibt es verschiedenste 'Ähnlickeitsmetriken', die beispielsweise auf der oberflächlichen Ähnlichkeit von Problem

und Fall basieren. So kann auch eine wissensintensivere Bewertung vorgenommen werden, indem die Deskriptoren des Problems gemäß ihrer Wichtigkeit für die Charakterizierung des Problems gewichtet werden.
Ziel des Tasks ist die Rückgabe einer Menge von Fällen die dem neuen Fall hinreichend ähnlich sind.

- **Auswahl**
  Dieser Task arbeitet auf der Menge der ausgewählten ähnlichsten Fälle und soll den 'Best-Match' finden, der Prozess kann schon im initialen Matching vorbereitet werden. So wird der 'Best-Match' Fall durch genauere Auswertung des Grades des initialen Match bestimmt. Dies kann mit Hilfe von erstellten Begriffen zur Darlegungen der Ungleichheit von Features anhand des Wissens im Semantischen Netzwerkes[11] durchgeführt werden. Typischerweise generiert der Auswahlprozess dabei Hilfskonstrukte zur Abschätzung wie: Konsequenzen, die auszuwerten sind, und Erwartungen, die zu erfüllen sind. Diese könne zu jedem der aufgefundenen Fälle angegeben werden. Die beiden möglichen Kriterien werden unter Verwendung des systemeigenen allgemeinen Domänenwissen abgeschätzt. So können die Fälle gemäß einer Ähnlichkeitsmetrik oder eines Ranking-Kriteriums "gerankt" werden. Wissensintensive Auswahlmethoden generieren Darlegungen, die diesen Rankingprozess unterstützen, und der Fall, mit der stärksten Darlegung bezüglich einer Ähnlichkeit zum neuen Problem wird ausgewählt.

Die hier vorgenommene Unterteilung des Retrievalprozesses in mehrere Subtasks ist eine zunächst rein theoretische Beschreibung - der Ablauf des Prozesses und eingesetzte Methodiken können sich je nach Anwendungsgebiet unterscheiden.
Wie im Vorgang des Matching und der anschließenden Auswahl der ähnlichsten Fälle zu erkennen ist, ist ein wichtiger Aspekt des Retrievalvorgangs die Ähnlichkeitsbestimmung zwischen den Features, die eine neues Problem ausweisen und den Features, die Features der Fälle in der Fallbasis beschreiben. Allgemein kann die Bestim-

[11] Ein formales Modell zur Wissensrepräsentation, bestehend aus Begriffen und ihren Relationen zueinander.

mung zur Ähnlichkeit von Fällen auf verschiedenen Abstraktionsebenen vorgenommen werden und so gibt es die unterschiedlichsten Methoden, die in Abhängigkeit der Fallrepräsentation per Algorithmus die Berechnung der Ähnlichkeit durchführen.

5.5.2 Ähnlichkeitsbestimmung

Die Bestimmung der Ähnlichkeiten von Fällen ist ein wichtiger Aspekt bei der Realisierung eines CBR-Systems. Damit direkt verbunden ist die Frage nach der Modellierung und Formalisierung eines Ähnlichkeitsbegriffes, der es also ermöglicht, Ähnlichkeiten zwischen einzelnen Fällen (formal) auszudrücken.

Maße Um ein Maß zur Abschätzung zu erhalten, können zum einen einfache, generell einsetzbare Abstandsmaße wie Hamming[12]- oder Euklidabstand[13] eingesetzt werden. Zum anderen gibt es wissensintensive Ähnlichkeitsmaße, die bei der Bestimmung von Ähnlichkeiten der Merkmale Verwendung finden. Diese sind auf spezifische Anwendungsdomänen spezialisiert und haben so den Vorteil, dass zusätzliches Hintergrundwissen einbezogen werden kann[38].

Eine Vielzahl der Retrievalverfahren basieren auf der grundsätzlichen Annahme, dass ähnlichkeitsbasiertes Retrieval, bei dem ähnlichste Fälle gefunden werden, den größten Nutzen zur Lösung des Zielproblems liefern. Diesbezüglich kann der Begriff Ähnlichkeit, bei spezifischen Anwendungen mit dem Begriff Nützlichkeit für eine Problemlösung angenähert werden.
Die Maße, welche zur Berechnung der Ähnlichkeit eingesetzt werden, können auf verschiedenen Formalismen aufgrund ihrer unterschiedlichen Repräsentationen basieren. Dazu muss eine Einstufung, auf welcher Ebene eine Bestimmung der Ähnlichkeit stattfindet, vorgenommen, und die Wahl der Repräsentation der Fälle berücksichtigt werden.
Eine zentrale Frage ist also, wie eine Bewertung von Ähnlichkeiten umgesetzt werden kann. Ein Großteil der Anwendungen im Fallbasierten Schließen beurteilen Ähnlichkeiten hinsichtlich ihrer "Oberflächen Features" (engl.: surface features). Diese sind

[12] Bestimmung der Unterschiedlichkeit von Zeichenketten.
[13] Eukl. Abstand zweier Punkte in der Ebene bzw. Raum

in einem Fall diejenigen Features, die direkt aus ihrer Fallbeschreibung bereitgestellt werden können. Typischerweise können dies Repräsentationen der Attribut-Wert-Paare (Unterabschnitt 5.5.2) sein.[12]

Bewertung von 'Surface'-Ähnlichkeiten

Bei den Ansätzen die Retrieval, basierend auf 'Surface Features', durchführen, kann die Ähnlichkeit zwischen Fällen typischerweise als eine reale Zahl im Intervall zwischen $[0, 1]$ repräsentiert werden. Die Berechnung erfolgt gemäß eines gegebenen Ähnlichkeitsmaßes. Generell sind die aus dem Retrievalschritt stammenden Fälle, die "k ähnlichsten" zum gestellten Problem. So wird dieser Ansatz als 'k nearest neighbor'[11] Retrieval bezeichnet.[12] Die hier eingesetzte Methode ist das wohl verbreitetste Berechnungsverfahren im Retrievalschritt eines CBR-Systems.

'Nearest Neighbor' Mit Hilfe von 'Nearest-Neighbor-Algorithmen' können Bewertungen der Ähnlichkeiten berechnet werden. In einer Verallgemeinerung des Prinzips der Ähnlichkeitsbestimmung mit 'Nearest-Neighbor-Verfahren', kann in einer einfachen Gleichung[22], die die *Ähnlichkeit von zwei Fällen* ausdrückt, angegeben werden:

$$Similarity(C, Q) = \sum_{i=1}^{n} f(C_i, Q_i) \times w_i$$

Dabei ist C ein Fall aus der Fallbasis, Q der aktuelle Fall, also die Anfrage und n die Anzahl der Features eines Falles. Mit der Funktion f kann eine spezifische Ähnlichkeitsfunktion für das Feature i definiert werden. Ein Feature i kann durch ein Gewicht w gemäß der Wichtigkeit innerhalb des Falles gewichtet werden. Das Ergebnis der Berechnung liegt in dem besagten Intervall und ist damit ein typisches Maß für die Ähnlichkeit zwischen Fällen der Fallbasis und der Anfrage an das System. Die Features eines Falles können auch als Attribute bezeichnet werden.

Attribut-Wert-Repräsentation Es gibt eine Vielzahl von Möglichkeiten der Abschätzung von Ähnlichkeiten, die unterschiedlichen Ansätze werden ihrer unterschiedlichen Fallrepräsentationen entsprechend eingesetzt. So ist ein weiteres, häufig angewandtes Prinzip, die Repräsentation der Fälle als einfacher Feature-Vektor oder

Menge von Attribut-Wert-Paaren. Hierfür wird ein lokales Ähnlichkeitsmaß definiert, jedes Attribut zu bewerten, um eine globale Bewertung berechnen zu können. Es resultiert daraus der gewichtete Durchschnitt der lokalen Ähnlichkeiten. Nach dieser Annahme lässt sich eine allgemeine *Formel zur Ähnlichkeitsbestimmung zweier Fälle* [22] definieren:

$$SIM(C,Q) = \frac{\sum_{i=1}^{n} w_i sim_i(C,Q)}{\sum_{i=1}^{n} w_i}$$

Die beiden Fälle C, Q werden wie in der ersten Gleichung bezeichnet als: Ein Fall C aus der Fallbasis und ein Fall Q aus der Anfrage. Das lokale Ähnlichkeitsmaß entspricht einer lokalen Ähnlichkeitsfunktion sim_i, die die Attributähnlichkeit berechnet. Mit Hilfe der Gleichung können also Objekte gleicher *Attribut-Wert-Paare* verglichen werden.

Einige Anwendungen im CBR verlangen das Ableiten von Features aus den Beschreibungen der Fälle durch Inferenz mit Domänenwissen.
Sofern Fälle durch komplexe Strukturen, wie z. B. Graphen, repräsentiert werden, benötigt das Retrieval eine Bewertung der strukturellen Ähnlichkeiten.
Ähnlichkeitsbewertungen von abgeleiteten Features und von Strukturen sind sehr aufwändig in der Berechnung. Eine Möglichkeit zu garantieren, dass ohne hohen Berechnungsaufwand Fälle gefunden werden, ist durch ein 'Index-Vokabular' (engl.: indexing vokabular) gegeben. Das Index-Vokabular hält mit einer expliziten Beschreibung des Falles die Features fest, die dessen Relevanz bestimmen. Der Ansatz des Index-Vokabulars basiert auf der Dynamic-Memory-These, die im Abschnitt 5.4 der Fallrepräsentation und deren Organisation behandelt wurden.
Im Rahmen dieser Arbeit, wird ausschließlich eine Betrachtung der Ähnlichkeiten auf Fallbasen in Anwendungsebene [39] vorgenommen. Eine Ähnlichkeitsfunktion bestimmt ein Maß, welches sich auf die in der Anwendung existierende Ähnlichkeit begründen lässt. In der betrachteten Ebene werden Dokumente also ähnlich genannt, wenn sie gleiche Produktbeschreibungen beinhalten.

5.5.3 Retrieval-Verfahren

Bei der Bestimmung der Fälle aus der Fallbasis können folgende elementare Retrieval-Verfahren [39] benannt werden. Die Verfahren können universell eingesetzt werden:

- Sequentielles Retrieval: Jeder Fall der Fallbasis wird einzeln mit der Anfrage verglichen
- Relationales Retrieval: Vorauswahl und Retrieval auf der Vorauswahl
- Indexorientiertes Retrieval: Retrieval mit Hilfe einer Indexstruktur.

Prinzipiell wird beim sequentiellen Retrieval die gesamte Fallbasis sequentiell mit der Anfrage verglichen. Somit können die Fälle, die in Frage kommen, ihrer Ähnlichkeit entsprechend in einer Liste abgespeichert werden. Das sequentielle Verfahren kann per Pseudocode beschrieben werden und kann auf folgender Datenstruktur[39] basieren.

Sequentielles Retrieval: Datenstruktur

Datenstruktur[39]

```
1  TYPE
2      CaseType              =...
3      SimilarityType        =...
4      SimCase               =RECORD
5              case          =CaseType
6              similarity    =SimilarityType
7      END
8
9  VAR
10     SimCaseQueue          =ARRAY [1..m] of SimCase
11     QueryCase             =CaseType
12     Case                  =CaseType
13     CaseBase              =ARRAY of Case
```

Eine konkrete Anwendung gibt *CaseType* und *SimilarityType* eine genaue Definition. Damit kann der Formalismus der Fallrepräsentation und die Art der Ähnlichkeitsberechnung angegeben werden. Mit der Datenstruktur SimCase kann ein Eintrag

erzeugt werden, der aus einem konkreten Fall und einer Ähnlichkeit zu einem anderen Fall, einer Anfrage Q, besteht. Diese Datenstruktur wird für SimCaseQueue verwendet. Hierbei können die m ähnlichsten Fälle und die Ähnlichkeit zu Q gespeichert werden. Die Queue wird in einem Array abgelegt. Schließlich werden die "m ähnlichsten Fälle" entsprechend ihrer Ähnlichkeit zur Anfrage Q in SimCaseQueue gespeichert.
Ein Algorithmus[39], der auf der hier angegebenen Datenstruktur arbeitet, kann ebenfalls angegeben werden.

Sequentielles Retrieval: Algorithmus

Algorithmus[39]

```
1  PROCEDURE SearchSeq (CaseBase, QueryCase, m) :SimCaseQueue
2  VAR i INT
3  BEGIN
4          SimCaseQueue[1..m].similarity := 0
5          FOR i := 1 TO n DO
6          IF SIMImp (QueryCase, CaseBase[i] > SimCaseQueue[m].similarity
7  SimCaseQueue := SimCaseQueue + CaseBase[i]
8          END
9          RETURN (SimCaseQueue)
10 END
```

Im Algorithmus wird jeder Fall der Fallbasis mit der Anfrage Q, also dem QueryCase, verglichen. Falls der aktuell betrachtete Fall ähnlicher ist, als der bisher am wenigsten ähnliche Fall (SimCaseQueue[m]) in SimCaseQueue, so wird dieser Fall in das Array eingefügt und der letzte Fall, der damit am wenigsten Ähnlichkeit aufweist, aus dem Array geschoben. Zur Vereinfachung kann angenommen werden, dass die Operation "+" automatisch für eine Einordnung der Fälle in das Array der bisher gefundenen Fälle sorgt. Diese Einordnung kann durch einen beliebigen Sortier- oder Einfügealgorithmus durchgeführt werden.
Die Performanz des Algorithmus ist stark abhängig von der Ähnlichkeitsfunktion, im Pseudcode mit SIM_{Imp} angegeben, da jeder Fall der Fallbasis mit der Anfrage verglichen und wenn nötig in die Queue eingeordnet werden muss. Die Wahl einer

der Anwendung geeigneten Funktion zur Berechnung der Ähnlichkeit ist im vorangegangenen Abschnitt zu entnehmen.

5.6 CBR Anwendung: Textuelles Case-Based Reasoning

In dieser neueren Anwendungsrichtung des Fallbasierten Schließens, geht es um den Umgang und die Verarbeitung zur Extraktion von Informationen aus textuellen Dokumenten. Diese können im Allgemeinen in natürlichsprachlicher Form vorliegen oder auch beliebiger Herkunft sein, wie: Reporte, Dokumentationen, FAQs oder informeller Notizen und Kommentare zu Sachverhalten. Die Bewältigung von Problemstellungen textueller Art ist ursprünglich eine Disziplin des typischen Information Retrieval (IR) mit den klassischen Modellen wie: 'Vector Space Model', 'Probabilistic Model' und 'Inference Network Model'.[26][28]

Einfluss des IR Komponenten aus Ansätzen des 'IR' finden maßgeblich Verwendung in der Anwendung des textuellen CBR. Ein großer Vorteil des textuellen CBR ist, dass im Vergleich zum IR konkrete Fakten zum Lösen einer Anfrage, die bestimmte Informationen zurückgeben soll, geliefert werden können. Dies steht im Gegensatz zur einfachen Auskunft über die Existenz des Angefragten, wie es im Information Retrieval übich ist.[27]
Im Information Retrieval gibt es nur eine recht schwache Anforderung, um ein geeignetes System erstellen zu können. Diese ist, dass eine Ansammlung von Dokumenten verfügbar ist, aus der eine Fallbasis erstellt werden kann. So können Techniken des 'IR' in theoretisch jeder Domäne angewandt werden.
Diese Anforderung gilt ebenfalls für das textuelle CBR. Zusätzlich muss die Verfügung von Wissen gewährleistet werden, um den wissensbasierten Ansatz des textuellen Schlußfolgerns umzusetzen. Dabei ist dieses Wissen von domänenspezifischer Art, und Informationen daraus gehen z.B. direkt in die Erstellung des Indexvokabulars oder der Ähnlichkeitsabschätzung ein. Bei diesem Prozess der Wissensakquisition ist ein Domänenexperte notwendig. Die Anforderungen, die an ein textuelles System gestellt werden, haben zur Folge, dass eine textuelle CBR Anwendung für eine spezifische Domäne erstellt wird, so sind implementierte Systeme nur sehr schwer in

andere Problemdomänen zu übertragen.

Datenrepräsentation Wie auch schon in den Überlegungen zur Fallrepräsentation muss auch für ein textuelles Dokument zunächst eine möglichst geeignete interne Repräsentation der Daten gefunden werden. Darin enthaltene Entitäten können damit als Informationen aufbereitet werden, um auch eine Benutzung weiterer Hilfskonstrukte wie 'Indexterme'[14] zu ermöglichen. Im spezifischen Prozess zur Fallerstelllung (engl.: case authoring process) müssen die Fälle, wie sie in der Problemdomäne vorkommen, encodiert werden, um dem im System eingesetzten Formalismus gerecht zu werden. Dies impliziert ebenfalls eine Festlegung auf Fallrepräsentationen und Umsetzung der Wissensakquisition.

Dokumenteigenschaften In allen Verarbeitungsansätzen textueller Dokumente, also auch für Anwendungen des textuellen CBR, sind zunächst einmal die unterschiedlichen Eigenschaften der zugrundeliegenden Dokumente zu beachten. Diese sind: Struktur (voll-, semi-, unstrukturiert), Sprache (natürliche Texte aus versschiedenen Sprachen oder aus Fachwörtern) und Eindeutigkeit (Ambiguity Problem[15] und Paraphrase Problem[16]) der Dokumente. Ein gutes Verarbeitungssystem erstellt eine sinnvolle Strategie, die sich der hier genannten Probleme annimmt.[27]
Das Wissen eines Systems kann gemäß des Containermodells (Abschnitt 5.2) modelliert werden und zusätzliches Wissen, wie beispielsweise bei der Erstellung von Ähnlichkeitsmaßen, kann in Form eines semantischen Ähnlichkeitsmaßes eingebettet werden. Ein großer Vorteil des textuellen CBR Systems ist seine Auslegung auf eine ganz spezifische Domäne. Eng zusammenhängend damit ist die Durchführung einer ersten Akquisition des Wissens, um wichtige Merkmale (engl.: features) der Domäne zu identifizieren und ein 'Termdictionary' aus diesen zu konstruieren. Die Features können basierend auf Schlüsselworten, Termen oder Ausdrücken erkannt werden. Sie bestehen damit aus bestimmten Worten oder Ausdrücken, die in den Dokumenten bezüglich der betrachteten Domäne häufig vorkommen. Diese Tatsache ermöglicht es

[14] siehe auch Indexvokabular

[15] Wörter können mehrere Bedeutungen haben.

[16] Sachverhalte können durch unterschiedliche Wörter ausgedrückt werden.

außerdem, einen ersten Schritt in Richtung Repräsentation des Textes für das System zu vollziehen.

Schlüsselworte eines Dokuments Um also Fälle zu erhalten, von denen aus gegebenen Dokumentsammlungen gefolgert wird, ist es erforderlich, Fälle aus vorhandenen Textdokumenten zu extrahieren. Am einfachsten ist dabei die Benutzung von Schlüsselworten, die in einer Domäne spezifisch sind und damit ein geeignetes *Indexvokabular* aufbauen können. So kann in einer groben Abschätzung der Inhalt der Dokumente abgebildet werden. Eine aussagekräftigere Repräsentation der textuellen Dokumente kann mit Techniken des 'Natural Language Processing' erreicht werden. Hier kommen bekannte Tools wie 'Part-of-Speech Tagger' zum Einsatz, die natürlichsprachliche Texte in üblicher Art und Weise, wie z. B. durch 'Stemming', aufbereiten und als anspruchsvolle Repräsenation zu Verfügung stehen.

Zur eigentlichen Informationsextraktion können auch weitere Repräsentationsformen ausgenutzt werden. Ein Großteil der Inhalte eines Dokuments liegt in Klartext (engl.: plain text) vor. Dokumente, vor allem aus technischen Domänen, beinhalten zusätzlich strukturiertere Informationen. Dies sind Ausdrücke der Form eines 'Attribut-Wert-Paares' (engl.: attribut-value pair), wie "12 Volt" oder auch "Pentium IV", also Terme von Kombinationen numerischer und alphabetischer Zeichen. Für diese Informationen können sogenannte 'Trigger' definiert werden, die diese Art von Repräsentationen auffinden und die Informationen, die mit diesem Trigger assoziiert werden, extrahieren. Die "Buchstaben-Wert-Paare" werden dabei als Features des korrespondierenden Falles aufgefasst.

Es können für das textuelle CBR explizit zwei größere Teilaufgaben im Prozess der Verarbeitung angegeben werden. So ist eine erste Ebene die der Fallrepräsentation und Strukturierung der zu parsenden Daten, die zweite Ebene ist die eigentliche Verarbeitung, das Retrieval auf Grund von Ähnlichkeitsbestimmungen.
Fallstrukturen und *Fallrepräsentationen* können im textuellen CBR mit folgender Erklärung dargestellt werden. Um an die in den textuellen Dokumenten enthaltenen Informationen zu gelangen, wird zunächst jedes Dokument *geparst*, um dieses in einen

sachgemäß strukturierten Fall unter Verwendung des gegebenen Lexikons aus Informationsentitäten (IE) zu überführen. Der hier beschriebene Prozess des Parsing[27] beinhaltet:

- Wissen über die Struktur der Dokumente
- Mapping von Texten zu Mengen von Informationsentitäten
- Identifikation elementarer linguistischer Strukturen in den Dokumenten durch oberflächliche NLP-Techniken wie Part-of-speech Tagging
- Anwendung einfacher Informationsextraktionstechniken um strukturierte Einheiten wie Attribut-Wert-Paare aufzufinden

Informationsentitäten Wie zu sehen ist, sind die *Informationsentitäten* beim textuellen CBR ein wichtiges Mittel zur Repräsentation der Inhalte von Textdokumenten. Sie können wie eine Art Symbol angesehen werden, das eine spezifische Bedeutung in der angewandten Domäne repräsentiert. So kann eine einzelne Informationsentität für bestimmte Schlüsselworte oder Satzteile stehen, die grammatikalische Variationen, Abkürzungen und sogar fremdsprachliche Ausdrücke mit einschließen. Die Menge aller Informationsentitäten kann ferner gemäß spezifischer Typen beschrieben werden:

- Schlüsselwörter
- domänenspezifischer Ausdrücke
- speziellen Codes und Nummern
- Attribut-Wert-Paare und
- weiteren Attributen die zur Modellierung der Domäne benötigt werden.[27]

Der bisher beschriebene Prozess kann automatisiert durchgeführt werden und es ist keine manuelle Fallerstellung nötig.

Der nächste Schritt im Gesamtprozess bezieht sich auf die Ähnlichkeitsermittlung

und das eigentliche Retrieval. Sobald die Dokumente als Fälle konvertiert vorliegen, kann in ähnlicher Weise mit den Anfragen (engl.: query) umgegangen werden. Ein entscheidender Punkt ist dabei die Berechnung der Ähnlichkeit. Die Ähnlichkeit von Anfragen und Fällen wird auf der Ähnlichkeit der konstituierenden Informationsentitäten berechnet. So kann die Ähnlichkeit eines Falles C zu einer Anfrage Q wie folgt definiert[27] werden:

$$SIM(Q,C) = \sum_{e_i \in Q} \sum_{e_j \in C} sim(e_i, e_j)$$

Dabei sind e_i und e_j die Informationsentitäten, welche die Anfrage Q und den Fall C entsprechend repräsentieren. Damit spezifiziert die Funktion *sim* die *lokale* Ähnlichkeit von zwei Informationsentitäten.

Um deutlicher aufzuzeigen, wie die verschiedenen Typen von IEs zur Fallähnlichkeit beitragen, kann die Komposition der hier angegebenen Ähnlichkeitsfunktion genauer betrachtet werden. Eine Zusammenstellung wird in Beachtung der Typen von Informationsentitäten vorgenommen, dies bedeutet, dass ein allumfassendes (allgemeines) Ähnlichkeitmaß die Ähnlichkeit berücksichtigt. Dies basiert auf:

- allgemeinen IEs;
- relationalen (ähnlichen) IEs;
- Attribut Werten, wie angegeben in den Dokumenten;
- Attribut Werten, entnommen während der automatisierten Informationsextraktion.

Mögliche Komponenten der Ähnlichkeitsfunktion sind in aufsteigender Wissensintensitivität aufgelistet und tragen dementsprechend zur allgemeinen Performanz des Systems bei, da der Retrievalprozess die IEs der Fälle und deren Ähnlichkeitsbeziehungen in direktem Maße in Anspruch nimmt. So ist es naheliegend, dass entsprechend wissensintensivere Maße zu einer Verbesserung der Qualität des Systems beitragen.[27]

Wie bei allen Systemen, die sich aufgrund ihrer Anwendungsart in die Klasse des Information Retrieval einordnen lassen, kann eine Auswertung der Ergebnisse oder Bewertung der Leistung, die das System in der Lage ist zu erbringen, erfolgen.

Performanzmaße Merkmale zur Abschätzung der Performanz des allgemeinen CBR, als auch des textuellen CBR, sind die Maße 'Precision' und 'Recall.' Wobei in der Anwendung des CBR bei der *Precision* gemessen werden kann, wie viel Prozent der Fälle, die eine Antwort auf aktuelles Problem liefern, relevant sind. Der *Recall* gibt an, wie viel Prozent aller relevanten Fälle in der Antwort angegeben wurden.

KAPITEL

SECHS

Konzeptionelles Design: HMM

6.1 Einleitung

Die Problemstellung dieser Ausarbeitung besteht darin bestimmte Daten, in diesem Fall Produktbeschreibungen aus den vorliegenden Rechnungen automatisch zu extrahieren. Diese Aufgabe wird mit dem im Abschnitt 4.1 vorgestellten Hidden-Markov-Modell bewerkstelligt werden. Bei den Rechnungen handelt es sich nicht um unstrukturierte, natürlichsprachliche Texte , sondern um semistrukturierte Texte aus den Produktbeschreibungen extrahiert werden sollen. Weshalb das in Kapitel 4 vorgestellte Hidden-Markov-Modell für diesen speziellen Fall angepasst werden muss. Die Anpassung sieht folgendermaßen aus:

1. Alle Dokumente in ein Dokumentenkorpus zusammenfassen.
2. eingescannte Dokumente mittels OCR-Software verarbeiten, um den Dokumententext zu extrahieren.
3. Mittels Hidden-Markov-Modell oder Case-Based-Reasoning-System Informationsextraktion (IE) von Produktbeschreibungen aus den Dokumenten durchführen.

In diesem Kapitel wird der dritte Punkt näher betrachtet, also die Extraktion der Produktbeschreibungen aus den Rechnungsdokumenten. Es werden zwei unterschiedliche Systeme angewandt, um die Extraxtion der Daten aus den Rechnungen zu

gewährleisten. Als erstes wird ein Hidden-Markov-Modell, wie im Abschnitt 4.1 beschrieben benutzt, um die Produktbeschreibungen aus den Rechnungen zu extrahieren. Danach kommt ein Case-Based-Reasoning-System zum Einsatz um ebenfalls die Informationen zu den Produkten aus den Rechnungen zu extrahieren.
Die folgenden Abschnitte beschreiben die Vorgehensweise, um mit einem Hidden-Markov-Modell die gewünschten Informationen aus den Rechnungen zu extrahieren.

6.2 Analyse des Dokumentenkorpus

Bei den Quelldokumenten handelt es sich um Eingangsrechnungen und Katalogdaten, welche als Bilder und im Text Format vorliegen. Der Text dieser Rechnungen wird durch eine Optical Character Recognition (OCR) Software extrahiert und liegt danach als semi-strukturierter Text vor. Da es jedoch bei vielen Dokumenten vorkam, dass es zu OCR Fehlern kam, waren viele Rechnungsdokumente und Kataloge inkonsistent. In dieser Arbeit wird davon ausgegangen, dass die von der OCR Software extrahierten Rechnungen Konsistent sind und deshalb wurden die Rechnungen und Kataloge nach der Extrahierung manuell korrigiert, um von einem Idealen Zustand auszugehen in der sich die Rechnungsdokumente befinden sollten.
In den folgenden Abschnitten werden erst die Eigenschaften des Dokumentenkorpus beschrieben und darauf folgend die zu extrahierenden Rechnungsdaten analysiert.

6.2.1 Dokumentenkorpus

Der Dokumentenkorpus, welcher von der Firma Schottler GmbH bereitgestellt wurde, beinhaltet 45 Dokumente, bei denen es sich ausschließlich um Rechnungen handelt. Diese Rechnungen liegen als eingescannte Bilder im JPG-Format oder als Textdateien vor, welche durch ein Texterkennungsprogramm der Schottler GmbH generiert wurden. Alle 45 Rechnungen stammen aus Deutschland. Innerhalb der 45 Rechnungen gibt es im wesentlichen immer den gleichen Rechnungssteller.

6.2.2 Rechnungsdaten

Es handelt sich bei den Dokumenten um Rechnungen von demselben Rechnungssteller und somit weisen sie dieselben strukturellen Merkmale auf. Es existieren also allgemeine Eigenschaften und Muster, nach denen die Rechnungsdaten identifiziert werden können. In den folgenden Abschnitten werden diese Eigenschaften und Muster der Rechnungsdaten, welche in dieser Arbeit extrahiert werden sollen, vorgestellt.

Rechnungsdatum Das Rechnungsdatum kennzeichnet das Datum, an dem die Rechnung ausgestellt wurde.

Ein Datum in einem öffentlichen Dokument besitzt, regional bedingt, verschiedene Formate. Zum Beispiel ist das typische Datumsformat im deutschsprachigem Raum Tag.Monat.Jahr.In einer Rechnung können noch andere Daten stehen, zum Beispiel ein Datum, wann die Rechnung bezahlt werden muss oder wann eine bestimmte Ware geliefert wurde. Die Position, an welcher das Datum in der Rechnung steht, kann ein Indiz dafür sein, ob es sich um ein Rechnungsdatum handelt oder nicht. Bei der Durchsicht der Trainingsdokumente wurde festgestellt, dass das Rechnungsdatum meist das erste Auftreten eines Datums in einer Rechnung ist, da es typischerweise im Rechnungskopf steht.

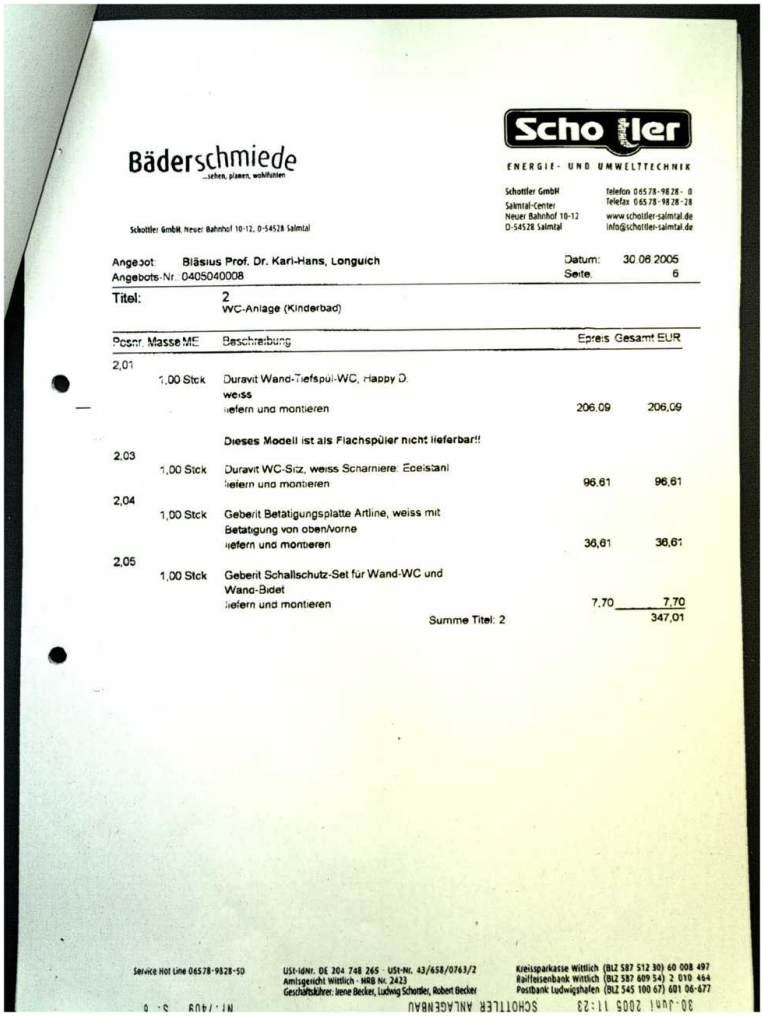
Positionsnummer Alle Rechnungen von der Firma Schottler GmbH besitzen die Eigenschaft, dass nach dem Titel gleich danach die Positionsnummer angegeben wird. Diese ist eine zwei stellige Gleitkommazahl und gibt an, an welcher Position sich das Produkt auf der Rechnung befindet.

Maßeinheit Hier ist die Menge angegeben wie viel Produkte bestellt werden sollen. Mögliche Werte sind zum Beispiel "Stück","Satz" oder "Meter".

Beschreibung Nach der Angabe der Maßeinheit folgen in den Rechnungen jeweils die Produktbeschreibungen. Darin wird Angegeben welches Produkt bestellt werden soll und welche zusätzlichen Eigenschaften es besitzen soll (Eigenschaften wie z. B. Länge, Breite oder Höhe). In der Arbeit geht es darum, eben diese Produktbeschreibungen zu extrahieren. Da vor jeder Produktbeschreibung eine Mengenangabe steht, kann man dies als ein Indiz aufnehmen, dass auf eine Produktbeschreibung hindeutet.

Endpreis und Gesamtpreis Diese zwei Zahlenwerte stehen rechts neben der Produktbeschreibung und signalisieren das Ende einer Produktbeschreibung. Leider ist das nicht ein 100 prozentiges Indiz dafür, dass das Ende einer Produktbeschreibung erreicht worden ist, da es nicht in allen Rechnungen vorkommt, das der Endpreis und Gesamtpreis neben der Produktbeschreibung vorhanden ist.

Auf folgender Abbildung 6.1 ist ein Rechnungsdokument aus dem Dokumentenkorpus zu sehen.



Bäderschmiede
...sehen, planen, wohlfühlen

Schottler GmbH, Neuer Bahnhof 10-12, D-54528 Salmtal

Schottler
ENERGIE- UND UMWELTTECHNIK

Schottler GmbH
Salmtal-Center
Neuer Bahnhof 10-12
D-54528 Salmtal

Telefon 06578-9828-0
Telefax 06578-9828-28
www.schottler-salmtal.de
info@schottler-salmtal.de

Angebot: Bläsius Prof. Dr. Karl-Hans, Longuich
Angebots-Nr.: 0405040008

Datum: 30.06.2005
Seite: 6

Titel: 2
WC-Anlage (Kinderbad)

Posnr.	Masse	ME	Beschreibung	Epreis	Gesamt EUR
2,01	1,00	Stck	Duravit Wand-Tiefspül-WC, Happy D. weiss liefern und montieren	206,09	206,09
			Dieses Modell ist als Flachspüler nicht lieferbar!!		
2,03	1,00	Stck	Duravit WC-Sitz, weiss Scharniere: Edelstahl liefern und montieren	96,61	96,61
2,04	1,00	Stck	Geberit Betätigungsplatte Artline, weiss mit Betätigung von oben/vorne liefern und montieren	36,61	36,61
2,05	1,00	Stck	Geberit Schallschutz-Set für Wand-WC und Wand-Bidet liefern und montieren	7,70	7,70
			Summe Titel: 2		347,01

Service Hot Line 06578-9828-50
USt-IdNr. DE 204 748 265 · USt-Nr. 43/658/0763/2
Amtsgericht Wittlich · HRB Nr. 2423
Geschäftsführer: Irene Becker, Ludwig Schottler, Robert Becker
Kreissparkasse Wittlich (BLZ 587 512 30) 60 008 497
Raiffeisenbank Wittlich (BLZ 587 609 54) 2 010 464
Postbank Ludwigshafen (BLZ 545 100 67) 601 06-677

30. Juni 2005 11:23 SCHOTTLER ANLAGENBAU NT-7409 S. 0

Abbildung 6.1: Eine Rechnung der Firma Schottler GmbH

6.2.3 Katalogdaten

Es handelt sich bei den Dokumenten um Katalogen, die teilweise dieselben strukturellen Merkmale aufweisen. Es existieren also allgemeine Eigenschaften und Muster, nach denen die Katalogdaten identifiziert werden können. Ein bekanntes Merkmal bei Katalogen ist es, dass die Daten tabellarisch angeordnet sind und dadurch sich eine Struktur ergibt die es erlaubt gezielt Daten zu extrahieren. Die Kataloge beinhalten lediglich technische Details über diverse Produkte der Bauwesendomäne. Auch hier wird auf die Eigenschaften und Muster eingegangen, wodurch sich eine Extraktion der benötigten Informationen aus den Katalogen ermöglicht wird.

Produktbezeichnung Alle Kataloge beginnen am Anfang mit einer Produktbezeichnung. Meist definiert die Produktbezeichnung die Kategorie des Produktes. Ob es sich Zum Beispiel bei einem Katalog generell um ein Spiegel Handelt, ohne jedoch auf Details einzugehen.

Modell-Nr In der Tabelle stehen die Modell-Nr über den Attributen des Produktes. Das heißt, dass die Modell-Nr eines Produktes eine Spalte in der Tabelle bildet. Der Bezeichner "Modell-Nr" steht auf der linken Seite der Tabelle über den Bezeichnern der Attribute eines Produktes. Was zum Beispiel ein Indiz ist, dass in dieser Zeile der Tabelle Modell Nummern zu einem bestimmten Produkt zu erwarten sind.

Maßeinheit Wie schon erwähnt steht unter einer Modell-Nummer jeweils Maßeinheiten zu einem Produkt mit einer bestimmten Modell Nummer. Die Bezeichner der Maßeinheit befinden sich unter der Modell Nummer Bezeichnung in der Tabelle. Dort stehen dann Bezeichner wie zum Beispiel "Äußere Länge", "Äußere Breite", "Äußere Höhe" usw. . Für diese Arbeit ist es wichtig, die Attribute, die neben diesen Attribut Bezeichnern stehen zu extrahieren.

Auf folgender Abbildung 6.2 ist ein Produkt aus einem Katalog zu sehen.

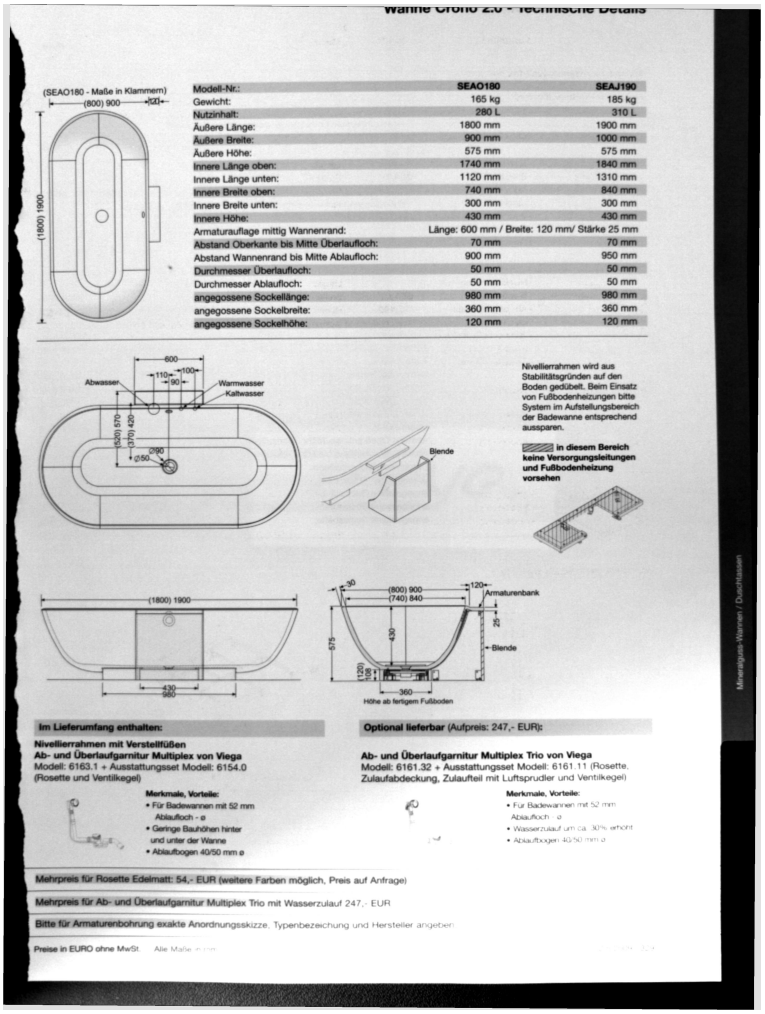


Abbildung 6.2: Ein Produkt aus einem Katalog

6.2.4 Zusammenfassung

Da diese Trainingsmenge nur deutschsprachige Rechnungen und nur ein Rechnungshersteller umfasst, ist die gegebene Trainingsmenge recht homogen. Deswegen können für die weiteren Experimente der Informationsextraktion aus diesen Rechnungskorpus nicht davon ausgegangen werden, dass das System auf einer größeren heterogenen Dokumentensammlung dieselben Resultate erzielt werden können, wie bei den vorhandenen Rechnungsdokumenten.

Außerdem werden die beiden Systeme, die noch vorgestellt und evaluiert werden, nur

eine bestimmte Domäne abdecken können (und zwar die Bauwesen Domäne).

6.3 Informationsextraktion mittels HMM

Nachdem die Theoretischen Grundlagen in Abschnitt 4.1 hinsichtlich des Aufbaus und der Funktionsweise einer Informationsextraktion mittels Hidden-Markov-Modelles erklärt worden ist, wird hier des Weiteren erklärt, wie man das Hidden-Markov-Modell zu konfigurieren hat. Damit der Problemstellung, die richtigen Informationen aus den Rechnungsdokumenten zu extrahieren, entgegen wirken zu können, müssen essenziell wichtige Schritte des Hidden-Markov-Modell eingehalten werden. Man kann den vereinfachten Ansatz, wie in Abbildung 6.3 gezeigt, anwenden.

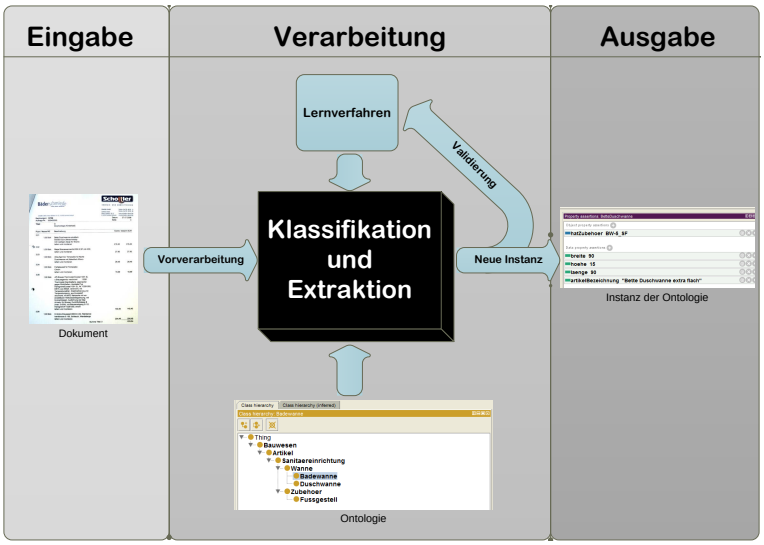


Abbildung 6.3: Ablauf der Informationsextraktion anhand eines HMM-Modells

6.3.1 Vorverarbeitung

Wie auch im vorherigen Kapitel erwähnt, werden die Rechnungen von der Firma Schottler GmbH bereitgestellt und liegen nun in Textformat vor. Diese semistrukturierten, in Text Format vorliegenden Rechnungsdokumente, werden nun für das anschließende Training mit dem Hidden-Markov-Modell vorbereitet. Zur Vorbereitungsphase gehören folgende Schritte:

1. OCR-Fehler aus den gegebenen Rechnungsdokumenten manuell entfernen um Extrakations Genauigkeit des HMM-Systems zu erhöhen und damit gleichzeitig dafür zu sorgen, daß die gleiche Struktur jeweils gewahrt wird (d. h. die Rechnungen sollten sich in der Struktur nicht unterscheiden, da sie die Rechnungen derselben Firma darstellen).

2. Die Rechnungsdokumente werden in diesem Schritt soweit modifiziert, daß in den Rechnungen nun jeweils die Produktbeschreibungen mit XML Tags versehen sind. Diese kennzeichnen die Daten, welche aus den Rechnungen zu extrahieren sind.

3. Am Ende eines jeden Rechnungsdokuments wird ein "ENDOFDOC" gesetzt, dass das Ende eines Rechnungsdokuments signalisieren soll.

Bei Schritt 1 werden sämtliche OCR Fehler manuell entfernt, um die Genauigkeit der Informationsextraktion zu steigern. Des Weiteren entfernt man die OCR Fehler, um einen gewissen idealen Zustand anzustreben und mit dem das HMM-System zurecht kommen soll. Das Hidden-Markov-Modell soll sich die Struktur genau einprägen können, aber durch OCR Fehler wäre das dann kaum möglich, beziehungsweise würde es, die Rechnungsdokumente betreffend dann keine einheitliche Struktur geben. Zusätzlich wurden alle deutschen Umlaute ersetzt, wie zum Beispiel "ü" durch "ue", "ö" durch "oe" und "ä" durch "ae". Diese Ersetzung erfolgte, da das verwendete HMM-System ursprünglich für natürlich-sprachliche Texte auf englischer Sprache konzipiert wurde. Damit die in Abschnitt 6.2 erwähnten Eigenschaften wie z. B. das jeweils zwischen den Produkten bzw. den Produktbeschreibungen(die letztendlich zu extrahieren gilt) es einen Zeilenabstand gibt nicht verschwindet, wurden die Rechnungsdokumente so modifiziert das zwischen den Produktbeschreibungen eine Leerzeile eingefügt worden ist damit diese Eigenschaft erhalten bleiben. Rechtschreibfehler und unbekannte Zeichen die nicht ordnungsgemäß durch die OCR Software extrahiert werden konnten, wurden ebenfalls entfernt.
Bei Schritt 2 geht es darum die Trainingsdokumente, das sind die Rechnungsdokumente die für das Training mit dem Hidden-Markov-Modell ausgewählt worden sind, aus dem Dokumentenkorpus so zu präparieren sodass das Hidden-Markov-Modell

lernt die Produktbeschreibungen aus den Rechnungsdokumenten zu extrahieren. Wie auch schon erwähnt worden ist, werden die Produktbeschreibungen nun an bestimmten Positionen mit XML-Tags versehen, die kennzeichnen, daß sich ab dieser Position sich innerhalb des XML-Tags eine Produktbeschreibung befindet. Das Ziel ist es, eine HMM zu trainieren, indem man mithilfe eines vorgetaggten bzw. annotierten Textes die Parameter eines HMM's so anpasst, dass das HMM für die Wortfolgen in diesem Text die richtigen Wortfolgen mit möglichst geringer Fehlerrate ausgibt.
Nach Schritt 1 und 2 erhalten wir also folgende Beispielrechnung aus den Rechnungsdokumenten, die für das Training ausgesucht worden sind.

Ein Trainingsdokument nach der Vorverarbeitungsphase

```
Baederschmiede
..sehen, planen, wohlfuehlen
Schottler GmbH, Neuer Bahnhof 10-12,D-54528 Salmltal-Doerbach
Angebot.          Blaesius Prof. Dr. Karl-Hans, Longuich
Angebots-Nr.: 0405040008
Schotler
ENERGIE-   UND   UMWELTTECHNIK
Schottief GmbH                          Telefon 06578-9828- 0
Salmtal-Center                           Telefax 06578-9828-28
Neuer Bahnhol 10-12                 www.schottler-salmtal.de
D-54528 Salmtal Datum: 28.06.2005
Seite:5
Titel:
WC-Anlage (Kinderbad)
Posnr. Masse ME        Beschreibung
2,01
1,00 Stck <Produkt>Duravit Wand-Tiefspuel-WC, Happy D weiss</Produkt>
liefern und montieren
2,03
2,04
2,05
1,00 Stck     <Produkt>Duravit WC-Sitz, weiss Scharniere: Edelstahl</
    Produkt>
liefern und montieren

1,00 Stck     <Produkt>Geberit Betaetigungsplatte Artline, weiss mit
Betaetigung von oben/vorne</Produkt>
liefern und montieren

1,00 Stck     <Produkt>Geberit Schallschutz-Set fuer Wand-WC und
Wand-Bidet</Produkt>
liefern und montieren
...
...
...
ENDOFDOC
```

Ein Produkt aus dem Produktkatalog nach der Vorverarbeitungsphase

```
Systemprogramm
Ausstattung
Typ-Nr
Masse
borda diago echo aqua
<Titel>Spiegel</Titel>
<Produktbez>Leuchtspiegel "Regio"</Produktbez>


OKM: 1940 mm Watt               Hoehe:  <Hoehe>  900 mm</Hoehe>
1 aufgesetzte Stab-   LSR...   Tiefe:  <Tiefe>  150 mm</Tiefe>
leuchte. alufarbig  11  ...8010 Breite: <Breite> B00 mm</Breite>  838,- 636,- 638,- 638,-
2 offene Regale mit 24  ...9010 Breite: <Breite> 900 mm</Breite>  687,- 687,- 687,- 687,-
je 1 Glasablage     24  ...1010 Breite: <Breite>1000 mm</Breite>  731,- 731,- 731,- 731,-
1 Schalter          24  ...1110 Breite: <Breite>1100 mm</Breite>  782,- 782,- 782,- 782,-
1 Steckdose         24  ...1210 Breite: <Breite>1200 mm</Breite>  834,- 834,- 834,- 834,-
1 durchgehende      21  ...1310 Breite: <Breite>1300 mm</Breite>  884,- 884,- 884,- 684,-
Ablage, dekorgleich 21  ...1410 Breite: <Breite>1400 mm</Breite>  934,- 934,- 934,- 934,-
                                Breite: <Breite>1500 mm</Breite>  976,- 976,- 976,- 978,-

Mehrpreis fuer Anti-Beschlag-System (600 x 400 mm. 230 V / 50 W): 137,- EUR
Durchgehende Montageschiene nicht moeglich.

<Produktbez>Spiegel mit vertikaler Beleuchtung</Produktbez>

2 Leuchtstoffroehren  Hoehe: <Hoehe>800 mm</Hoehe>
1 Schalter rechts LSD... Tiefe: <Tiefe>35 mm</Tiefe>
2x13    ...0800 Breite: <Breite> 800 mm</Breite>    611,-   611,-   611,-   MV
2x13    ...0900 Breite: <Breite> 900 mm</Breite>    646,-   646,-   646,-   646,-
2x13    ...1000 Breite: <Breite>1000 mm</Breite>    686,-   686,-   686,-   686,-
2x13    ...1100 Breite: <Breite>1100 mm</Breite>    726,-   726,-   728,-   728,-
4x13    ...1200 Breite: <Breite>1200 mm</Breite>    796,-   796,-   796,-   796,-
4x13    ...1300 Breite: <Breite>1300 mm</Breite>    835,-   835,-   835,-   836,-
4x13    ...1400 Breite: <Breite>1400 mm</Breite>    877,-   877,-   877,-   877,-
4x13    ...1500 Breite: <Breite>1500 mm</Breite>    920,-   920,-   920,-   920,-
Mehrpreis fuer Anti-Beschlag-System (600 x 400 mm. 230 V / 50 W): 137,- EUR
```

```
<Produktbez>Spiegel mit Leuchtkranz</Produktbez>

1 Leuchtstoffroehre Watt Hoehe:  <Hoehe>  800 mm</Hoehe>
1 Schalter LSJ... Tiefe: <Tiefe> 250 mm</Tiefe>
Leuchtstoffroehre   13  ...0800 Breite: <Breite> 800 mm</Breite>  715,- 622,- 622,- 622,-
13 W zur indirekten 13  ...0800 Breite: <Breite> 900 mm</Breite>  751,- 663,- 663,- 663,-
Beleuchtung des     21  ...1000 Breite: <Breite>1000 mm</Breite>  804,- 699,- 688,- 699,-
Waschtische         21  ...1100 Breite: <Breite>1100 mm</Breite>  841,- 729,- 729,- 729,-
Rueckwand           21  ...1200 Breite: <Breite>1200 mm</Breite>  914,- 794,- 794,- 794,-
Korpusfarbe         28  ...1300 Breite: <Breite>1300 mm</Breite>  984,- 864,- 854,- 854,-
                    28  ...1400 Breite: <Breite>1400 mm</Breite> 1080,- 938,- 936,- 996,-
                    28  ...1500 Breite: <Breite>1500 mm</Breite> 1117,- 972,- 972,- 972,-
Bei den Preisgruppen diago. echo und aqua in den Farben Weiss Hochglanz und Rot
Hochglanz ist der Spotboden Weiss Seidenglanz. Bei Pergamon Hochglanz
ist der Spotboden Pergamon Matt.
Beschlagfarben B0001 - Chrom. B0003 - Edelmatt. bitte angeben.
Preise in EuRO ohne MwSt. Alle Masse im mm
2 5/2009 97
```

6.3.2 Training

Hier werden die vor verarbeiteten Rechnungsdokumente in das HMM-System eingegeben, um das HMM-System darauf zu trainieren. Dabei gibt es 2 verschiedene Trainingsvarianten, um das HMM zu trainieren. Die erste Möglichkeit besteht darin, daß man entweder von einer simplen Struktur bzw. Modell ausgeht und darauf das HMM trainiert oder es besteht die Möglichkeit eine komplexere Struktur bzw. Modell zu lernen und darauf das HMM zu trainieren.
Der Algorithmus der zum Trainieren der HMM benutzt wird nennt sich Baum-Welch-Algorithmus, welcher im Unterabschnitt 4.1.1 behandelt worden ist. Der Algorithmus funktioniert so, daß basierend auf einem anfänglichen Modell die Wahrscheinlichkeit der Realisation berechnet wird. Während der Berechnung wird festgehalten, wie oft Übergänge und Ausgabesymbole verwendet worden sind.
Im nächsten Schritt werden die Parameter neu berechnet, daß häufiger benutzte Übergänge und Ausgabesymbole höhere Wahrscheinlichkeiten erhalten als weniger häufig benutzte. Nach der Anpassung wird das Modell für die Realisierung eine höhere Wahrscheinlichkeit errechnen, also besser angepasst sein.

Die zwei Schritte werden so oft wiederholt, bis sich die Modellparameter nur noch unwesentlich ändern. Dabei ist die Vorverarbeitung, insbesondere das Vortaggen der Daten sehr wichtig und hat großen Einfluss auf das Lernverhalten des HMMs. Zum Beispiel wurde beim annotieren bzw. taggen darauf geachtet, daß sich die Produkttags "<Produkt>" sich vor den Wörtern "liefern und montieren" befinden. Damit wurde sichergestellt dass das System lernt das sich vor diesem Konstrukt aus genau diesen Wörtern ein Produkt befindet und auch dort endet. Dadurch das vor einer Produktbeschreibung immer wieder eine Maßeinheit vorkommt, wurden die Produkt Eröffnungstags "<Produkt>" so gesetzt, daß sie nach der Maßeinheit erscheinen. Dadurch wird zum Beispiel gelernt, dass Wörter die nach einem solchen Muster auftauchen, eine höhere Wahrscheinlichkeit besitzen zu einer Produktbeschreibung zu gehören. Die Wörter die nicht getaggt worden sind, werden vom Algorithmus des HMM-Systems in die Kategorie "kein Produkt" gesteckt. Also werden diese Wörter den Hintergrund Zuständen (Background States) zugeordnet. Die Produktbeschreibungen hingegen entsprechen den Ziel Zuständen (Target States).
Da der Kopf also der Header der Rechnungen auf jeder Rechnung dieselbe Anschrift und Telefonnummer u.s.w beherbergt, geht das System nach dem Lernverfahren davon aus, wann immer es in anderen Unbekannten Rechnungen auftaucht, dass es ein Background State, also ein Hintergrund Zustand ist und nicht für die Informationsextraktion in Frage kommt. Das heißt nicht das Background States unwichtig sind. Ganz im Gegenteil, sie sind ein wichtiger Bestandteil eines jeden HMM-Systems, da sie auf Ziel Zustände hindeuten können. Das System würde im obigen Beispiel erkennen dass das Wortgebilde "1,00 Stck" ein Background State (ein Hintergrund Zustand) ist, aber jedoch ein Zustand, der mit großer Wahrscheinlichkeit auf ein Ziel Zustand hindeutet. Während des Trainings lernt das Hidden-Markov-Modell bei diesen Rechnungen, falls das Wortgebilde "liefern und montieren" auftaucht, daß auf ein End-Zustand in dem aktuellen Modell hinweist. Da bei einem Hidden-Markov-Modell der aktuelle Zustand nur von genau einem vorherigen Zustand abhängt, würde das System nach einem solchen Ausgabesymbol bzw. diesem Obervationssymbol("liefern") lernen, von einem Ziel Zustand in ein Hintergrund-Zustand zu wechseln, daß auf ein Ende einer Produktbeschreibung hindeutet.

Nachdem ein HMM-Modell trainiert worden ist, d. h. dessen Parameter richtig justiert worden sind, ermöglicht das System die Speicherung des gelernten HMM-Modells auf ein permanentes Speichermedium, sodass man es nicht erneut trainieren muss.

6.3.3 Informationsextraktion

Nachdem das Hidden-Markov-Modell auf die Trainingsdokumente trainiert worden ist, wird nun das trainierte HMM auf unbekannte Rechnungsdokumente im Dokumentenkorpus angewandt. Dies wird bewerkstelligt, indem der Viterbi-Algorithmus der in folgendem Paragraph 4.1.1 erklärt worden ist, auf das Problem angewandt wird. Dadurch wird die am wahrscheinlichste Sequenz von versteckten Zuständen bei einem gegebenen Hidden-Markov-Modell und einer beobachteten Sequenz von Symbolen bestimmt. Diese Zustandssequenz wird auch als Viterbi-Pfad bezeichnet. Das Resultierende Ergebnis des Viterbi Algorithmus ist dann eine Zeichenkette, welche aus Nullen und Einsen besteht. Dabei steht eine Null für ein Background State (Hintergrundzustand) und eine Eins für ein Target State (Zielzustand), dass es zu extrahieren gilt.
Folgendes Beispiel soll das Verfahren verdeutlichen:

```
Man nehme folgende Eingabe an : ''1,00 Stck Duravit Duschwanne''
Man nehme folgende Ausgabe an : [0,0,1,1]
```

Im obigen Fall würde das System im unbekannten Dokument das Wort "Duravit" und "Duschwanne" als Produktbeschreibung erkennen und somit extrahieren, da es jeweils auf ein Zielzustand hindeutet.
Anschließend werden die extrahierten Information zur weiteren Verwertung durch das System in eine Textdatei gespeichert. Dabei steht jede Zeile in der Textdatei für eine Produktbeschreibung die extrahiert worden ist.

6.3.4 Instanziierung der Ontologie

Kontext-Feature

Features die zur Klassifizierung mittels maschinellen Lernens eingesetzt werden, sind zum Beispiel Wort-Häufigkeiten, Themen-Signaturen, "WordNet" beziehungsweise "GermaNet"-Features.
Bei der Verwendung von Wort-Häufigkeiten wird jeder Instanz in einem Trainingsset eine Menge solcher Features zugeordnet. Diese beschreiben, wie oft die Wörter, welche rund um diese Instanz stehen, im Zusammenhang mit einer bestimmten Kategorie auftreten. So können neue Instanzen einer Kategorie zugeordnet werden. Andere Möglichkeiten ohne Einsatz von maschinellen Lernen, die Bedeutung von Wörtern zu identifizieren, ist das Festellen der Ähnlichkeit von Wörtern anhand kontextueller Patterns. Zum Beispiel werden die Wörter "Bier" und "Wein" in ähnlichen zusammenhängen benutzt. Kennt man die Bedeutung eines Wortes, so lässt sich unter Umständen auf jene des anderen Wortes schließen. D. h. wenn bekannt ist das "Bier" ein alkoholisches Getränk ist, kann man bei dem Wort "Wein" auch auf ein alkoholisches Getränk schließen.

Unbeaufsichtigte Methoden

Eine weitere Möglichkeit nachdem die Informationen erfolgreich aus den Rechnungen extrahiert worden sind ist, daß man nun Zeilenweise das Text Dokument mit den extrahierten Informationen durchgeht (da einzelne Zeilen jeweils eine Produktbeschreibung präsentieren) und versucht in dieser Produktbeschreibung nach Termen zu suchen die zu einem Konzept zuzuordnen sind. Hier basiert die Ontologie Population auf einem Vektor-Feature Ähnlichkeit zwischen jedem Konzept c und einem Term t das klassifiziert werden soll. Diese Methode nennt sich die **Class-Word-Annäherung** [40] und gehört zu der Klasse der unbeaufsichtigten Methoden (engl.: unsupervised methods). Um zum Beispiel zu bestimmen, wie viel das Wort "Duravit" eine geeignete Instanz von der Klasse "Badewanne" ist, findet diese Methode die Feature-Vektor-Ähnlichkeit zwischen dem Wort "Duravit" und dem Wort "Badewanne". Jeder Instanz von einer Testmenge T wird wird einem Konzept der Kon-

zeptmenge C zugeordnet. Die Features werden vom Korpus extrahiert und folgender Klassifikationsalgorithmus wird dann angewandt: [40]

```
classify(T,C,Corpus)
  foreach(t in T) do {
    v_t= getContextVector(t,Corpus);}
  foreach(c in C) do {
   v_c= getContextVector(c,Corpus);}
  foreach(t in T)do{
   classes[t] = argmax sim(vt, vc);}
  return classes[];
end classify
```

Die in diesem Algorithmus vorkommenden Methoden zur Berechnung der Kontextvektoren, kann mit Hilfe einer Co-occurence-Analyse [46] gefunden werden. Im ersten Schritt des Algorithmus würde nach der Co-occurence-Analyse jeweils für die Terme t, die jeweils zu einer Produktbeschreibung zugehörig sind, die Häufigkeit des gemeinsamen Auftretens von Termen t mit anderen Wörtern im Korpus C analysiert. Aus den bestimmten Häufigkeiten kann nun die Ähnlichkeit zweier Worte bestimmt und die Menge der Konzepte so in einem metrischen Raum angeordnet werden, d. h. es ergeben sich so genannte Kontextvektoren. Bei dem Kontext kann es sich um einen Satz,einen vordefinierten Wortabstand oder einen Absatz handeln. Nehmen wir zum Beispiel an, dass der erste Term t in einer Produktbeschreibung das Wort "Bette" wäre. Und sich dann daraus wie folgt ein Kontextvektor ergibt:
Als erstes wird eine Tabelle erstellt der die absolute und relative Häufigkeit des gemeinsamen Auftretens der in Spalte und Zeile eingetragenen Wörter angibt. In Klammern ist die Wahrscheinlichkeit, daß die beiden Wörter gemeinsam in einem Absatz auftreten.

cooccurence	Bette	Badewanne	extraflach
Bette	16(p=0,064)	6(p=0.024)	3 (p=0,012)
Badewanne	6(p=0,024)	26(p=0.104)	17 (p=0,068)
Extraflach	3(p=0,012)	17(p=0.068)	39 (p=0,156)

Nachdem diese Tabelle erstellt worden ist, werden nun die *bedingten Wahrscheinlichkeiten* p(a|b) und p(b|a) berechnet. Hierbei steht p(a|b) für die Wahrscheinlichkeit, dass in einem Kontext, in dem b auftritt, auch a auftritt. Nach der Berechnung für dieses Beispiel entsteht nun folgende neue Tabelle :

p(x|y)	Bette	Badewanne	extraflach
Bette	1.00	0.230	0.077
Badewanne	0.375	1.00	0.435
Extraflach	0.187	0.436	1.000

Die Ergebnisse der Co-occurrence-Analyse können nun zum Clustering verwendet werden oder auch direkt Ontologiebeziehungen gewonnen werden. So kann man alle Regeln, die ein bestimmtes Maß an Wahrscheinlichkeit überschreiten, direkt als Beziehungen in die Ontologie aufgenommen werden. Im Beispiel könnten dies alle Beziehungen sein, die eine Wahrscheinlichkeit von 0.4 überschreiten. Die Zeilen der Tabelle sind nun die Kontextvektoren der jeweiligen Terme.
Auch dieser Vorgang wird nun wiederholt, um die Kontextvektoren für die Konzepte der Ontologie zu berechnen. Danach wird jeder Term t einer Klasse zugeordnet, dessen Kontextvektor dem Kontextvektor des Terms t am ähnlichsten ist.

KAPITEL

SIEBEN

Konzeptionelles Design: Case-Based Reasoning

Nachdem wichtige Grundlagen, Methoden und Prinzipien durch den theoretischen Teil des 'Fallbasierte Schließens' (Kapitel 5) geliefert wurden, müssen beim Design, eines in der Problemdomäne einsetzbaren Systems, einige grundsätzliche Betrachtungen durchgeführt werden. So soll zunächst ein theoretisches Konzept angegeben werden, welches die in dieser Arbeit umgesetzte Methodik auf Basis des Kapitels 'Lernverfahren CBR' beschreibt. Die genaue Umsetzung auf Implementierungsebene, wird in der Beschreibung der Klassen, im darauf folgenden Kapitel gegeben.
Um die vorliegenden Eingabedaten im CBR-Framework verwenden zu können, muss zunächst eine geeignete interne Repräsentation der Informationen der Dokumente gefunden werden. Ziel wird sein, Wissen, in Form von, für die Anwendung der Domäne 'Bauwesen' und damit relevanten Informationen, zu gewinnen.
Textdokumente, als Rechnungsdokumente, liefern dem System eine Eingabe. Diese werden wie folgt verarbeitet:
Aus den Rechnungsdokumenten, die inhaltlich in textueller strukturierter Form Auflistungen bzw. Beschreibungen relevanter Konzepte der Domänenontologie enthalten, wird extrahiert. Dazu muss eine geeignete Transformation der Rechnungsdokumente, in eine für das System zur Verarbeitung geeignete und trotzdem wissensliefernde Repräsentationsform, vollzogen werden.
Um eine geeignete interne Repräsentation zu erkennen, kann zunächst eine genauere Betrachtung und Einstufung des angewandten Testkorpus vorgenommen werden.

7.1 Analyse des Eingabekorpus

Der Eingabekorpus wird aus einem repräsentativen Dokument der Dokumentklasse 'Rechnung' bestehen. Eine in der Anwendung typische Dokumentvorlage der Domäne wird verwendet. Diese wird ein exemplarisches Rechnungsdokument der Firma "Schottler" (Abbildung 7.1) sein und liegt in typischer Anordnung und Struktur einer üblichen Rechnung des Bereich Bauwesen vor. Textlicher Inhalt des Dokuments wird nicht in natürlichsprachlicher Form überbracht. In den Dokumenten werden ausschließlich Informationen in ihrer reinsten Form, in nicht (formulierter) sprachlicher Form und daher ohne grammatikalische Struktur, übermittelt. Die Struktur der Dokumente entspricht in weiten Teilen einer tabellarischen Auflistung von Informationen, in Form von z. B. Produktbestandslisten in einem allgemeinen *'DIN A4 Rechnungsformular'*. Es liegt damit ein strukturierter Dokumententyp vor.
In einem Dokument kann ein spezifischer Bereich des Gesamttextes, der relevante Informationen beinhaltet, die zur späteren Extraktion aufgefunden werden, verdeutlicht werden. Vereinfachend wird ein Rechnungsdokument in relevante und nicht relevante Bereiche kategorisiert. In dieser Arbeit wird folgende Unterteilung vorgenommen, die auf ein typisches Rechnungsbeispiel der Domäne zurckzuführen ist.

7.1.1 Dokumentstruktur

Struktur und Angaben der Rechnung kommen den Pflichtsachverhalten eines Rechnungsdokuments nach und basieren damit auf den Anforderungen der deutschen 'Umsatzsteuerrechtlichen Rechnungspflichten'. Abbildung 7.1 gibt ein Beispiel für eine entsprechende Rechnung. Ein hier abgebildetes Dokument wird als exemplarisches Rechnungsdokument aus der Menge der verschiedenen zu verarbeitende textuellen Eingabedokumente angenommen. Geringfügig andersartige Rechnungen werden einen ähnlichen Aufbau vorweisen und unterscheiden sich nur in der inhaltlichen Abfolge der Angaben. Eine Unterscheidung von Rechnungsdokumenten erfolgt oftmals auscchließlich in Bereichen, die z. B. den Rechnungssteller im Kopf des Dokumentes betreffen oder auf das Design bzw. Layout der Dokumente zurückzuführen sind. Diese Tatsache und die schematische Betrachtung wird für die weitere Verarbeitung nicht von Bedeutung sein und an dieser Stelle einzig zur Vereinfachung vorgenommen.

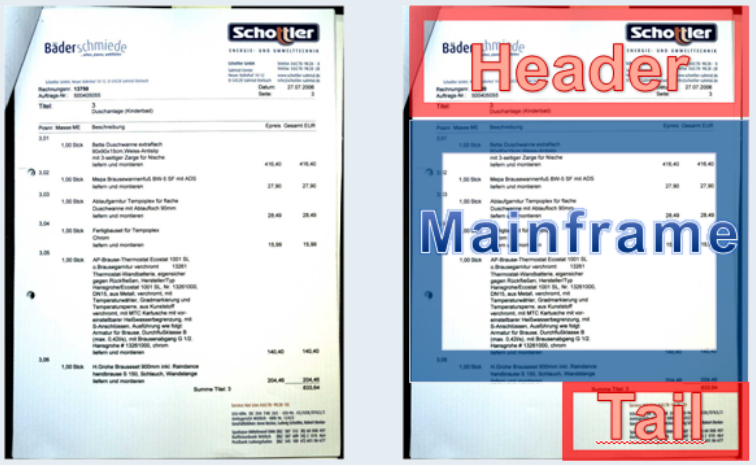


Abbildung 7.1: Domänenspezifisches Rechnungsdokument der Firma "Schottler" und schematische Dokumentaufteilung

Header Eine typische Rechnung hat, im hier rot gekennzeichneten 'Header' (siehe Abbildung 2.1), die vollständige Firmenbezeichnung mit Firmenname und Adresse, die Angaben zum Rechnungsaussteller sind. Zudem muss ein Rechnungsempfänger angegeben werden und eine, dem jeweiligen Vorgang zuordungsfähige Rechnungsnummer bzw. Angebotsnummer. Ein Pflichtfeld der Rechnung nach deutschem Umsatzsteuerrecht ist zusätzlich das Ausstellungsdatum der Rechnung. Variabel zu wählende, aber notwendige Zusatzinformationen geben an, welche Art von Dokument ausgestellt wurde. Dabei können z. B. Einträge wie Angebote, Rechnungen oder Zahlungsaufforderungen vorgenommen werden.
Ein für die Extraktion der Informationen relevanter Bereich der Rechnung ist das hier gekennzeichnete 'Mainframe', im abgebildeten Dokument blau eingerahmt. Die hier enthaltenen Informationen sind in Form einer Tabelle zeilenweise und geordnet aufgelistet. Dabei stehen in der ersten Spalte fortlaufende Positionsnummern der gelisteten Artikel. Eine zweite Spalte gibt Auskunft über Stückzahlen bzw. Mengen des jeweiligen Produktes.

Mainframe Das Dokumentfeld mit Bezeichnung 'Beschreibung', in dem Produkt- und Artikelbezeichnungen sowie weiteren Angaben zu den Produkten, wie Produktei-

genschaften, Maßangaben und zusätzliche Produktinformationen zu finden sind, ist der für den Vorgang der Extraktion relevante Textbereich. Das Mainframe beinhaltet später die in textueller Form vorliegenden und für eine darauf folgende Population einer Ontologie wichtigen 'Konzepte' und deren 'Attribute'. Die eigentliche Extraktion der Informationen durch das CBR-System wird dabei ausschließlich in diesem Bereich durchgeführt. Die Daten liegen hier in klar strukturierter Form vor, und weisen eine hohe Informationsdichte auf (engl.: data density). Die hier vorhandenen Informationen werden fast vollständig extrahiert. Eine weitere zusätzliche Angabe, die optional, also nur bei Rechnungsdokumenten und nicht bei Angeboten, angegeben ist, ist die Spalte für Einzelpreise und Gesamtpreise, welche aufsummiert den Rechnungsbetrag bildet. Angaben dieser Art und die übrigen, nicht relevanten Einträge, sind bei einer Extraktion zu vernachlässigen und werden in späteren Fallrepräsentation bzw. Anfrage an das System nicht weiter berücksichtigt.
In anschließender Konzeptionierung kann anschaulich geklärt werden, dass nur der ausgewiesene Bereich der Rechnungsdaten eine Rolle in der internen Repräsentation des Systems spielen wird. In der später angewandten Methodik sogenannter *Matches*, wird der Bereich der Produktbeschreibung betrachtet.
Die Betrachtung der Relevanz von Informationen wird ebenfalls in der durchgeführten Evaluierung (Kapitel 10) von fundamentaler Bedeutung sein. Die Auswertung der Ergebnisse relevanter Informationen wird allein das Mainframe betrachten.

Regelmäßige Muster innerhalb des Mainframes

Innerhalb des zur Extraktion der Informationen als relevant ausgewiesenen Bereiches des Textkorpus, können zwei, in der domänenspezifischen Dokumentart regelmäßig auftretende Muster, festgehalten werden. Folgende Regelmäßigkeiten konnten festgestellt werden.

- Das **erste Muster** ist, die in einer Rechnung als Pflichtfeld ausgezeichnete Angabe bezüglich der Stückzahlen oder Mengen des jeweils in der Zeile aufgelisteten Produktes. Die hier angegebene Information kann in speziellen Fällen auch den Umfang (z. B. "in Stunden") der durchgeführten Leistung, oder eine Maßangabe (z. B. "in Meter") bezeichnen.

Das Muster ist unter der Spalte mit Bezeichnung "Masse ME" aufzufinden, und wird, sofern es Stückzahlen angibt, explizit mit "**Stck**" oder "**Satz**" angegeben.

Abbildung 7.2: Das Muster "Stückzahl" in einem typischen Rechnungsdokument

Es kann folglich angenommen werden, dass dieses Muster den Anfang eines Feldes eines Produktes und seiner darauffolgenden Produktbeschreibung kennzeichnet.

- Ein **zweites Muster**, welches festgehalten werden kann, ist die Angabe "**liefern und montieren**", das in aller Regel am Ende einer Produktbeschreibung in dem Dokument angegeben wird. Es kennzeichnet in den meisten Fällen das Ende der ergänzenden Beschreibung der Produkte.

  Dieses Muster kann verwendet werden, um bei der Extraktion auszuweisen, bis zu welcher Stelle Angaben zu Produktinformationen in der Datei zu extrahieren sind.

Da von einem optimalen Eingabedokument, welches diese Art von Einträgen folglich ausnahmslos beinhalten wird, ausgegangen werden kann, werden diese Regelmäßigkeit in der eigentlichen Implementierung von Nutzen sein. Die regelmäßigen Muster im Eingabekorpus ermöglichen folglich einen regelbasierten Verarbeitungsansatz für die Extraktion der Informationen.

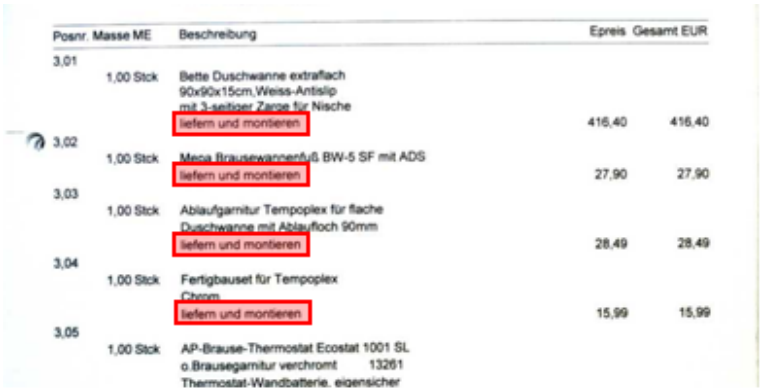


Abbildung 7.3: Das Muster "liefern und montieren" in einem typischen Rechnungsdokument

Tail Zur Vollständigkeit gibt es einen weiteren ausgewiesenen Teil des Dokuments, den ebenfalls rot markierten Bereich des 'Tails', in dem optional zusätzliche Kontaktinformationen, wie Servicehotlines oder Bankverbindungen, aber auch steuerliche- bzw. rechtliche Sachverhalte angegeben werden. In den hier vorliegenden Rechnungen ist dies ein Eintrag bezüglich der Umsatzsteuer-Identifikationsnummer. Informationen aus Tail und Header werden im weiteren Prozess zur Informationsgewinnung mit Hilfe von Case-based Reasoning nicht von bedeutender Relevanz sein.

Die schematische Aufteilung der Dokumente ist der methodischen Idee von *Hamza* und *Bellaid* in ihrer Arbeit zu 'Case-Based Reasoning for Invoice Analysis and Recognition' [19] nachempfunden. Das Prinzip der Kategorisierung von Textfeldern wird im Rahmen dieser Arbeit als prinzipielles Mittel zur Analyse der verarbeiteten Dokumente eingesetzt. In den darauffolgenden Abschnitten wird es ausschließlich zur schematischen Vereinfachung und Anschauung des relevanten und zu extrahierenden Bereiches des Dokumentkorpus dienen.

7.2 Allgemeine Konzeptionierung und Design

Nachdem eine erste Betrachtung der Struktur der angewandten Dokumente vorgenommen werden konnte, kann im nächsten Schritt zur Umsetzung eines Systems, eine geeignete interne Repräsentation der Eingabedaten gefunden werden.

Das in dieser Arbeit gelieferte Design eines 'Case-Based Reasoning Systems zur Informationsextraktion', wird *textuelle Repräsentationen* der Eingabedaten zum Erstellen der Fälle umsetzen. Ein textueller Repräsentationsformalismus wurde in Abschnitt 5.3.1 der Ausarbeitung beschrieben. Im Prozess der Verarbeitung der Daten soll das Textdokument eine gegenwärtige Rechnung — eine formale Problembeschreibung, und damit den Problemteil des theoretischen Problem-Lösungs-Paares — (Kapitel 5) formal definieren. Diese wird primär zum manuellen Aufbau der initialen Fallbasis, und nach Fertigstellung zum Erstellen neuer bisher unbekannter Fälle, in Form einer Anfrage an das System und daraufhin zum autonom adaptiven Lernen neuer Fälle, beitragen.

Festzustellen ist, dass unter Annahme des theoretischen Ansatzes zu textuellen Repräsentationen nach *Bergmann und Kolodner* [6], der Textkorpus in weitgehend strukturierter Form vorliegt und zusätzlich eine (Vor)Verarbeitung durch Tools des Natural Language Processing nicht erfolgen muss. Der Textkorpus des hier entworfenen Systems ist nach durchgeführter Analyse (Abschnitt 7.1) und daraus folgender Annahme, als eine strukturierte sequentielle Datenfolge einzustufen.

7.2.1 Methodik: Informationsentitäten des Textdokuments

Auf Grundlage der vorgenommenen Analyse der Eingabedokumente, und der damit erschlossen Annahmen, bietet sich ein Einsatz der Methodik, einer Verwendung von im Text ausgewiesenen *Entitäten* an. Diese werden wichtige relevante Informationen im Kontext einer gesamten Rechnung ausweisen. So kann ein spezifisches Dokument auf eine Anzahl von **Informationsentitäten** reduziert werden. Im Umkehrschluss repräsentiert die, aus einem Prozessschritt erkannte Menge von Entitäten, ein spezifisches Dokument innerhalb des System. Eine Einbindung von Informationsentitäten im Prozess kann auf Ebene der textuellen Repräsentation, trotz der differierenden Textstruktur, natürlichsprachlicher unstrukturierter Art und strukturierter tabellarischer Art, vorgenommen werden.
Der Einsatz von Informationsentitäten zur Formalisierung der (Eingabe)Daten, und der darauf aufbauenden Konstruktion einer formalen Problembeschreibung, wird wie

in den theoretischen Grundlagen in Abschnitt 5.6, im Ansatz des 'Textuellen Case-Based Reasoning', aufgegriffen. In dem folgenden Entwurf eines CBR-Systems, wird sowohl die Idee einer allgemein textuellen Repräsenation, als auch der methodische Ansatz des textuellen CBR verwendet.

7.2.2 Prozessmodell

Die Methodik der Umsetzung des CBR-Frameworks innerhalb der Domäne "Onto-Bau" wird nach Berücksichtigung der hier genannten Ansätze und Prinzipien aus mehreren grundsätzlichen Hauptprozeduren bestehen, die im Folgenden benannt und kurz beschrieben werden. Das Design orientiert sich dabei an dem Modell des CBR-Zyklus 5.2 und am allgemeinen Retrievalprozess mit seinen Subtasks (Unterabschnitt 5.5.1). In Abbildung 7.4 ist der Prozessablauf anhand der einzelnen Module graphisch dargestellt.

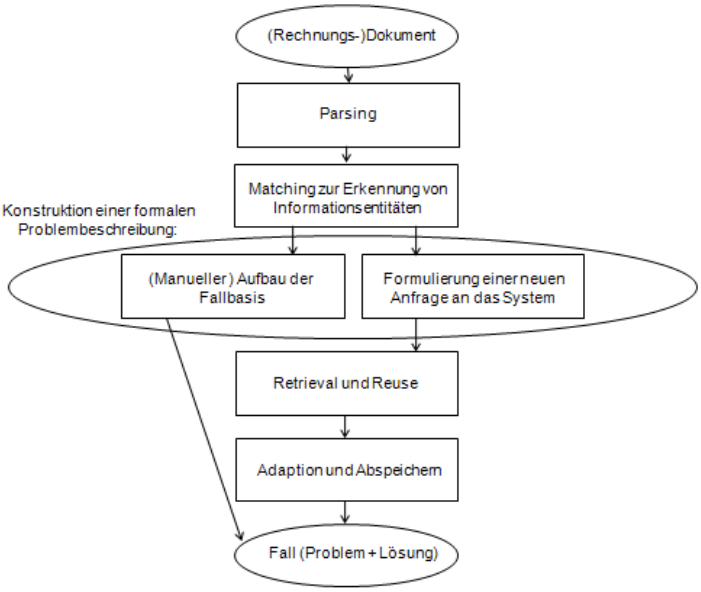


Abbildung 7.4: Vereinfachtes Prozessablaufmodell

Der Hauptprozess besteht aus folgenden konzeptionellen (Unter)Tasks, die in einem Modell der Teilprozesse des Systems beschreiben werden können:

1. **Parsing** der nach dem OCR-Scan aufbereiteten Textdokumente
2. **Matchingmethode** zur Erkennung domänenspezifischer Informationsentitäten
3. **Konstruktion einer formalen Problembeschreibung** anhand der extrahierten Informationsentitäten
4. **Aufbau der Fallbasis**, der Vorgang wird zu Anfang automatisch durchgeführt
5. **Formulierung einer Anfrage** (engl.: queue) an die Fallbasis
6. Durchführung der eigentlichen **Retrival- und Reuse-Schritte**, Auffinden eines evtl. ähnlichen Falles in der Fallbasis, Wiederverwendung der Lösung des gefundenen Falles
7. **Adaption der Lösung** und **Speichern** des neuen Falles in die Fallbasis

Die einzelnen Aufgaben, die das System im Laufe der Extraktion von relevanten Informationen bewältigt, werden im folgenden Abschnitt theoretisch beschrieben. Die Umsetzung des Verfahrens auf Ebene der Implementierung wird in der Beschreibung der Klassen erfolgen (Abschnitt 8.2).

7.3 Konzeptioneller Entwurf

In diesem Abschnitt werden die jeweiligen konzeptionellen Komponenten des Prozessmodells detaillierte erklärt.

7.3.1 Datenvorverarbeitung zum Textparsing

Damit die Eingabedaten der Domäne in, für die Verarbeitung geeigneter Form vorliegen, muss zunächst eine Aufbereitung der Daten erfolgen.
In Abbildung 7.1 (links) ist ein typisch domänenspezifisches Rechnungsdokument

dargestellt. Ein solches Rechnungsdokument wird mit der OCR-Hard-, und Software [1] optisch gescannt, und die im Dokument enthaltenen Daten werden ausgelesen. Diese liegen nach der Erfassung in elektronischer Form vor. Die Ausgabe der Texterkennung, liefert ein optional gewähltes Format und kann damit in einer für die Anwendung geeigneten Struktur vorliegen. Ein für den hier durchgeführten Task geeigneter Textkorpus kann mit folgender Struktur (Abbildung 7.5) angegeben werden.

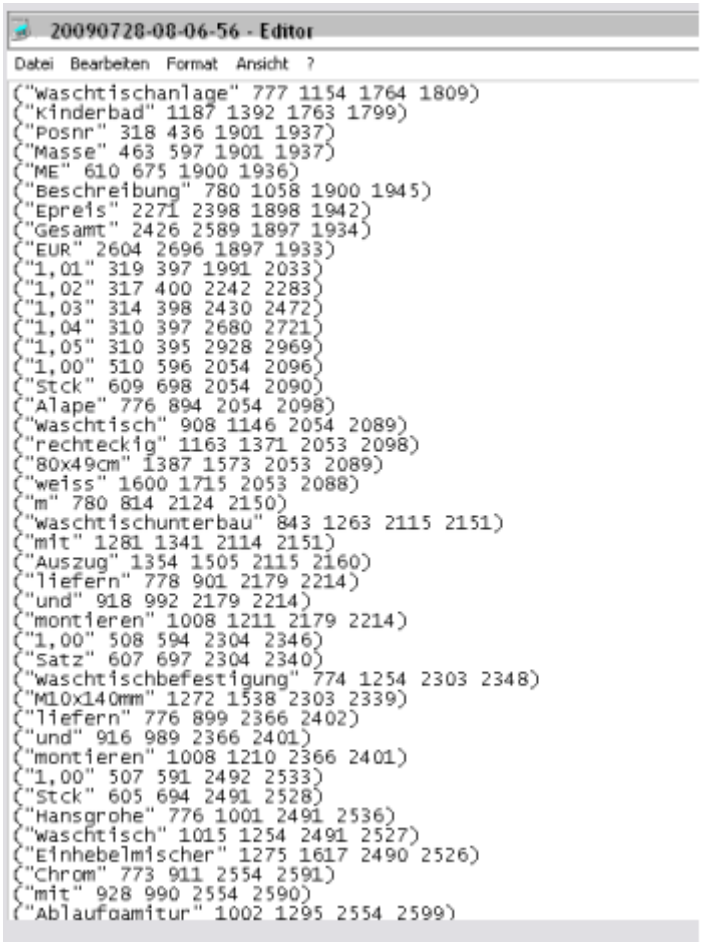


Abbildung 7.5: Rechnungsdokument nach OCR-Scan

Das Textdokument liegt nach Verarbeitung durch den OCR als textuelle Datensequenz vor. Das resultierende Dokument ist bei hier gewählter Art von Ausgabeformat, ein kompletter Scan der Rechnung, welcher Worte beziehungsweise einzelne Buchstaben oder Zahlen, die in der Rechnung vorkommen, abbildet. Wie in Abbil-

[1] Optical Character Recognition: Optische Zeichenerkennung

dung 7.5 zu erkennen, werden die Zeichen des Textes, also ganze Worte, einzelne Buchstaben und Zahlen, jeweils als Ganzes erkannt. Jedes als isolierte Einheit erkanntes Element des Textes, wird in einer eigenen Zeile aufgeführt. Diese Textelemente, werden jeweils zeilenweise ausgegeben und in einer Textdatei, die in einem Texteditor geöffnet werden kann, aufgelistet. Zusätzlich wird jedem Textelement des Dokuments ein Quadrupel zugeordnet, welches Koordinaten beinhaltet, die kennzeichnen, an welcher Position des Dokuments sich das Element befindet. Die Koordinatenangaben werden kein wichtiges Element in der Umsetzung sein, eine genauere Beschreibung wird daher ausgelassen.

Aufbereitung der Lesefehler

An dieser Stelle kann eine Aufbereitung der von der OCR-Software gelieferten Daten erfolgen. Da die Dokumentdaten mit einer optischen Texterkennung eingelesen werden, kann bei der Ausgabe des Prozesses der 'Optischen Zeichenerkennung' und Übertragung in eine gewünschte Struktur, fehlerhafte Textdokumente entstehen.

Fehlerkorrektur Dieses Phänomen wird auch "Verrauschen der Eingabedaten" genannt, da die ursprünglichen originalen Daten nicht exakt abgebildet werden konnten. Das Ausgabeformat resultiert in Zeichen, in unserem speziellen Fall der Rechnungsdokumente also Buchstaben oder Worte, die falsch oder ungenau übertragen wurden und somit nicht mit exakt dem Zeichen aus der Eingabedatei, übereinstimmen. Ein typisches Erkennungsproblem ist dabei z. B. die Übertragung der Buchstaben "rn", anstelle des eigentlichen Buchstaben "m". Ein zusätzliches Problem stellt das Erkennen und Aufnehmen von nachträglich in das Dokument eingefügten Notizen und Zusatzangaben dar. Diese Zeichen in der Eingabedatei werden durch die Hardware ebenfalls aufgenommen und verarbeitet. Hierbei werden für die weitere Verarbeitung unbrauchbare, da nicht erkennbare, textuelle Zeichen und Symbole in der Ausgabe generiert, die eine "saubere" Abarbeitung der Datensequenz erschweren. Die durch die beschriebenen Ursachen generierten Fehler werden manuell vom Anwender korrigiert, da eine anschließende Methode des Matching von Schlüsselworten, Korrektheit und Eindeutigkeit voraussetzt. Nach Aufbereitung der Daten, d. h., nach

Korrektur oder Löschung der fehlerhaften Zeichen, wird für die hier duchgeführte Anwendung kein weiteres Preprocessing verlangt.

7.3.2 Textparsing

Um die nach Aufbereitung vorliegenden Daten im System verwenden zu können, erfolgt ein Prozess des 'Parsing'. Dieser Vorgang wird durchgeführt, um eine in bestimmten Format vorliegende Eingabe, in eine für das System verwendbare Struktur oder Repräsention umzuwandeln bzw. diesem bereitzustellen. Die Daten werden dem CBR-Framework zur Weiterverarbeitung zugeführt.
Die nach der Verarbeitung der OCR-Software, also nach Umwandlung eines original Rechnungsformulars in eine Textdatei, vorliegenden sequentiellen Daten, werden durch 'Parsing' dem System zugeführt.

7.3.3 Matchingverfahren der domänenspezifischen Begriffe

Im hier beschriebenen Prozessschritt werden die nach dem Parsing vorliegenden Daten verarbeitet. Demnach müssen die, für die Anwendung in der Domäne spezifische Informationen aufgefunden werden, um eine geeignete interne Repräsentation dieser zu erhalten. Das hier gelieferte theoretische Prinzip wird auf einer begrifflich symbolischen Ebene umgesetzt.

Prinzip: Matching

Die aus dem geparsten Text vorliegende Wortsequenz (siehe Abbildung 7.5) wird zunächst nach zuvor definierten "Schlüssel(worten)" der Domäne "abgesucht". Dieser Vorgang wird im Folgenden als *Matching von Entitäten*, die den Kontext einer spezifischen Rechnung abbilden sollen, bezeichnet. Diese Schlüssel, werden im Dokument aufgefunden, sofern eines der zuvor definierten Worte gefunden wird. Dieses Auffinden ist ein sogenannter **Match**. Auf diese Weise lässt sich ein Dokument auf die spezifischen Informationen reduzieren, die mit einem jeweiligen Rechnungsdokument geliefert werden. Das Resultat ist intuitiv eine *Repräsentation aus begrifflichen Matches*. Ein Match entspricht in der konzeptionellen Repräsentation, genau einer Informationsentität.

Matching Die Matchingprozedur, ermöglicht damit den Zugriff auf typische, in der Domäne vorkommende, Begrifflichkeiten aus den Dokumenten. Im Begriffskontext einer Ontologie werden diese als die Konzepte bezeichnet. Die definierten Begriffe finden, sofern diese "*gematcht*" werden, zuerst ein Konzept (Produkt) im Text. Ein solches identifiziertes Konzept weist auf weitere zugeordnete Informationen im Anschluß des Produktes, welche die Attribute des Konzepts darstellen, hin.
Das 'Matching' wird durchgeführt, um die spätere Extraktion aufgefundener Informationen vorzubereiten. So kann zur Einleitung des Gesamtprozesses ein, mit eigens ausgesuchten Begriffen bestücktes, "Vokabular" verwendet werden, das beim hier beschriebenen Matchingverfahren als eine Art Lexikon oder Wörterbuch (Abschnitt 3.2) Einsatz findet.

Matchingliste zum Aufbau der Fallbasis Aus den definierten Matches, die explizit Produkten (Konzepte der Ontologie) entsprechen und damit die formalen Informationsentitäten des Falles anzeigen werden, wird zunächst manuell eine 'Matchingliste' (das Wörterbuch) aufgebaut. Diese Liste wird anfangs in den Prozess eingebunden, um automatisch eine Fallbasis mit ersten Fällen aus Dokumenten zu initiieren. Der Prozess der Generierung von neuen Fällen wird auf einfachste Weise gestartet. Die Verwendung einer 'Matchingliste' orientiert sich an dem Subtask des Retrieval zur 'Identifikation von Features' (Unterabschnitt 5.5.1) in einem Dokument.

Begriffe der Domäne Grundannahme für diese Methodik ist, dass das System für eine spezifischen Domäne ('Bauwesen') konzipiert ist. Dies impliziert, dass das Framework auf einer annähernd geschlossenen Menge von Begrifflichkeiten arbeiten wird. So kann dieses mit einer Teilmenge von Begriffen initialisiert werden.
Bestenfalls kann diese initiale Definition eines angewandten Vokabulars aus einer Datenbank, welche mit spezifischen Wissen der Domäne bestückt ist, oder mit Begriffen direkt aus der Domänenontologie, erstellt werden. Damit kann ein möglichst großer Bereich der Anwendung mit Begriffen abgedeckt und auf einfachste Weise ein initiales Matching zum Aufbau einer Basis ermöglicht werden.
In der beschriebenen Implementierung wird das Vokabular der Matchingliste aus einer Menge von ausgewählten Dokumente, die zum minimalen Aufbau einer Fallba-

sis beitragen, aufgebaut. Im weiteren Verlaufe des Generierungsprozesses wird beim Matching primär auf Einträge aus der Fallbasis zum Auffinden neuer Matche[2] im Dokument zurückgegriffen. Insgesamt kann das Matching mit definierten Einträge aus der Matchingliste und gelernten Einträgen aus der Fallbasis angewandt werden. Theoretisch kann damit eine minimale Anforderung an das System erfüllt werden, indem mit einer eigens definierte Matchingliste, bestehend aus initialen Schlüsselworten gestartet wird, um später auf eine sich erweiternde Fallbasis zugreifen zu können. Das Gesamtframework läuft dementsprechend zunächst auf einem geschlossenen Vokabular, welches im weiteren Verlauf des Prozesses dynamisch durch gelernte Einträge erweitert wird.

Der Einsatz der Matchingprozedur bereitet damit den nächsten Schritt des Prozesses vor. Das Matching liefert eine Menge von Informationen (Produkte), die aus dem bisher verarbeiteten Dokument entstammen und in der Formalisierung als Informationsentitäten aufgefasst werden. Das Prinzip des textuellen Case-Based Reasoning (Abschnitt 5.6) mit seiner Verwendung von Informationsentitäten, die Kontexte von Texten festhalten, liefert dafür die konzeptionelle Idee.

Die durch initiale Matching aufgefundenen Begriffen, welche folglich im nächsten konzeptionellen Abschnitt Informationsentitäten genannt werden, konstruieren formal einen Fall.

7.3.4 Konstruktion einer formalen Problembeschreibung

Nachdem die im Matchingprozess identifizierten Informationsentitäten eines Dokuments vorliegen, kann daraus die Erstellung einer formalen Problembeschreibung vorgenommen werden. Diese Problembeschreibung formuliert, wie in der theoretischen Grundlagen zur Repräsentation eines Falles (Unterabschnitt 5.3) definiert, in einer allgemeinen Fallrepräsentation einen der beiden grundsätzlichen Teile des theoretischen Problem-Lösungspaares (Kapitel 5). Das Fallwissen wird in dem System nur theoretisch in Form der Zuordnung: Problembeschreibung zu Problemlösung, vorliegen.

[2] im Sinne von Begrifflichen Übereinstimmungen

Informationsentitäten definieren Problembeschreibung

Eine bisher verarbeitete Textdatei weist nach dem Matching der domänenspezifischen Begriffe eine bestimmte Anzahl verschiedenster formaler *Informationsentitäten*, die den Matches des verarbeiteten Dokuments entsprechen, auf. Ein Fall besteht damit aus einer spezifischen Menge von Informationsentitäten. Dies impliziert, dass eine explizite Angabe der Menge von aufgefundenen Informationsentitäten, eine eindeutige formale Beschreibung eines Dokuments liefern wird. Die jeweiligen Mengen repräsentieren unterschiedlichster Zusammenstellung eines Dokuments.
Ein spezifisches Textdokument wird genau der Repräsentation eines einzelnen Falles entsprechen. Prinzipiell liefert also die Reduzierung eines Dokuments auf seine Informationsentitäten, eine formale Problemdefinition dieses gegenwärtigen Dokuments. Diese Annahme ist äuquivalent mit einer eindeutigen Abbildung eines textuellen Dokuments auf eine Menge von Informationsentitäten. In nachfolgender, vereinfachter Darstellung (Abbildung 7.6) ist dies schematisch angedeutet.

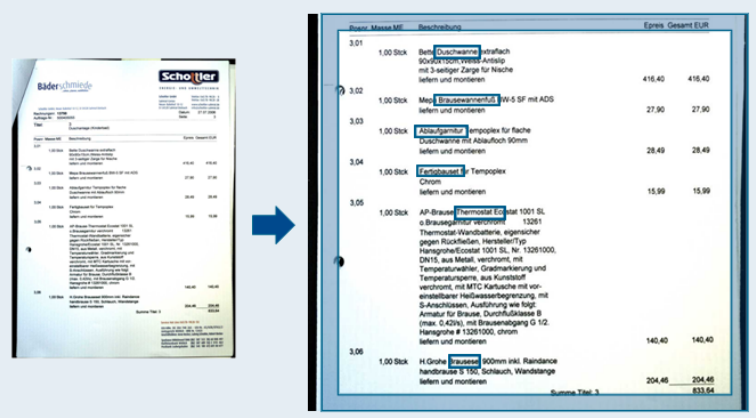


Abbildung 7.6: 'Mapping' des Rechnungsdokuments auf eine Menge von Informationsentitäten

Die textuelle Repräsentation eines Dokuments wird auf eine Menge von Informationsentitäten abgebildet. Die daraus resultierende Menge ist eine Ansammlung von Informationsentitäten, die eine konkrete Situation (Rechnung) wiedergibt und damit eine Problembeschreibung liefert.
Im weiteren Verlauf muss eine genauere Unterscheidung getroffen werden. So kann

die Menge der Informationsentitäten zwei Annahmen unterliegen, die folgende Definition liefern:

Definition:

1. *Ein Fall in der Fallbasis besteht aus einer Menge von Informationsentitäten.*

2. *Eine Anfrage (engl.: query) ist eine Menge von Informationsentitäten.*

Die genauere Betrachtung und damit verbundene Einstufung der aus dem Matching isolierten Mengen aus Informationsentitäten, wird auf ähnliche Art und Weise in einer speziellen Anwendung des Fallbasierten-Schließen, dem sogenannten 'Case Completion'[3] [26] vorgenommen, um zwischen Fällen und Anfragen, die aus Mengen von Informationsentitäten bestehen, unterscheiden zu können. In dieser Anwendung, sowie beim Case Completion, sind die Informationsentitäten damit atomare Bestandteile der Fälle bzw. Anfragen.

In der weiteren Konzeptionierung ist eine Unterscheidung notwendig, da eine vorliegende Menge von extrahierten Entitäten für zwei unterschiedliche Zwecke verwendet wird. Die aus den Entitäten gebildete formale Problembeschreibung wird im System entweder zur Erstellung eines Falles, welcher in den Fallspeicher zum Aufbau der initialen Fallbasis hinterlegt wird, verwendet, oder, nach Abschluss der Initialisierung der Fallbasis, zur Formulierung der Problembeschreibung einer neuen Anfrage an das System. Bei zweiter Möglichkeit entpricht die Menge der Informationsentitäten folglich einer neuen Fallinstanz, für die im Weiteren, die in der Implementierung umgesetzte CBR-Methode mit entprechenden Schritten durchlaufen werden muss. Der neue Fall wird, wie auch theoretisch im Zyklus des CBR 5.2 aufgezeigt, nach CBR-Vorschrift verarbeitet.
Die Generierung von Fällen zum Aufbau der Fallbasis und die Konstruktion eines neuen Falles als Anfrage auf diese Fallbasis, sind zwei voneinander unabhängige Vorgänge. Dies impliziert, dass die hier vorgeschlagene Repräsentation eines formalen

[3]Prinzip Fallvervollständigung: Bei dem Prozess der Problemlösung, liegt ein neuer Fall unvollständig und mit nur vagen Informationen vor. Im weiteren Verlauf werden weitere relevante Informationen aus bestehenden Fällen gesammelt um eine geeignete Problemlösung zu erstellen.

Falles, jeweils nur für einen der beiden Schritte vorliegen kann.

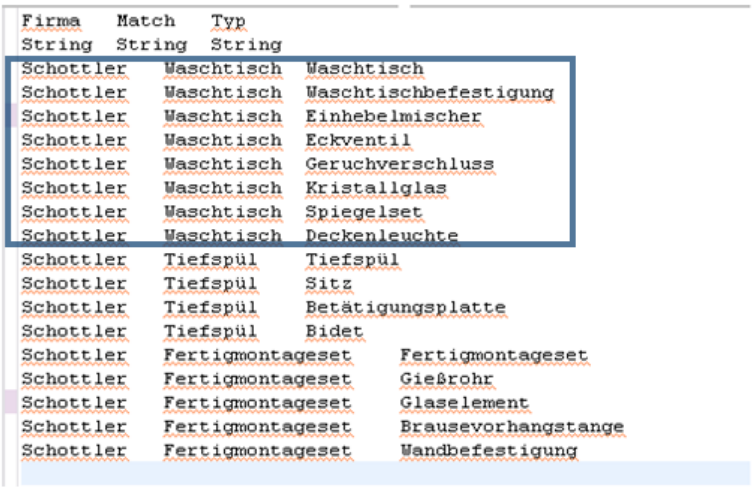
Mit der Matchingmethode werden Informationsentitäten automatisiert aus der Textsequenz gewonnen. Eine durch einen Domänenexperten eingeleitete Weiterverarbeitung dieser Daten, erfolgt zuerst im Folgenden beschriebenen initialen Vorgang. Nach Abschluss dieses Teilprozesses, liefert das System automatisiert für gegenwärtige Dokumente eine Problembeschreibung und diese werden ausschließlich zur Formulierung neuer Anfrage genutzt.

7.3.5 Initialisierung: Aufbau einer minimalen Fallbasis

Die aus den vorangegangenen Schritten erstellte Problembeschreibung wird zur Konstruktion des initialen Fallspeichers benutzt. Um die gewünschte Repräsentation eines Falles in der Fallbasis, aber auch die Struktur der globalen Fallbasis garantieren zu können, wird eine, in der Implementierung festgelegte Initialisierung von Fällen der Fallbasis, durch das System automatisiert ausgeführt. Diese Initialisierung wird, vom Anwender zu Anfang des Prozesses manuell mit Definition der Matchingliste eingeleitet, der Aufbau der Fallbasis mit Fällen aus Matches der Problembeschreibung erfolgt automatisch.
Dem System wird eine initiale Fallbasis, mit einer Anzahl von hinterlegten Fällen übergeben, mit denen zukünftige neue Fälle der Anfragen nach cbr-typischen Methoden abgearbeitet werden können. Um eine Verarbeitung zu ermöglichen, wird das System mit (Vor)wissen bestückt. Dieser Sachverhalt folgt der Annahme zu den Erklärungen bezüglich wissensintensiver Ansätze und Wissenscontainern (Abschnitt 5.2). Globales Wissen des Systems liegt als Repräsentationen von Fällen in der Fallbasis vor.

Interne Repräsenation eines Falles in der Fallbasis Ein Fall in der Fallbasis wird auf Grund der vorgegeben Struktur der Fallbasis des Frameworks 'FreeCBR' vereinfacht, aber im Sinne der gewählten Formalisierung eines Problems aus dem Design repräsentiert. Die Vereinfachung basiert auf der Annahme, dass der erste Match eines Dokuments die Definition eines konkreten Falles liefert. Ein Fall in der

Fallbasis liegt in der in Abbildung 7.7 angegebenen Repräsentation vor.



Firma	Match	Typ
String	String	String
Schottler	Waschtisch	Waschtisch
Schottler	Waschtisch	Waschtischbefestigung
Schottler	Waschtisch	Einhebelmischer
Schottler	Waschtisch	Eckventil
Schottler	Waschtisch	Geruchverschluss
Schottler	Waschtisch	Kristallglas
Schottler	Waschtisch	Spiegelset
Schottler	Waschtisch	Deckenleuchte
Schottler	Tiefspül	Tiefspül
Schottler	Tiefspül	Sitz
Schottler	Tiefspül	Betätigungsplatte
Schottler	Tiefspül	Bidet
Schottler	Fertigmontageset	Fertigmontageset
Schottler	Fertigmontageset	Gießrohr
Schottler	Fertigmontageset	Glaselement
Schottler	Fertigmontageset	Brausevorhangstange
Schottler	Fertigmontageset	Wandbefestigung

Abbildung 7.7: Repräsentation des Falles "Waschtisch" in der Fallbasis

Dem Fall 'Waschtisch' können seine zugehörigen Produkte aus dem Dokument zugeordnet werden. Das erste "gematchte" Produkt, also die erste Informationsentität, bezeichnet einen Fall.

7.3.6 Problembeschreibung als Anfrage an das System

Nach Fertigstellung der im initialen Vorgang erstellten Fallbasis das System wurde mit Vorwissen ausgestattet werden neue Dokumente ausnahmslos zur Formulierung von Anfragen an das System verwendet. Die aus den vorangegangenen Schritten (Parsing, Matching) aufgefundenen Informationsenitäten, die nach gegebener Annahme eine Problembeschreibung definieren, werden eine konkrete Anfrage, die es nach dem theoretischen CBR-Prinzipien abzuarbeiten und damit zu lösen gilt, formulieren. Die Menge der Informationsentitäten, die in einem neuen zu lösenden Fall damit das Problem spezifiziert, sind weiterhin zentrales Modul. Eine neue Anfrage, aber auch ein im Speicher angelegter Fall liefert eine konkrete Definition und folglich seine spezifische Eindeutigkeit.

Definition: *Eine neue Anfrage an das System kann eindeutig definiert werden durch:*

- *die aufgefundenen Elemente (Informationsentitäten) in der Menge, die im Kontext eines Dokumentes stehen,*
- *die Anzahl der in dieser Menge vorkommenden Elemente,*
- *die Reihenfolge (Zusammenstellung) dieser Elemente.*

Innerhalb der Problemdomäne hat damit jeder Fall eine nahezu individuelle Ausprägung und weist eine spezifische Zusammenstellung von Informationsentitäten auf.
Die Definition gibt einerseits Beleg für die Unterschiedlichkeit der Fälle und neuen Anfragen, andererseits ist abzuleiten, dass die Angaben in den Dokumenten in gewissen Kontexten stehen. Die Rechnungsdokumente können zueinander kontextuell unterschiedlich sein, ihre Produktangaben im Dokument stehen in den meisten Fällen in einem gemeinsamen Kontext. So werden neue Fälle mit neuen Problembeschreibungen generiert und neue Kontexte gemäß der CBR-Methodik erschlossen. Eine Problemlösung durch Retrieval, Adaption und anschließende Speicherung wird vollzogen.
Wie bei der Konstruktion der initialen Fälle, ist die Formalisierung eines Problems zur Anfrage (Query) ein erster Bestandteil der Repräsentation eines formalen Falles. Eine Problembeschreibung des Paares Problem-Lösung wird damit geliefert und die theoretische Zuordnung einer geeigneten Lösung, wird in den folgenden konzeptionellen Phasen des Prozesses vorgenommen.

Die Klassenbeschreibung der Implementierung (Kapitel 8.2) gibt eine spezifische Erklärung der theoretischen Modellierung der Fallbasis. Das Framework 'FreeCBR' wird die Repräsentation einer Fallbasis unterstützen, auf die ein globaler Lösungszugriff erfolgt. Theoretische Fallrepräsentationen werden in einer einheitlichen Fallbasis hinterlegt, in der keine explizite Unterscheidungen der Fälle im Sinne von isolierten Problem-Lösungspaaren vorgenommen wird. Einzelne gespeicherte Fälle werden konzeptuell zu unterscheiden sein. Bei einer Anfrage, die künftig auf die erstellte Fallbasis erfolgt, kann die Fallbasis als einheitliches Konstrukt betrachtet werden.
In Abbildung 7.7 ist ein Ausschnitt des Textfiles, der die Fallbasis des Frameworks enthält und in der praktische Umsetzung des CBR-Systems Einsatz findet, zu sehen.

Da nach der Initialisierungsphase zur Erstellung einer Fallbasis eine stetige Erweiterung durch neue Fälle, die nun autonom vom System gelernt werden und die in die Basis abgelegt werden, erfolgt, werden neue Problembeschreibungen aus Dokumenten fortan nur zur Erstellung einer Query verwendet.

7.3.7 Retrieval und Reuse

Das Retrieval des Systems wird durch das Framework 'FreeCBR' sequentiell durchgeführt. Mit vorliegender Problembeschreibung als eine gegenwärtige Anfrage an das System, kann ein Retrievalschritt durchgeführt werden.
Die Problembeschreibung besteht weiterhin formal aus der Menge der Informationsentitäten. Diese Menge wird sequentiell mit den Problembeschreibungen der Fälle, die in der Fallbasis gespeichert vorliegen, abgeglichen.

Sequentielle Suche mit Ähnlichkeitsmaß

Durch eine Suchprozedur des Framework "FreeCbr" werden gefundene Informationsentitäten aus dem neuen Fall, der Anfrage, mit den Informationsentitäten der Fälle im Speicher sequentiell verglichen. Eine gewichtete Ähnlichkeitsfunktion des Frameworks berechnet ein Ähnlichkeitsmaß und sucht nach einem ähnlichsten Fall. Übereinstimmungen von gleichen Matches, also zwischen Elementen der jeweiligen verglichenen Mengen, können so aufgefunden werden. Mengen mit einer großen Anzahl von Übereinstimmungen innerhalb des Mainframes sind sich kontextuell ähnlich. Anhand der Berechnung kann in der Fallbasis folglich eine "Problembeschreibung" gefunden werden, die der aktuellen Anfrage am ähnlichsten ist. Im Retrieval wird ein Fall, eine Menge von Informationsentitäten gefunden, die eine höchste Anzahl gleicher Informationsentitäten aufweist. Angewandt wird also ein einfaches, auf die Anwendungsdomäne spezialisiertes Ähnlichkeitsmaß. Theoretisch beschreibt dieses kontextuell ähnliche Mengen von Informationsentitäten.
Explizit beinhaltet ein aufgefundener Fall gleiche Produkte und Produktbeschreibungen, die dem aktuellen Problemfall der Anfrage eine optimale Lösung, in Form von ergänzenden Angaben, liefern können.
Damit kann die gelieferte Lösung zur Verarbeitung der aktuellen Anfrage beitragen

und ein ausgewählter Fall wird hierfür wiederverwendet. Eine detailliertere Beschreibung der Matchingmethode des Retrieval, wird in der Implementierungsbeschreibung gegeben.
Eine entsprechende Lösung muss für die vorliegende Problemstellung angepasst werden. Diese Aufgabe wird in der Lösungsadaption beschrieben.

7.3.8 Lösungsadaption und Speicherung des Falles

Nachdem ein zur Lösung geeigneter Fall aus der Fallbasis vorliegt, wird die Adaption der Lösung für den gegenwärtigen Fall vorgenommen. Dazu muss zunächst wieder vergegenwärtigt werden, wie eine solche Lösung eines in der Anwendung aufkommenden Problems, theoretische aussieht.

Lösungsansatz der Fälle

Eine Lösung liefert einer Problembeschreibung, die aus einer Menge von "gematchten" Informationsentitäten besteht, eine zugeordnete Menge an kontextuell ähnlichen Informationsentitäten. In den, in der initialen Fallbasis definierten Fällen, entspricht diese Menge exakt der Menge von Entitäten, die in der eigentlichen Problembeschreibung formuliert ist. Hier liegt eine eindeutige Zuordnung von Problem und Lösung vor, und die Menge der Informationsentitäten aus Problembeschreibung und Lösung ist bei Initialisierung identisch. Für die initial gespeicherten Fälle entspricht die Problembeschreibung genau der Lösung des Falles. Die Fälle in der Fallbasis liefern neuen Anfragen eine Lösung, die von diesen adaptiert werden. Daher sollte die initiale Fallbasis einen möglichst kontextuell breiten Bereich von Anfragen, mit einer Lösung abdecken können. Sobald neuere Fälle gelöst wurden, erweitern diese automatische die bestehende Fallbasis, und weiteres Domänenwissen wird generiert.

Für die Anwendung innerhalb der Domäne, bedeutet eine Problemabarbeitung, dass aus der Domäne, bestehend aus Konzepten um domänenspezifische Produkte und deren Produktbeschreibungen, aus einer Eingaben von Rechnungsdokumenten verschiedenster Art, eine Anfrage an das System gestellt wird, und diese mit einer Ausgabe von Produkten aus dem Dokument und weiteren Angaben, die im Kontext stehen,

löst. Die Anfrage ist eine Menge aus Informationsentitäten, die Produkten entsprechen, die der Rechnung entstammen. Die entsprechende Lösung, auf eine Anfrage, d. h., auf eine Anzahl von Produkten, generiert eine erweiterte Menge aus Produkte, die im Kontext mit den in der Problembeschreibung vorkommenden Produkten stehen.

Problemlösung

Um eine Problemlösung in das System integrieren zu können, muss nach Vorlage der Problembeschreibung aus den Mengen der Informationsentitäten, eine geeignete Lösung für die Problembeschreibung gefunden werden. Eine optimale Fallbasis, besteht aus gelösten Fällen, die den Umfang der Eingabe aus unterschiedlichen Rechnungen abdeckt und somit einem Problem, die ähnlichste Lösung bereitstellt. Eine Lösung wird dann gemäß der jeweiligen Einträge in der Rechnung und dem Kontext ausgegeben.
Für die hier beschriebene Anwendung liegt nach Formulierung, einer geeigneten Lösung zu der gelieferten Problembeschreibung, ein, für die Anwendung im 'Fallbasierten Schließen' formal ein 'Problem-Lösungs Paar' vor. Ein solches ist in Abbildung 7.8 angedeutet und repräsentiert einen Fall in der gelieferten Umsetzung auf theoretischer Ebene.

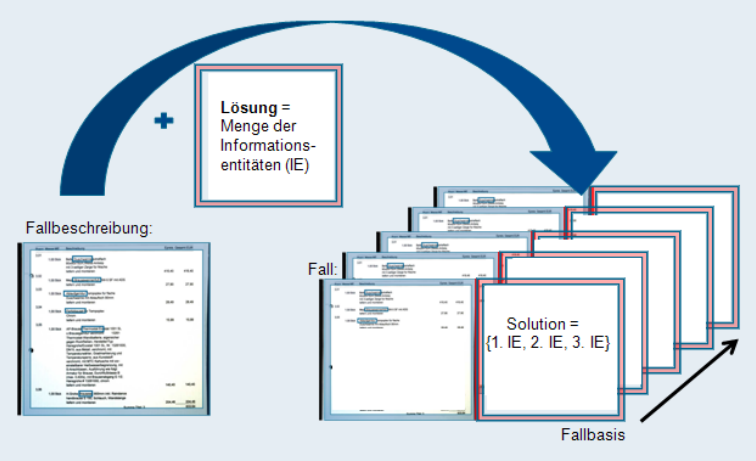


Abbildung 7.8: Erweiterung der Fallbasis mit Problembeschreibung und zugeordneter Lösung

Allgemeine Lösung Allgemein betrachtet, liefert eine Lösung eines Falles die extrahierten Informationen aus einem spezifischen Dokument der Eingabe. Konkret sind dies Produkte mit ihren Zusatzangaben aus dem eingegebenen Rechnungsdokument.
Um die Produkte, die als jeweilige Informationsentitäten des Dokumentes vorliegen, zu extrahieren, wird eine generelle Extraktionsvorschrift angeben. Diese Definition beschreibt, welche einem Match folgenden Angaben im Dokument, also im Kontext der Informationsentität stehende Beschreibungen, extrahiert werden sollen. Über ein in den Dokumenten regelmäßig auftretendes Muster, wird festgelegt, bis zu welcher Stelle nach einem Match extrahiert werden soll.

Fallspezifische Lösung Die eigentliche und damit fallspezifische Lösung eines vorliegenden Falles, ist in erster Linie die vollständige Ausgabe der Menge von Informationsentitäten die dieser beinhalten kann. Ziel ist es, bei Eingabe einer spezifischen Rechnung, dem Dokument eine seinem Kontext entsprechenden Menge von Informationsentitäten als Lösung zu liefern. Diese Menge kann zum einen nur explizit aus den extrahierten Informationsentitäten des gegenwärtigen Dokuments bestehen. Zum anderen kann sie Informationsentitäten liefern, die im Kontext zu den bisher gefundenen stehen.
Eine Lösung liefert damit Erkenntnisse über Inhalte einer Rechnung, in Form von aufgeführten Produkten und deren Beschreibungen, sowie potentiellen Zusatzprodukten. Hinsichtlich dessen wird im Gesamtsystem weiteres Wissen über kontextuelle Zusammenhänge der Entitäten gefolgert.

Beispiel *Ein auf die Problemdomäne bezogenes Beispiel kann dies verdeutlichen. In einem Rechnungsdokument wird einzig die Produktangabe "Waschtisch" erkannt. Erschlossener Kontext aus der Prozedur des Fallbasierten-Schließens liefert nicht aufgeführte Produkte wie "Waschtischbefestigung" und "Waschtischplatte", die zur möglichen Vervollständigung des Falles und somit als zusätzliche Angabe als Teil der Gesamtlösung angegeben werden.*

Lernfeatures Das CBR-System unterstützt damit eine kleine Anzahl von eigenen Lösungsmethoden. Diese sind als Lernen einer Problemlösung im Gesamtsystems zu verstehen und werden als 'Lernfeatures' bezeichnet. Somit kann in Folge eines ersten Lernfeatures, eine unvollständig Rechnung mit weiteren geeigneten Produkten angereichert werden. In einem weiteren Feature kann eine abweichende oder noch unbekannte Produktbezeichnung, die gegenwärtig nicht in der Fallbasis vorliegt, korrekt erkannt werden, um diese in einen richtigen Kontext von Produkten einzuordnen.

- **Lernfeature A**, *Vervollständigung*

  Es kann eine Vervollständigung des als Fall vorliegenden Dokuments durch hinzufügen geeigneter Einträge, die im Dokument noch nicht explizit angegeben sind, erfolgen. Der neue Fall kann um bisher fehlende Produkte erweitert werden. Die Menge der gelieferten Informationsentitäten aus der Anfrage, wird um die Elemente, des aus dem Retrieval stammenden Falles, vervollständigt.

- **Lernfeature B**, *Erkennung*

  Eine bisher unbekannte Informationsentität kann aufgrund seiner zusätzlich beschreibenden Angaben aus dem Dokument auf seine eigentliche Bezeichnung rückgeschlossen werden, indem wiederum die Mengen der Fallbasis, durch den Retrievalschritt, mit der aktuellen Menge verglichen wird.

7.3.9 Adaption der Menge

Ein entsprechender Adaptionsschritt zur Lösung erfolgt dabei folgendermaßen:
Wie in der Retrievalmethode beschrieben, wird ein Fall mit einer Menge ähnlichster Elemente aufgefunden. Dieser Fall ermöglicht einen Lösungansatz, der von der Anfrage adaptiert wird. Die ausgegebene Lösung besteht aus einer Menge von Informationsentitäten, die im Kontext zur Anfrage, aufgrund einer Anzahl gleicher Übereinstimmungen aus dem Retrieval steht. Die Menge enthält demzufolge übereinstimmende Produkte.

Ein nach den Lernfeatures adaptierter neue Fall, der zuvor ein Problem aufgeworfen hat und mit Hilfe der Fallbasis gelöst wurde, wird in der Fallbasis modifiziert abgespeichert und steht zukünftig für neue Anfragen bereit. Die Fallbasis wird mit jedem Fall erweitert und das Wissen im System angereichert.

KAPITEL

ACHT

Implementierung

Um die Entwürfe der Informationsextraktion mittels Hidden-Markov-Modell aus Kapitel 6 und dem Case-based reasoning System aus Kapitel 7 zu evaluieren, werden diese Entwürfe in einer Implementierung umgesetzt. Für beide Implementierungen werden zwei Frameworks benötigt. Das Framework, daß benutzt wird um das Hidden-Markov Modell zu implementieren, kommt von der Universität Stanford. Das Framework, daß benutzt wird, um das Case-Based Reasoning zu implementieren, wird von einem Autor namens Lars Johanson bereitgestellt. Dieses Case-Based-Reasoning-System ist im Free-CBR Paket enthalten. Beide Frameworks basieren auf der Programmiersprache Java, sie sind frei verfügbar und stehen unter der Apache License, beziehungsweise der GNU General Public License. Die Implementierung erfolgt in der Programmiersprache Java und nutzt die Java Standard Edition in der Version 1.6.

Das HMM-Framework, daß von der Universität Stanford kommt, ist auf der jeweiligen Internetseite der Universität im NLP-Paket[1] enthalten und wurde von der Natural Language Processing Group dort entwickelt. Die NLP-Gruppe besteht aus Forschern, Programmierern und Studenten, die es sich zur Aufgabe gemacht haben, Algorithmen zu entwickeln, die es Computern erlauben, menschliche Sprache zu bearbeiten und zu verstehen. Stanford NLP stellt eine Reihe von Analyse-Werkzeugen der natürlichen Sprache zur Verfügung. Diese können rohe englischen Sprachtex-

[1] Natural Language Processing, http://nlp.stanford.edu/software/stanford-corenlp-v1.0.2.tgz

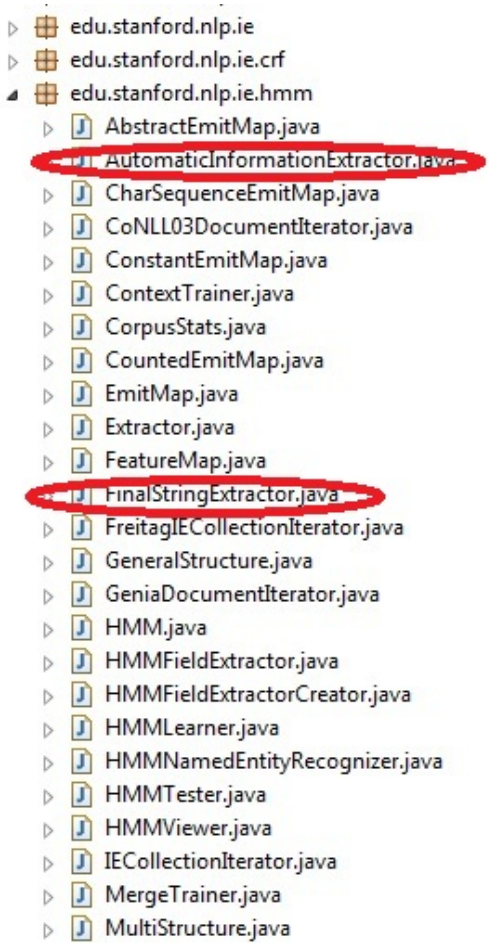


Abbildung 8.1: Das HMM-Paket der Universität Stanford

teingänge nehmen und die Grundformen von Wörtern ausgeben und ihre Wortarten normalisieren. Sie erhöhen die Struktur von Sätzen in Bezug auf Ausdrücke und Wortabhängigkeiten und zeigen an, welche nominale Wortverbindungen sich auf dieselben Entitäten beziehen. Das Paket beinhaltet unter anderem alle Werkzeuge, die für die Bearbeitung von Dokumenten in englische Sprache notwendig sind.
Für diese Arbeit jedoch ist das enthaltende HMM-Paket[2] von Bedeutung. Es beinhaltet die Methoden, um ein HMM-Modell zu lernen, trainieren und für das nachfolgende Extrahieren der Informationen aus den Dokumenten zu gewährleisten.

[2] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/hmm/ ... ... package-summary.html

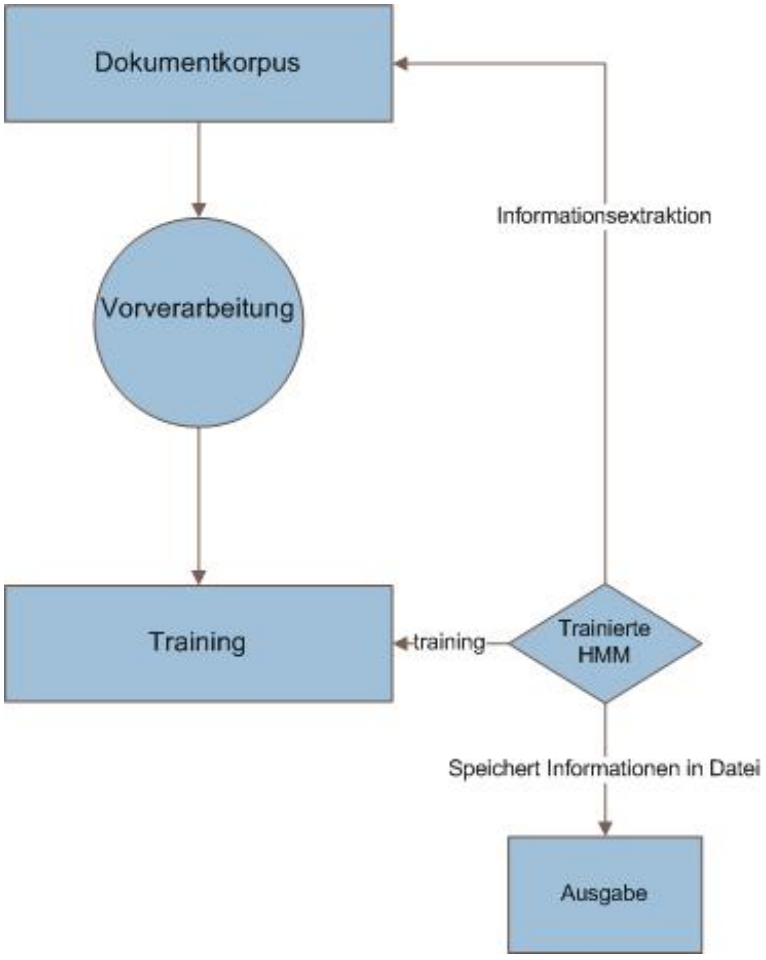


Abbildung 8.2: Informationsextraktion durch HMM-System

8.1 Implementierung des Hidden-Markov-Modells

Um die Aufgaben, die in Kapitel 6 besprochenen worden sind, bewältigen zu können, mussten zusätzlich zum HMM-System der Stanford Universität zwei weitere Klassen implementiert werden. Alle Java-Klassen der NLP sind, wie in der Java-Entwicklung üblich, in einer Java-Paketstruktur (engl.: java packages) abgespeichert (siehe Abbildung 8.1). Folgende vier Klassen, die sich im HMM-Paket befinden sind nötig, um folgende Aufgaben — wie in Abbildung 8.2 zu sehen ist — lösen zu können:

1. *AutomaticInformationExtractor*
2. *FinalStringExtractor*
3. *Extractor*
4. *HMM*

Die ersten beiden Klassen, die auf folgender Abbildung 8.1 rot markiert sind, wurden dem System zusätzlich implementiert und hinzugefügt, da eine Anpassung auf die Problemstellung nötig ist. Die beiden Klassen bedienen sich der *HMM*-Klasse, um eine automatische Informationsextraktion zu gewährleisten.

8.1.1 AutomaticInformationExtractor-Klasse

Diese Klasse wurde, wie schon bereits erwähnt, zusätzlich zum System implemen-

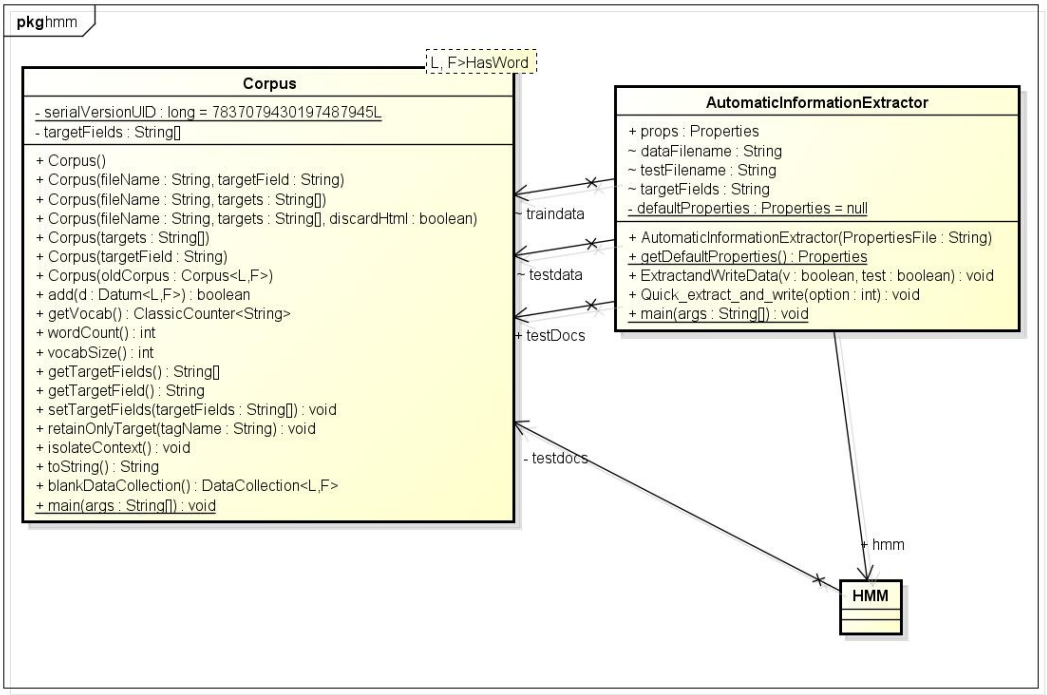


Abbildung 8.3: UML-Diagramm der AutomaticInformationExtractor-Klasse

tiert. Dadurch ist es möglich, individuell Methoden und Attribute zu definieren, ohne die im Paket mitgelieferten Klassen zu verändern. Es bedient sich also der Klassen im HMM-Paket und arbeitet mit einigen Klassen zusammen.

Die *AutomaticInformationExtractor*-Klasse dient in erster Linie zum automatischen Extrahieren von Informationen aus einem Testdokument bzw. -korpus. Nachdem spezifiziert worden ist, welche Zielfelder zu extrahieren sind, wird automatisch jeweils die HMM-Struktur für das jeweilige Zielfeld gelernt. Pro Zielfeld (Target Field) ist das Zielfeld "Produkt", da wir Produktbeschreibungen extrahieren wollen. Die HMM-

Struktur wird nach der stochastischen Optimierungsmethode, die von Freitag & Mc Callum (siehe Abschnitt 4.2) entwickelt worden ist, für das jeweilige Zielfeld gelernt oder es wird von einer simplen Struktur ausgegangen. Anschließend wird die gelernte HMM-Struktur bzw. die simple HMM-Struktur mit der Baum-Welch-Methode trainiert. Die Baum-Welch-Methode ist in dem Java-Paket enthalten. Es kann zusätzlich angegeben werden, ob bereits ein fertig gelerntes Hidden-Markov-Modell zur Verfügung steht und die Extraktion damit ausgeführt werden soll. Oder es wird eine Simple HMM-Struktur, die bereits im System vorhanden ist, geladen. Nach dem Training beginnt die Extraktion der Daten aus einem angegebenen Test Dokument. Die so gewonnenen Daten werden anschließend in einer Text Datei abgespeichert. Die Ausgabedatei besitzt den selben Namen wie ihr jeweiliges Zielfeld.

Die main-Methode

Das Programm wird, wie in Java üblich, über die *main()*-Methode gestartet.

Die main-Methode der AutomaticInformationExtractor-Klasse

```
1     public static void main(String[] args)
2         throws FileNotFoundException,
3                 IOException,
4                 ClassNotFoundException {
5         AutomaticInformationExtractor test
6         = new AutomaticInformationExtractor(args[0]);
7         test.QuickExtractAndWrite(1);
8       // test.ExtractandWriteData(false, false);
9     }
```

Nach dem Aufruf der *main()*-Methode wird eine Instanz der Klasse AutomaticInformationExtractor erzeugt und dem Konstruktor als Parameter der Pfad zur einer *Properties*-Datei übergeben. Die *Properties*-Datei ist eine Textdatei in der die Einstellungen und Optionen zum HMM-System angegeben sind.

Die hier verwendeten Optionen haben folgende Bedeutung:

testFile Diese Option ist optional und gibt den Pfad zu einem Testkorpus an, der separat zum testen bzw. zur Informationsextraktion genommen wird.

dataFile Gibt den Pfad zum Trainingskorpus an, auf das das HMM trainiert werden soll.

Nachdem eine Instanz der Klasse nun erstellt worden ist, gibt es zwei verschiedene Methoden bezüglich der Informationsextraktion.

QuickExtractAndWrite(int)-Methode Durch diese Methode ist es möglich eine bereits vorhandene HMM-Struktur (die man bereits vorher erstellt und trainiert hat) zu laden oder eine simple Struktur, die bereits im System vorhanden ist, zu laden und damit die Informationsextraktion zu starten.

ExtractAndWriteData(boolean, boolean)-Methode Durch den Aufruf dieser Methode findet die Informationsextraktion anhand einer neu gelernten und trainierten HMM-Struktur statt.

QuickExtractAndWrite-Methode

Java-Code der QuickExtractAndWrite-Methode

```
1   public void QuickExtractAndWrite(int option)
2           throws FileNotFoundException, IOException,
                ClassNotFoundException {
3
4       HMM candidateHMM;
5       String allTargetFields = this.targetFields;
6       if (allTargetFields == null || allTargetFields.length() == 0) {
7           System.out.print("mind. 1 targetfield muss definiert sein")
                ;
8       }
9       String[] targetFields = allTargetFields.split(" ");
10      for (int i = 0; i < targetFields.length; i++) {
11          Structure structure = null; // structure to train this hmm
                with
12          Properties tmp_props = Extractor.getDefaultProperties();
13          Corpus tmp_train = new Corpus(dataFilename, targetFields[i
                ]);
14          Corpus tmp_test = new Corpus(testFilename, targetFields[i])
                ;
15          if (option == 1) {
16              hmm = new HMM(structure, HMM.REGULAR_HMM);
17              hmm.train(tmp_train); // trainiere hmm mit Baum Welch
                    Algorithmus
18          }
19          else {
20              File hmmFile = new File(props.getProperty("hmmFile"));
                    // serialized
21              ...
```

Die Methode ist dafür zuständig die Trainings- und Testdaten einzulesen. Pro Zielfeld (siehe Zeile 10) werden zwei Instanzen mit dem Namen "tmp_train" und "tmp_test" vom Typ Korpus erstellt. Je nach Wahl des "option" Parameters wird dann eine simple HMM-Struktur vom System eingelesen oder eine bereits trainierte HMM wird eingelesen, um die Informationen aus dem Testkorpus "tmp_test" zu extrahieren.

Wenn der "Option" Parameter den Wert 1 besitzt, wird eine neue simple HMM-Struktur definiert und mit dem Baum-Welch Algorithmus trainiert. Ansonsten wird eine bereits trainierte HMM von der Festplatte eingelesen. Der Pfad zu der trainierten HMM-Datei ist in der Properties-Datei enthalten. Anschließend werden die Daten der HMM ausgegeben, indem die Methode "printProbs()" der Klasse *HMM* aufgerufen wird. Zur weiteren Verwendung wird dann die trainierte HMM in eine Datei mit dem Namen "Produkt.hmm" abgespeichert. Zur Extraktion der Produktbeschreibungen aus den Testkorpus wird nun die *FinalStringExtractor*-Klasse benutzt. Als Ausgabe dieser Extraktion wird ein Array vom Typ String verwendet. Am Ende wird das Array noch in eine Ausgabedatei geschrieben. Jede Zeile dieser Datei definiert dabei jeweils eine Produktbeschreibung.

ExtractandWriteData-Methode

Diese Methode verhält sich ähnlich wie die vorher beschriebene *QuickExtractAndWrite*-Methode. Der Unterschied besteht darin, daß hierbei automatisch die HMM-Struktur für das jeweilige Zielfeld nach der stochastischen Optimierungsmethode von Freitag & Mc Callum [16, 17] gelernt wird. Mit Hilfe der *learnStructure*-Methode der Klasse *StructureLearner* wird die jeweilige neue Struktur gelernt und mithilfe der Baum-Welch-Methode trainiert.

8.1.2 FinalStringExtractor-Klasse

Diese Klasse wurde zusätzlich zum bestehenden System implementiert. In dieser Klasse werden die extrahierten Informationen, die anhand eines bereits trainierten HMMs extrahiert worden sind, in ein String Array gespeichert und der Klasse *AutomaticInformationExtractor*-Klasse zur weiteren Verarbeitung übergeben. Diese Daten sollen später dazu dienen, eine bereits bestehende Ontologie zu popularisieren. Die Klasse besteht aus einer einzigen Methode, die, wie eben erwähnt, dazu dient, durch ein vorliegendes HMM die Viterbi-Sequenz auf ein unbekanntes Rechnungsdokument (= ein Dokument, das nicht zur Trainingsmenge gehört) zu berechnen.

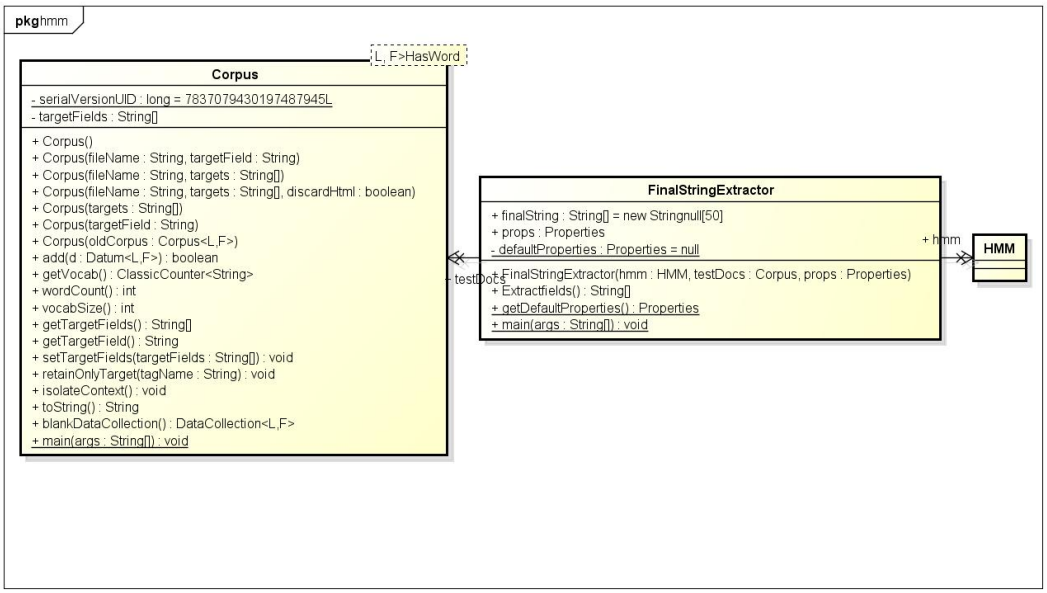


Abbildung 8.4: UML-Diagramm der FinalStringExtractor-Klasse

Extractfields-Methode

Nachdem das unbekannte Rechnungsdokument über die zweite Codezeile eingelesen worden ist, wird im nächsten Schritt die Extraktion der Informationen aus dem Rechnungsdokument durchgeführt. Dazu wird der Viterbi-Algorithmus der *HMM*-Klasse eingesetzt. Die so berechnete Viterbi-Sequenz ist eine Folge von Zuständen, die für die Extraktion noch umgewandelt werden muss. Durch die *getLabelsForSequence(bestStateSequence)*-Methode ist dies möglich, da die Sequenz sich nun aus Nullen und Einsen zusammensetzt. Die Null steht für ein Hintergrundzustand und eine Eins in der Sequenz steht für ein Zielzustand. Die jeweiligen Produktbeschreibungen sind in einer Menge von Listen des Typs String gespeichert und werden nach einander an der entsprechenden Position im finalString Array abgespeichert. So ergibt sich, daß jede Position im Array für eine Produktbeschreibung steht und so zeilenweise in eine Textdatei abgespeichert werden kann.

8.1.3 Extractor-Klasse

Diese Klasse ist Bestandteil des Java Pakets der Universität Stanford und ist eine so genannte "Command-line"-Dienstklasse, um ein HMM zu trainieren und anschließend

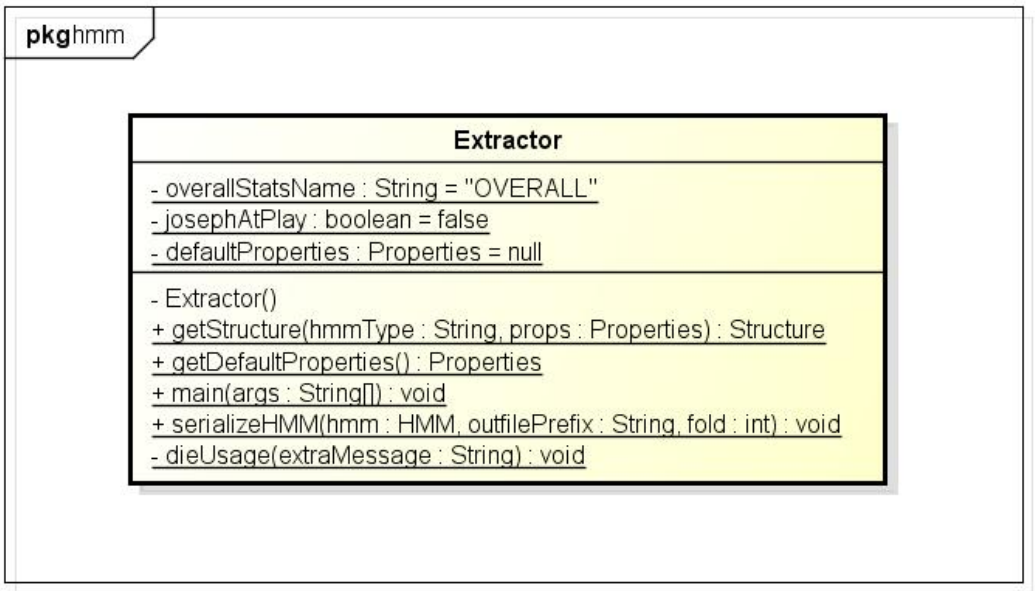


Abbildung 8.5: UML-Diagramm der Extractor-Klasse

die Performanz der trainierten HMM zu testen. Die Klasse kann entweder zum Laden einer bereits existierenden HMM Datei oder zum Trainieren einer neuen benutzt werden. Die *Properties*-Datei ermöglicht eine große Flexibilität im Punkt HMM-Training. Hier werden die wichtigsten Einstellungen erläutert, um am einfachsten und schnellsten ein HMM-Extractor erstellen und ausführen zu können.

Als erstes spezifiziert man den Pfad zum **dataFile**(Testkorpus) das es zu trainieren gilt. Ein Bruchteil der Dokumente von derselben Datei kann auch verwendet werden, um den Extractor im Anschluss an das Training zu prüfen. Das wird mit der Option **cvSlices** definiert, welches standardmäßig auf 10 steht. Das bedeutet, dass 1/10 des Dokuments für das Testen zurück gehalten wird.

Der nächste Schritt betrifft die Option **targetFields**. Hier sollte angegeben werden, welche Zielfelder es zu extrahieren gilt. Diese wären zum Beispiel die im Trainingsdokument vorhandenen XML-Tags.

Auch ist es Pflicht, ein **hmmType** zu spezifizieren. Hierbei empfiehlt es sich, die meist benutzten Optionen *s-merged* (single-flied Extractor) oder *merged* (multi-flied Extractor) zu benutzen. Beide dieser Weisen erlauben etwas Flexibilität im Spezifizieren der Übergangsstruktur einer HMM.

Die Option **contextType** wird benutzt, um die Topologie der zusammenhängenden Zustände anzugeben. Für single-Field-HMMs wird die Option *fixed* für Präfix und Suffix angewandt, um die Länge der Nachsilben-Ketten zu bestimmen. Für multi-field HMMs mit mehreren Zielfeldern ist die Option *flexible* eine bessere Alternative. Um Zielausdrücke zu modellieren, bestimmt **nts** die Anzahl von Zielzuständen. Das Setzen von **targetType** zu *chain* ordnet diese Zustände in einer Kette an und produziert die besten empirischen Ergebnisse zur Zeit.
Um letztendlich die trainierte HMM auf die Festplatte speichern zu können, erfordert es die Option **hmmOutfilePrefix** zu definieren. Weitere Spezifikationen und Optionen zum System sind unter anderem auf der Dokumentationsseite der Universität Stanford zu finden[3].

main-Methode

Als Parameter übergibt man dabei entweder lediglich nur den Pfad zur **Properties**-Datei oder zusammen mit der Option "true" für verbose zusammen an. Die Option verbose aktiviert die Möglichkeit eine detailliertere Ausgabe zu erhalten. Diese Methode besteht im weitesten Sinne aus vielen Fallunterscheidungen. Diese sind notwendig, damit das System weiss, welche Einstellungen in der Properties-Datei gemacht worden sind. Am Ende des Programms werden die Precision-, Recall- und F1-Werte ausgegeben. Precision und Recall sind zwei verwendete Metriken, um die Genauigkeit der Programmausführung zu erkennen.

[3] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/hmm/Extractor.html

8.1.4 HMM-Klasse

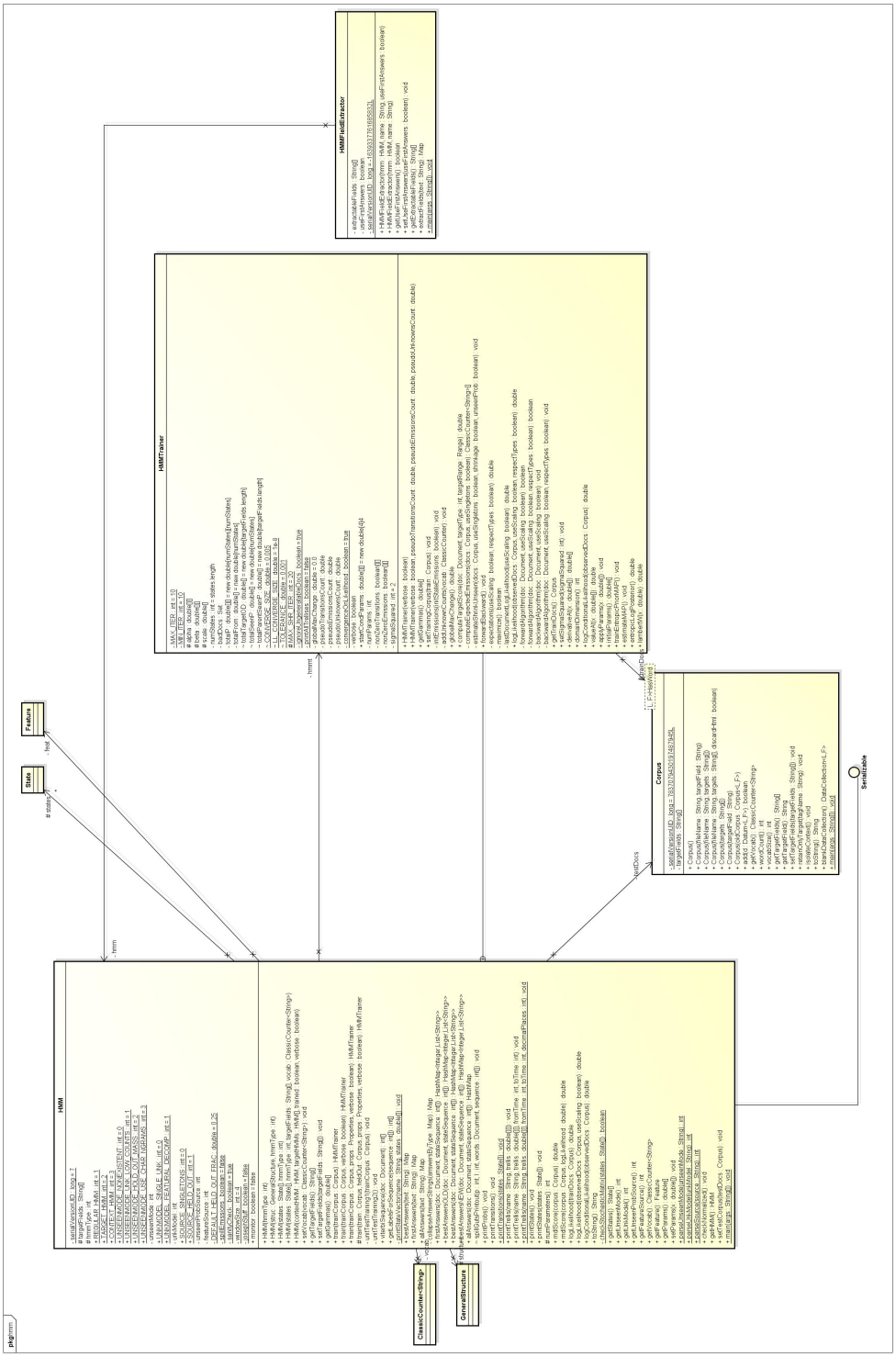


Abbildung 8.6: UML-Diagramm der HMM-Klasse

In dieser Klasse wird die Struktur eines HMMs initialisiert und trainiert. Zusätzlich zum Training bietet es Möglichkeiten der Informationsextraktion mit Hilfe des Viterbi-Algorithmus (siehe Abschnitt 4.2). Eine Standardstruktur, die bereits im System enthalten ist, oder die verschiedenen Struktur-Klassen, die im System vorhanden sind, können verwendet werden, um daraus eventuell eine neue Struktur zu erlernen beziehungsweise in eine neue Struktur zu transformieren. Drei Typen von HMMs können erzeugt werden:

1. regular HMM
2. target HMM
3. context HMM

Ein regular HMM ist ein geschlossener voller Extraktor. Ein target-HMM hingegen ist ein HMM, das nur Zielfelder repräsentiert. Ein context HMM erfährt, wo in Dokumenten Ziele erscheinen.
Der Trainingsprozess besteht dabei aus drei Schritten :

1. Reguläre Parameterabschätzung durch ein EM-Algorithmus (Expectation-Maximization-Algorithmus). Anfängliche Übergänge werden entweder als fest einprogrammierte Standardwerte gesetzt oder von der Struktur Objekt gewonnen. Anfängliche Emissionen werden veranlasst, einfache Wortzählungen zu verwenden. Die reguläre Parameterabschätzung, wird wie von Manning und Schütze (1999) beschrieben implementiert.[4] Für das Training eines HMM werden von den eingelesenen Trainingsdaten 3/4 als Trainingsdaten benutzt, wobei der Rest für die Validierung benutzt wird, um die Parameter einzustellen.

2. Bei unbekannten Beobachtungen gibt es drei Optionen, um unbekannte Beobachtungen abschätzen zu können. Das wäre einmal, dass der HMM-Code bei einem solchen Fall nichts unternehmen soll. Zweitens sollte ein geschlossenes Vokabular verwendet werden, wenn der passierte Korpus einige Wörter einer "featural"-Repräsentation auf eine disjunkte Menge von Zeichen (engl.: token) abbilden konnte. Der einfachste Weg, um diese Idee zu implementieren, ist es,

[4] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/hmm/HMM.html

alle einelementige Mengen auf ein unbekanntes Zeichen(UNK) abzubilden. Die dritte Möglichkeit ist, in Aussicht gestellte Daten zu verwenden, um zu schätzen, wie wahrscheinlich jeder Zustand ein ungesehenes auf eine featural-Zergliederung basiertes Wort ausstrahlen soll.

3. Shrink-Abschätzung benutzt herausgenommene Daten, um die so genannten shrink-Parameter abzuschätzen, was bedeutet, dass das HMM den optimalen Kompromiss zwischen dem spezifischen Modell lernt das mehr Struktur hat, aber mehr unter der Seltenheit mit dem allgemeinen Modell leidet, dass weniger Struktur und mehr Trainingsdaten hat.

Train-Methode

Die Train-Methode, die ein wichtiger Bestandteil eines jeden HMM-Systems ist, ist dafür zuständig das HMM auf die jeweiligen eingegebenen Einstellungen und Trainingsdokumenten zu trainieren. Dabei bestimmt die **Properties**-Datei auf welche Art das HMM trainieren werden soll. Entweder im "joint"-Modus oder das HMM wird bedingt (engl.: conditionally) trainiert. In den **Properties** wird außerdem geregelt, wie man mit unbekannten Wörtern umzugehen hat oder ob man die Option "shrinkage" aktivieren will oder nicht. Natürlich gibt es viel mehr Einstellungen die in der Properties-Datei angegeben werden kann. Die möglichen Optionen die eine Properties-Datei enthalten kann, kann man in der Java-Dokumentation des System ansehen. Falls keine **Properties**-Datei hierbei angegeben wird, werden die Standard-Einstellungen aus der *Extractor*-Klasse geladen. Wenn keine "held out"-Daten übergeben werden, Daten die zum Validieren benötigt werden, wird ein Teil des Trainingskorpus dafür verwendet werden. Zum Training des HMMs wird der Standard-Baum-Welch-Algorithmus benutzt. Das Resultat dieser Methode ist ein trainiertes HMM.

viterbiSequence-Methode

Übergeben wird der Methode ein Testdokument aus dem Testkorpus und berechnet dann die wahrscheinlichste Sequenz von versteckten Zuständen bei den beobachteten

Sequenz von Symbolen (also die Wörter der Rechnung). Diese Zustandssequenz wird unter anderem auch als Viterbi-Pfad bezeichnet. Die Ausgabe des Algorithmuses ist eine Viterbi-Zustandssequenz mit einem Startzustand und einem Endzustand. Mithilfe der *getLabelsForSequence*-Methode kann danach die Folge von Zuständen in eine binäre Zustandsfolge umgewandelt werden. So ist bekannt, welches Wort im Dokument zur Extraktion in Frage kommt.

8.2 Implementierung des CBR

Die Umsetzung des Case-Based Reasonings Systems basiert auf der Opensource Java-Implementierung des Programms 'FreeCBR' [5] von *Lars Johanson*.
Um den Entwurf des Systems aus Kapitel 7 anwenden zu können, wurden dem 'Free-CBR-Paket' eigene neue Klassen hinzugefügt. Das Programm 'FreeCBR' wird dabei als Framework genutzt.
Genauso wie beim HMM-System sind alle Java Klassen des 'Free-CBR' in einer Java-Paketstruktur abgespeichert. Die vier folgenden Klassen, die sich im 'Free-CBR-Paket' befinden sind relevant, um den im CBR-Konzept (Unterabschnitt 7.2.2) beschriebenen Prozess mit dem gegebenen Framework umzusetzen. Die erste Klasse 'CBR' ist die Hauptklasse des 'FreeCBR-Frameworks'. Das Paket besteht aus folgenden Klassen:

1. *CBR*
2. *CBRInvoice*
3. *Parser*
4. *CaseBaseMaintenance*

Die aufgeführten Klassen 'CBRInvoice' 'Parser' und 'CaseBaseMaintenance' gewährleisten den in Abbildung 8.7 aufgezeigten Ablauf der Informationsextraktion. Das Prinzip der Extraktion von Produktbeschreibungen aus der Implementierung kann auf schematischer Ebene abgebildet werden. Hierbei wird die Extraktion mit Hilfe des

[5] http://freecbr.sourceforge.net/

Auffindens, von den im Konzept erklärten Regelmäßigkeiten (Paragraph 7.1.1) und darauffolgenden Produktbeschreibungen anhand eines Strukturgramms, dargestellt.

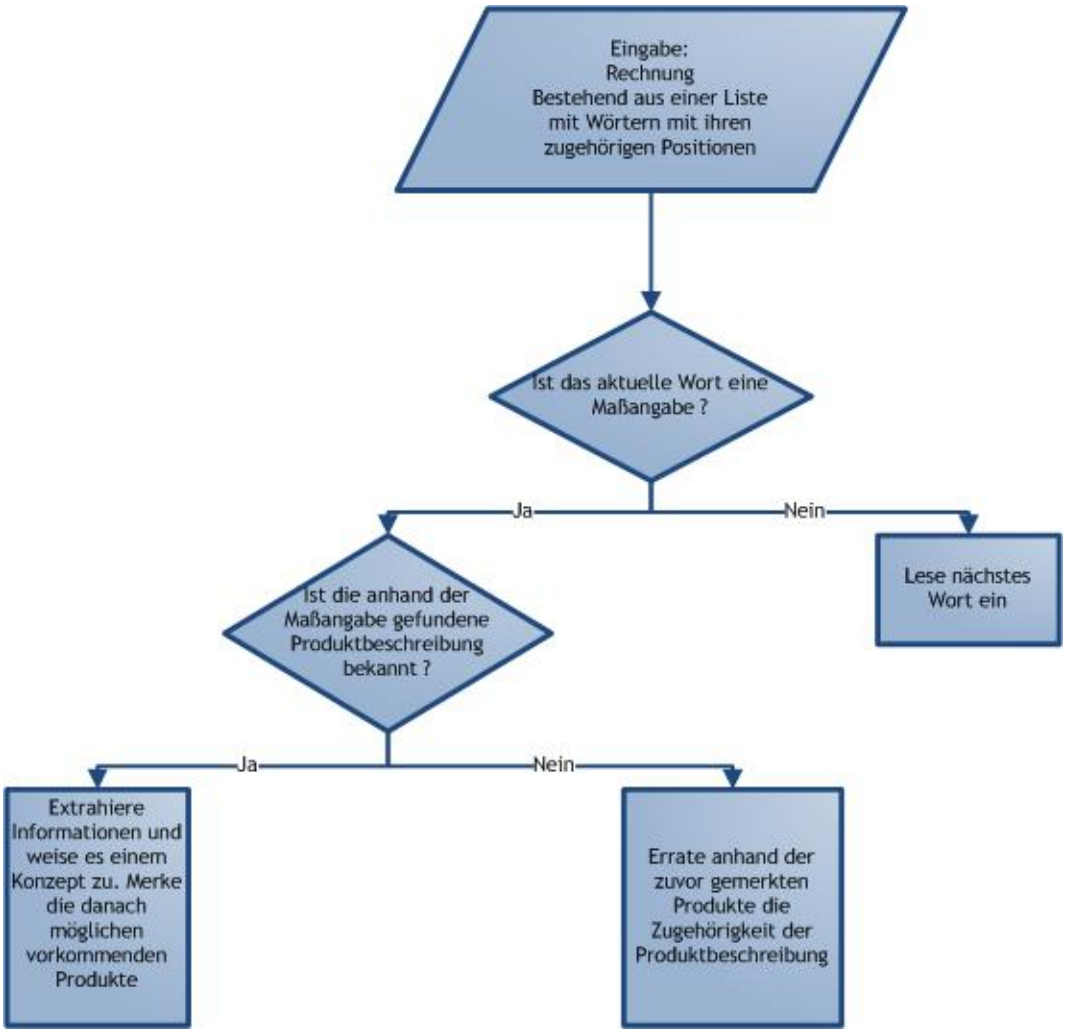


Abbildung 8.7: Informationsextraktion durch CBR System

Die letzten drei Klassen, CBR-Invoice, Parser und CaseBaseMaintenance wurden zu dem bestehenden System ergänzt, um das Framework auf die Problematik der "Extraktion aus Rechnungsdokumenten" mit geeigneten eigenen Klassen und Methoden umzusetzen. Die drei Klassen greifen auf die *CBR* Klasse zu, um primär die Fallbasis abzuspeichern und Zugriff auf die Standardmethoden des Programms zur Umsetzung des Case-Based Reasonings zu erhalten.

Typische Standardmethoden des CBR-Systems sind z. B. Hinzufügen von Fällen in

die Fallbasis oder die Suche nach ähnlichen Fällen, die sich bereits in der Fallbasis befinden. In der konkreten Umsetzung zur Extraktion aus den Rechnungen wird die Fallbasis dazu benutzt, um die Produkte eines Falles, die in der Fallbasis gespeichert werden, ausfindig zu machen. Die Fallbasis ist dabei ein einfaches Textfile, in dem die Produkte und bestimmte zusätzliche Produkte die mit dem ersten Produkt auftreten, eingetragen werden.

Diese werden unter Ausnutzung der Regelmäßigkeit im Dokument erkannt und extrahiert. Die Erkennung und Extraktion solcher Produkte wird im Ausgabefile des Frameworks angegeben. Die Identifikation der Produkte aus den Dokumenten wird später durch die Einordnung und Befüllung von Konzepten in die Ontologie der Domäne "Bauwesen" erweitert. Dieser letzte Schritt ist die 'Ontologiepopulation', die im Kapitel 11 beschrieben wird.

8.2.1 CBR-Klasse

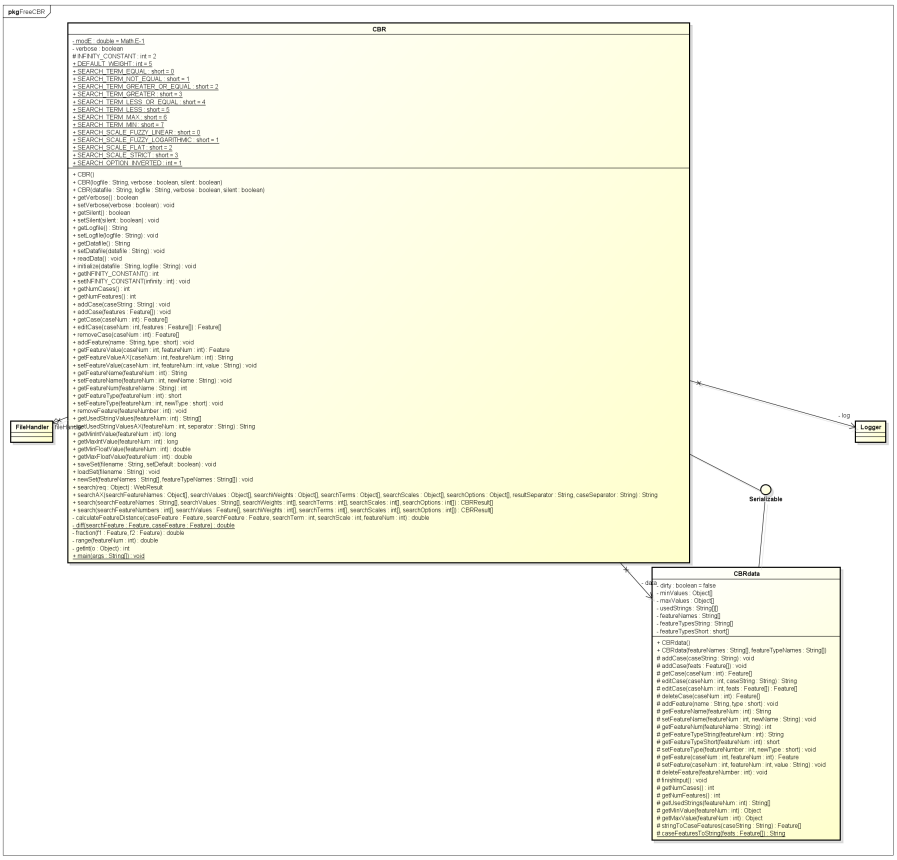


Abbildung 8.8: UML-Diagramm der CBR-Klasse

Die Klasse 'CBR' ist im 'Free-CBR' Package enthalten und entspricht der Hauptklasse des ganzen Systems. In der Hauptklasse wird unter anderem die Fallbasis, die vorher manuell erstellt worden ist, eingelesen und initialisiert. In der Fallbasis besteht jeder Fall aus einem vordefinierten Satz von Eigenschaften (Features), die der erstellten Fallbasis zugeordnet werden. Diese Eigenschaften werden durch einen Namen und einen Datentyp definiert. Mögliche Datentypen sind *String*, *MultiString*, *Float*, *Int* und *Bool*. Eine ausführliche Beschreibung der Repräsentation der Fallbasis erfolgte bereits im Kapitel zum CBR-Entwurf (Unterabschnitt 7.3.5). In der Implementierung wurde folgender Aufbau der Fallbasis gewählt: Wie in Abbildung 8.9 zu sehen, wurde ein erstes Feature mit Name 'Firma' erstellt, dieses bezeichnet den Rechnungsersteller. Ein zweites Feature wird als 'Match' definiert und zeichnet den ersten Match eines jeden Dokuments aus. Ein erster Match wird zur Bezeichnung eines Falles verwendet. Das letzte Feature mit dem Namen 'Typ' ist eine "Liste", der aus dem Dokument identifizierten und zu extrahierenden Produkte.

Firma	Match	Typ	
String	String	String	
Schottler	Waschtisch	Waschtisch	
Schottler	Waschtisch	Waschtischbefestigung	
Schottler	Waschtisch	Einhebelmischer	
Schottler	Waschtisch	Eckventil	
Schottler	Waschtisch	Geruchverschluss	
Schottler	Waschtisch	Kristallglas	
Schottler	Waschtisch	Spiegelset	
Schottler	Waschtisch	Deckenleuchte	
Schottler	Tiefspül	Tiefspül	
Schottler	Tiefspül	Sitz	

Abbildung 8.9: Featureeinträge in der Fallbasis

Die CBR-Klasse wird es ermöglichen, die erstellte Fallbasis nach Fällen zu durchsuchen. Eine eigene Prozedur in der Klasse sucht nach gespeicherten Fällen, die der aktuellen Problemstellung am ähnlichsten sind.

Fallsuche in der Klasse Die Fallbasis wird mit Hilfe eines gewichteten 'Euklidischen Distanzverfahrens'[6] nach Fällen, die der Problemstellung am ähnlichsten sind abgesucht. Ein dadurch erhaltener sogenannter "*Treffer-Prozentsatz*" (engl.: hit

[6] Ähnlich dem Pythagoras-Lehrsatz in n Dimensionen

percentage) wird nach folgender Formel berechnet[7]:

$$100 \cdot \left(1 - \sqrt{\frac{Falldistanz}{Summe\ der\ Gewichte}}\right)$$

Die Falldistanz ist die Distanz zwischen der aktuellen Suche und einem gespeicherten Fall. Der Wert ist eine Gleitkommazahl zwischen Null und Eins. Dieser kann wie folgt berechnet werden:

$$\begin{aligned} Falldistanz &= Gewicht_1 * {distanz_1}^2 + Gewicht_2 * {distanz_2}^2 \\ &\quad + \cdots + Gewicht_n * {distanz_n}^2 \end{aligned}$$

Der Faktor $distanz_i$ beschreibt die Distanz zwischen dem gesuchten Feature und dem aktuellen Feature aus dem gegenwärtigen Fall. Dieser Wert ist ebenfalls eine Gleitkommazahl zwischen Null und Eins. Der Wert Null bedeutet genauer Treffer, also eine Übereinstimmung. Wohingegen der Wert Eins eine maximale Distanz bedeutet. Die Gewichte aus der Formel sind wie folgt definiert. Das $Gewicht_i$ ist das Gewicht für ein Feature mit dem Index 'i'. Gewichte können je nach Relevanz oder Wichtigkeit der vordefinierten Features (Rechnung, Match und Typ) in die Berechnung eingehen. Ein Gewicht ist eine ganze Zahl größer gleich Null. Standardmäßig ist ein Gewicht auf den Wert Fünf gesetzt. Das bedeutet, dass die totale Falldistanz größer gleich Null und kleiner gleich $\sqrt{\sum_{i=1}^{n} Gewicht[i]}$ ist.
Der Umgang mit undefinierten bzw. unbekannten Features kann folgendermaßen beschrieben werden. Hierfür wurde zusätzlich ein Wert "?" eingeführt. Ist also der Wert eines Features aus einem Fall, oder der gesuchte Wert eines gegenwärtigen Features "?", dann wird dieses Feature "disqualifiziert" und fließt folglich nicht mit in die Berechnung und das Resultat ein. Besteht die Suche insgesamt aus einem "?" Wert, dann gibt es kein Resultat, demnach konnte kein entsprechender Fall gefunden werden.

Die Distanz zwischen einem gesuchten Feature, also einer Eigenschaft bzw. Eintrag in der Fallbasis und einem neuen Feature aus dem aktuellen Fall, wird mit eigenen Algorithmen des Frameworks berechnet.
In der beschriebenen Klasse 'CBR' existieren zwei Algorithmen, um die Distanz

[7] siehe Java Dokumentation

zwischen einem gesuchten Feature und einem Feature aus einem Fall zu berechnen. In der umgesetzten Implementierung bezüglich gegebener Problemstellung wird sich der Einsatz des sogenannten 'Normal Distance Algorithmus' anbieten. Die Distanzen werden also ausschließlich mit Hilfe dieses Algorithmus berechnet.
Der 'Normal Distance Algorithmus' ist wie folgt definiert:

$$distanz = min\left(1,\ \frac{differenz(gesuchtem\ \ Wert,\ \ Fall\text{-}Wert)}{(maxWert - minWert) \cdot \infty}\right)$$

Da in der konkreten Problematik nach Termen in der Fallbasis gesucht wird, eignet sich der Einsatz einer 'ist-Gleich-Suchfunktion'. Einer Funktion die auf Gleichheit prüft. So gilt für die Distanz im Falle eines genauen Übereinstimmen der Terme, das diese den Wert Null annimmt. Ist dies nicht der Fall, so wird der 'Normal Distance Algorithmus' angewandt und eine Berechnung auf Distanzwerte nach definierter Funktion durchgeführt.

Search Methode

Eine der wichtigsten Methoden in der *CBR*-Klasse ist die *Search-Methode*. Mit dieser Methode soll ein Lösungsansatz gefunden werden. Dies kann erfolgen, indem die Fallbasis mit einem bestehenden Problem nach einem möglichst ähnlichen Problem abgesucht wird. Ein "Problem" aus der Fallbasis in Form eines Falles, bestehend aus Einträgen von Produkten, wird eine Lösung liefern.
Ein wichtiger Vorteil der "framework-eigenen" Suche ist, dass diese spezifiziert werden kann. Der Anwender kann dadurch gezielt Einfluss auf die Suche nehmen und diese nach bestimmten Anforderungen einstellen. So wird angegeben, nach welchen expliziten Features in der Fallbasis gesucht werden soll. Dies bedeutet für die Suche, dass die Fallbasissuche auf bestimmte Features eingeschränkt werden kann.
Zusätzlich kann eine Gewichtung der Features vorgenommen werden. Die Gewichtung legt fest, welchen Anteil ein Feature in der Suche haben soll. Dabei werden ausschließlich Featurewerte berücksichtigt, die eine Gewichtung größer als Null für die Suche besitzen. Features mit einer Gewichtung "Null" werden nicht in die Suchmethode eingehen.

In der hier erstellten Anwendung, die eine Suche nach Produkten in der Fallbasis ausführt, wird einzig nach Features bezüglich der Angabe 'Match' und 'Typ' gesucht. Beide Features sind gleich gewichtet.
Des Weiteren ist es möglich, die Art der Suche festzulegen. Um die Art der Suche zu spezifizieren, kann ein Array mit Suchwerten angegeben werden. Diese Werte sind Definitionen der Art "CBR.SEARCH-TERM-EQUAL","CBR.SEARCH-TERM-GREATER" oder "CBR.SEARCH-TERM-GREATER-OR-EQUAL". Im Rahmen dieser Implementierung wird mit dem Suchwert "CBR.SEARCH-TERM-EQUAL" gearbeitet. Ein direkter Vergleich von Wörtern miteinander, ist in diesem Falle ausreichend.

8.2.2 CBRInvoice-Klasse

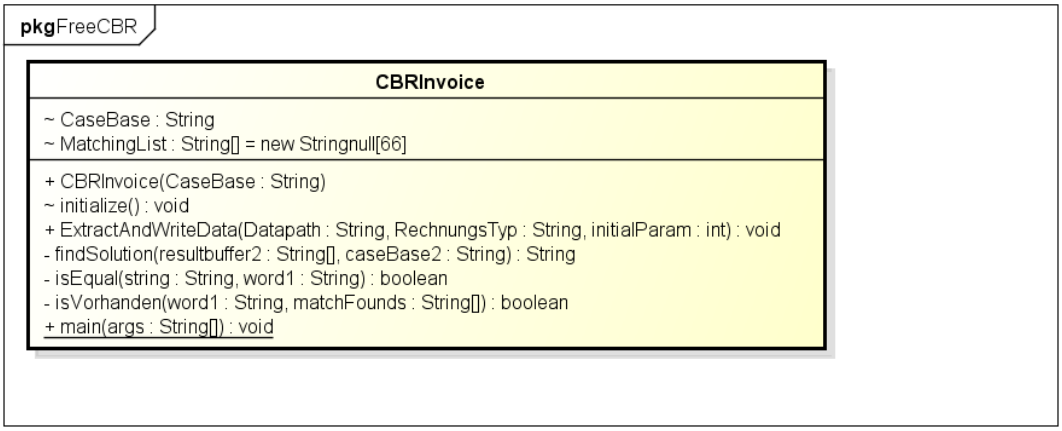


Abbildung 8.10: UML-Diagramm der CBRInvoice-Klasse

Die hier beschriebene Klasse ist eine eigens erstellte Erweiterung des 'Free-CBR' Pakets. Die Implementierung benutzt die zuvor beschriebene *CBR* Hauptklasse, um die Fallbasis einzulesen. Diese wird innerhalb des CBR-Systems initialisiert.
Die 'CBRInvoice' Klasse wird mit bestehendem Framework als Basis, eine eigens entworfene Anwendung durchführen können. So werden anhand eines initialisierten CBR-Systems, bestehend aus einer Fallbasis und der zuvor beschriebenen Suchmethode, aus einem repräsentativen Textdokument der Klasse 'Rechnung', relevante Informationen der Domäne 'Bauwesen' erkannt und extrahiert. Die erkannten Infor-

mationen eines Dokumentes werden in geeigneter Repräsentation in die definierte Fallbasis eingebaut. Mit Hilfe der aus dem Konzept beschriebenen Matching-Liste (Unterabschnitt 7.3.3) zu Anfang und der Fallbasis im weiteren Prozessverlauf werden Produkte erkannt.

Extraktionsprinzip der 'Invoice-Klasse' Das hier beschriebene Extraktionsprinzip basiert auf den im Design beschriebenen Regelmäßigkeiten (Unterabschnitt 7.1.1), d. h. regelmäßige Muster im Eingabekorpus.
In allen Rechnungen befindet sich neben einer Produktbeschreibung eine Angabe zu "Stückzahlen" des aufgelisteten Produkts. Das Programm geht davon aus, dass im weiteren Verlauf der Rechnung ein Produkt gefunden werden kann. Erfolgt innerhalb einer definierte Reichweite ein Match aus dem initialisiertem Wörterbuch oder einem Eintrag in der Fallbasis, konnte ein bekanntes Produkt aufgefunden werden. Falls kein sogenannter Match erfolgt, wird in der Fallbasis nach einer Lösung gesucht. Eine Lösung aus der Fallbasis liefert einem durch Match aufgefundenem Produkt seine aus einem vorliegendem Fall zugeordneten Produkte. So kann eine vorgeschlagene Lösungen in den Speicher geladen werden, um nicht erkannte Produkte identifizieren zu können.
Kann ein erstes Produkt nicht erkannt werden, weil noch keine Lösungen existiert, wird versucht, anhand der bisher erkannten folgenden Produkte auf das richtige Produkt zu schließen. Dazu wird in der Fallbasis durch die Suchmethode überprüft, welche Lösung am meisten zuzuordnen ist.
Die Lösungserstellungen folgen den im Konzept erstellen *Lernfeatures*: Vervollständigung und Erkennung (Unterabschnitt 7.3.8).
Produkte und deren zugehörigen Produktinformationen, die erkannt werden konnten bzw. durch eines der Lernfeatures zugeordnet wurden, werden in eine Ausgabedatei gespeichert. Ausnahmefälle, dies sind Produkte deren Zuordnung nicht möglich ist, werden mit dem Zeichen "?" in der Ausgabe markiert. Das Zeichen bedeutet, dass keine Zuordnung durch die 'Search Methode' möglich war. Das Produkt wird mit seinen Produktinformationen und dem Vermerk "?" in die Ausgabedatei gespeichert.

Die main-Methode

Die *Main-Methode* des Frameworks, ist in der eigens erstellten CBRInvoice-Klasse enthalten und wird innerhalb dieser aufgerufen.
Zunächst wird über den 'args-Parameter' der Datenpfad zur aktuellen Fallbasis übergeben, zu sehen ist dies in 'Zeile 2' des Java-Codes. Es wird eine Instanz der Klasse *CBRInvoice* erzeugt, welche die Methode *ExtractAndWriteData* aufruft und dabei die notwendigen Parameter übergibt. In Zeile drei des Programmcodes erfolgt der Aufruf der Methode zum automatischen Aufbau der initialen Fallbasis, dem ein Parameter übergeben wird, der die Anzahl der zu erstellenden Fälle angibt. Nach dem Aufbau der Fallbasis wird mit Zeile vier der eigentliche Test eines Rechnungsdokuments durch die Methode *ExtraktAndWriteData* gestartet. Dazu muss das zu testende Dokument in die Datei "test2.txt" eingefügt werden. Die Extraktion und der anschließende Einbau der gefundenen Produkte erfolgt durch das System.

main-Methode der CBRInvoice-Klasse

```
1    public static void main(String[] args) throws IOException {
2    CBRInvoice test_cbr = new CBRInvoice(args[0]);
3    test_cbr.ExtractAndWriteData("test2.txt","Schottler",3); //
         Automatischer Aufbau der Fallbasis mit einer Anzahl von
         initialen Fällen
4    test_cbr.ExtractAndWriteData("test2.txt","Schottler",0); //
         AnschlieSSender Test eines Rechnungsdokuments, welches in 'test2.
         txt' eingefügt wurde
5 }
```

ExtractAndWriteData-Methode

Diese Methode initialisiert zu Anfang den Parser. Damit wird die Möglichkeit geboten, den Dokumentenkorpus zu analysieren, um später die gewünschten Informationen extrahieren zu können. Mithilfe des 'Parsers', der in Kombination mit der Matching-Liste und der Fallbasis arbeitet, können im Dokumentenkorpus die Positionen einer potenziellen Produktbeschreibungen erkannt werden. Eine Produktbeschreibung besteht aus Produktangabe und den zugehörigen beschreibenden Informationen.

Innerhalb dieser Methode erfolgt ebenfalls optional der Aufbau der initialen Fallbasis. Diese kann entweder automatisch, wie zuvor angegeben, generiert oder manuell aus einer bestimmten Anzahl von Rechnungen erstellt werden. Bei automatischen Aufbau kann durch den Parameter *'initialParam'* angegeben werden, wie viele Rechnungen zum Aufbau der Fallbasis verwendet werden sollen.

Produktidentifikation In der Vorbereitung zur Identifikation für die spätere Extraktion wird wie folgt vorgegangen:
Nachdem eine Anzahl von "Matches" im Dokumentenkorpus gefunden worden ist, wird für jedes dieser Matche überprüft, ob es sich bei diesem Match um eine der definierten Regelmäßigkeiten in Paragraph 7.1.1 handelt. Falls dies der Fall ist, prüft die Methode, ob sich innerhalb eines definierten Nachfolgebereiches der aufgefundenen Regelmäßigkeit ein Produktmatch eines gespeicherten Produktes ergibt. Der definierte Bereich betrachtet fünf folgende Worteinheiten im Dokument. Sofern in diesem Intervall ein im System gespeichertes Wort "gematcht" wird, kann das Produkt als "bekanntes" Produkt gekennzeichnet werden. In der Fallbasis wird nach diesem aufgefundenen Produkt gesucht und eine Lösung, bestehend aus allen Produkten des Falls, temporär in einen sogenannten *'Resultbuffer'* geholt. Im 'Resultbuffer' werden die möglichen nachfolgenden, im Kontext stehenden Produkte, zu einem aufgefundenen Produkt zwischengespeichert. Ein Produkt, welches auf die Art erkannt wurde, wird zusammen mit den dazugehörigen Typ-Einträgen (Liste aus Produkten) aus der Fallbasis, die im Resultbuffer vorliegen, in die Ausgabedatei "Produkt.txt" gespeichert. Der sich so ergebende neue Fall wird in der Fallbasis als neue Lösung abgespeichert.
Das Prinzip der Verwendung eines Resultbuffer, dient vor allem auch zur Erkennung von ersten Produkten die zunächst als "unbekannt" gekennzeichnet worden sind. So wird bei einem als "unidentified" bezeichneten Produkt, nach Treffern im Resultbuffer durchsucht. Dieser Resultbuffer, in der Implementierung als *'Resultbuffer2'* definiert, sammelt alle nachfolgenden "gematchten" Produkte des Dokuments, um anhand dieser einen möglichst ähnlichen Fall aus Übereinstimmungen aus dem Resultbuffer zu finden. Ziel ist das Erschließen des ersten "unidentified" Produkts.

Die 'Resultbuffer' setzen die Lernfeatures *Vervollständigung* und *Erkennung* in der Implementierung um. Der Vorgang zum Erschließen eines geeigneten Produkts ist in Abbildung 8.11 dargestellt.

Folgende Abbildung 8.11 soll diesen Vorgang verdeutlichen.

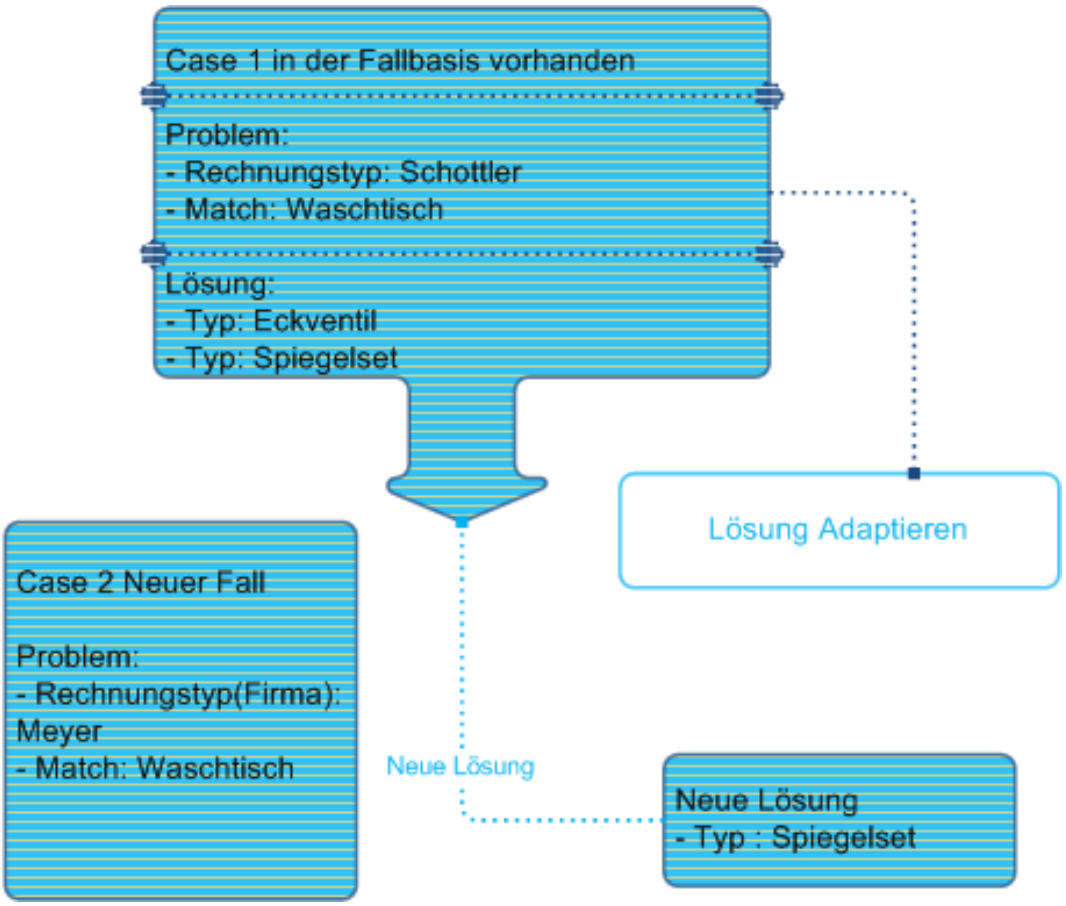


Abbildung 8.11: Aufnahme eines neuen Falles in die Fallbasis durch Adaption einer gefundenen Lösung

8.2.3 Maintenance-Klasse

Entfernung redundanter Fälle Die Klasse wurde ebenfalls nachträglich in das Framework hinzugefügt. So erfüllt sie primär die Aufgabe, mit Hilfe einer 'redundancyRemover-Methode', die sich in der Klasse *CaseBaseMaintenance* befindet, redundante Fälle aus der Fallbasis zu entfernen, um eine Auslagerung der Fallbasis zu vermeiden.

8.2.4 Parser-Klasse

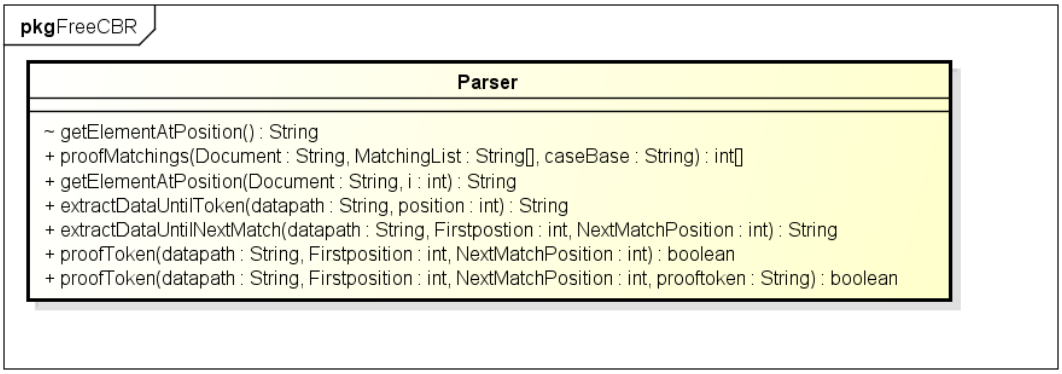


Abbildung 8.12: UML-Diagramm der Parser-Klasse

Die *'Parser'*-Klasse ist dafür zuständig, den eingegebenen Rechnungskorpus zu analysieren und diesen soweit zu verarbeiten, dass weitere Methoden auf diesen angewandt werden können. Die vorliegenden Dokumentdaten der Rechnungen werden zunächst in das System eingelesen, dann durch das 'Matching' identifiziert, um später die relevanten Informationen extrahieren zu können.
Die Klasse umfasst mehrere Methoden, die wichtigsten werden im Anschluss näher erklärt.

proofMatchings-Methode

Eine sehr grundlegende Methoden in der *'Parser'*-Klasse ist die *proofMatchings*-Methode. Diese Methode ist zur eigentlichen Identifikation der dem System bekannten Einträge durch 'Matching' implementiert worden. Hierbei wird der Dokumentkorpus sequentiell durchlaufen und jeweils nach einem aufgefundenen Muster überprüft, ob gespeicherte Einträge (Produkte) erkannt werden. Sofern eine Übereinstimmung festgestellt wird, speichert die Methode die jeweilige Stelle des im sequentiell vorliegenden Rechnungsdokument. Die Methode findet alle Stellen von "Matchen" der beschriebenen Art im Dokument und vermerkt für jedes dieser seine Stelle im Dokument. So werden die Stellen von potentiellen Produkten im Rechnungskorpus festgehalten. Dieser "Vermerk" ist für die darauf durchgeführte Extraktion von Bedeutung, da von nun an eine Stelle, von der eine Extraktion erfolgt, bekannt ist.

extractDataUntilToken und extractDataUntilNextMatch Methode

Liegen nach der *proofMatchings* Methode eine Anzahl von aufgefunden Produkten vor, so kann eine weitere Methode angesetzt werden.
Die beiden hier beschriebenen Methoden sind darauf ausgelegt, die eigentlichen Informationen aus dem jeweiligen Rechnungsdokumenten zu extrahieren. Grundsätzlich basieren die Methoden dabei auf zwei unterschiedlichen Vorgehensweisen.
Die *'extractDataUntilToken'* Methode, extrahiert die Informationen ab der zuvor im 'proofMatching' bestimmten Stelle im Rechnungsdokument, bis ein weiteres Muster aufgefunden wird. Dieses Muster ist "liefern und montieren" (Unterabschnitt 7.1.1). Es kennzeichnet in nahezu jeder Produktbeschreibung das Ende der jeweiligen Produktinformationen, so dass alle wichtigen Angaben zu einem Produkt im Ausgabefile vorliegen.
Bei der *'extractDataUntilNextMatch'* Methode werden die Informationen ebenfalls ab der durch ein Match aufgefundenen Stelle im Rechnungsdokument extrahiert. Hier erfolgt die Extraktion allerdings bis zu einem nächsten "Match", welches eine darauffolgende Produktbeschreibung auszeichnet. Bei dieser Methdode wurde zuvor kein explizites Muster "liefern und montieren" erkannt. Dies gewährleistet eine weitere Methode *'proofToken'*, die prüft, ob sich zwischen zwei durch Match aufgefunden Stellen im Rechnungsdokument das Muster "liefern und montieren" befindet. Nach dieser Routine wird anschließend die jeweilige Methode zur Extraktion aufgerufen, um die Informationen entsprechend zu extrahieren.
Die Extraktionsergebnisse liegen nach Anwendung der Extraktionsmethoden in dem Ausgabefile 'Produkt.txt' vor. In dem Ausgabefile wird ein jeweiliges aufgefundenes Produkt mit Zusatzinformationen als eigene Zeile eingetragen.

KAPITEL

NEUN

Evaluierung HMM

Zum Testen der Implementierungen gibt es eine Sammlung von eingescannten Rechnungen im PDF-Format, die die Firma Schottler bereitstellt und dazugehörige Textdateien, die das Ergebnis einer OCR-Software sind. Zur Evaluierung werden hierbei die im Kapitel 3 vorgestellten Standartmaße Genauigkeit (engl.: Precision) und Vollständigkeit (engl.: Recall) verwendet. Die *Genauigkeit* gibt an inwieweit die extrahierten Informationen korrekt sind. Die *Vollständigkeit* gibt an wie viele der tatsächlich vorhandenen Informationen gefunden wurden.

Art der Evaluierung Bei der im Kapitel 8 vorgestellten Implementierung wird eine summative Evaluierung durchgeführt. D h., dass bei dieser Evaluierung am Ende des Projektes das realisierte System mit den Projektzielen verglichen wird.

Anforderungen Die Anforderungen an eine Evaluierung sind einerseits Zuverlässigkeit, d. h., dass die Experimente wiederholbar sein sollten und andererseits Validität, d. h. experimentelle und reale Verhältnisse sollten übereinstimmen. Wenn diese beiden Voraussetzungen gegeben sind ist die Evaluierung gültig. In dieser Evaluierung ist die Zuverlässigkeit gewährleistet, wenn eine Evaluation auch mit anderen Testrechnungen beziehungsweise Dokumenten ähnliche Ergebnisse liefert. Aus diesem Grund wurden zusätzlich zu den Rechnungen der Firma Schottler auch Produktkataloge zur Evaluierung genutzt. Die Vali-

dität ist teilweise gegeben, da bei der Evaluierung real existente Rechnungen verwendet werden dafür aber der Dokumentenkorpus zu klein ist. Bei einem Korpus von 45 Rechnungen, welche bei der Klassifikation noch in zwei Mengen unterteilt werden müssen, ist einerseits nicht sichergestellt, dass die Ergebnisse aussagekräftig sind. Andererseits werden in diesen 45 Rechnungen nicht so viel verschiedene Rechnungen sein, wie es unter realen Bedingungen der Fall wäre.

9.1 Standardmaße

F-Maß Die Genauigkeit P (engl.: precision) bezeichnet den Anteil der korrekt extrahierten Produkte, im Vergleich zum Anteil aller Produkte die extrahiert worden sind. Eine hohe Genauigkeit bedeutet daher, dass fast alle gefundenen Produkte relevant sind. Die Vollständigkeit R (engl.: recall) bezeichnet den Anteil der korrekt extrahierten Produkte im Vergleich zu den insgesamt korrekten Produkten. Eine hohe Vollständigkeit bedeutet daher, dass fast alle relevanten Produkte extrahiert wurden. Aus diesen beiden Werten wird die Güte, also das F-Maß, eines IR-Prozesses[1] definiert.

$$F = 2 * (\frac{P * R}{P + R}) \tag{9.1}$$

9.2 Systemkonfiguration

Hardware Die CPU des Testsystems ist ein AMD Athlon(tm) mit 2 Prozessorkernen, 3,21 GHz Taktrate, 2 MB L2 Cache und einem 1066 MHz Front Size Bus. Der Arbeitsspeicher besteht aus zwei DDR2-800 Modulen im Dual-Channel-Modus und beträgt insgesamt 6 GB. Die Festplatte besteht aus einem 1 TB großen S-ATA Festplatte.

Software Auf dem Testrechner ist als Betriebssystem die 64 Bit Variante von Windows 7 Professionell installiert. Als Entwicklungsumgebung dient das Eclipse SDK1 in der Version 3.5.1 und das Java Development Kit (JDK) in der Version 1.6.

[1] Information Retrieval

9.3 Informationstextextraktion mittels HMM

Durch die Klasse *AutomaticInformationExtractor* wird die Informationsextraktion durch das HMM-System gestartet. Als Argument wird dieser Klasse der Datenpfad zur Konfigurationsdatei (engl.: Properties) übergeben, indem unter anderem der Datenpfad zum Testkorpus und Trainingskorpus mit angegeben ist. Es gibt zwei verschiedene Arten um den *AutomaticInformationExtractor* zu starten. In dieser Arbeit werden beide Varianten auf den Testkorpus angewandt. Die erste Variante ist eine simple HMM-Struktur, die vom System zur Verfügung gestellt wird, um mit dem Trainingskorpus zu trainieren und anhand dieser HMM-Struktur dann die Informationen aus dem Testkorpus zu extrahieren. Die zweite Variante besteht darin, dass das System die HMM-Struktur selbständig lernt und anhand dieser gelernten Struktur ebenfalls die Informationen aus dem Testkorpus extrahiert. Bei der ersten Variante ruft man dazu die *QuickExtractAndWrite*-Methode auf und übergibt ihr den Wert 1, um zu markieren, dass eine simple Struktur verwendet werden soll und nicht ein bereits vorhandenes Modell geladen werden soll. Bei der zweiten Variante dagegen ruft man die *ExtractandWriteData*-Methode auf und ruft anschließend ebenfalls die Methode *QuickExtractAndWrite* auf und übergibt dieses mal den Wert 0, damit für die Auswertung die gelernte Struktur verwendet wird. Die Ergebnisse, sprich die extrahierten Informationen, werden in einer Text Datei abgespeichert, die sich im jeweiligen Arbeitsverzeichnis (engl.: Workspace) befindet. Der Name der Datei ergibt sich aus der so genannten "Target Field" Option in der Konfigurationsdatei. Da in dieser Evaluierung Produkte extrahiert werden, wird dementsprechend die Datei "Produkt.txt" heißen. In der Konsole erfolgt die Ausgabe über die Detail der gelernten HMM. Dazu gehört auch die Zustandsübergangsmatrix und weiterhin wird für jeden Zustand des Modells angegeben, welche Wörter am meisten bei diesem Zustand auftreten bzw. emittiert werden (jeder Zustand verfügt über Emissionen). Über die *Extractor*-Klasse ist es möglich die Standardmassen zum Testkorpus automatisch berechnen zu lassen. Jedoch werden diese Werte in dieser Arbeit manuell berechnet werden, um ein genaueres Gesamtbild vom Programm zu bekommen.

9.3.1 Auswertung HMM mit simpler Struktur

Wie schon in der Einleitung dieses Kapitels erwähnt, gibt es verschiedene Möglichkeiten das Hidden Markov Modell System zu testen. In diesem Unterkapitel wird das HMM durch ein bereits im System vorhandenes simples Modell initialisiert und anhand von 20 Trainingsdokumenten trainiert. Die Trainingsdokumente bestehen aus 20 Rechnungen der Firma Schottler und einigen Produkt Katalogen. Das Ziel der Auswertung ist zu zeigen, welche Resultate das HMM-System, das ursprünglich für natürlich-sprachliche Texte auf englisch spezifiziert war, unter anderen Bedingungen beziehungsweise unter anderen Dokumenten (Rechnungen oder Katalogen) liefert. Wobei das Hauptaugenmerk auf den Tests mit den Rechnungen liegt. Bei den Katalogen wird lediglich getestet, ob es mit diesem HMM-System überhaupt möglich ist, Informationen auch von Katalogen extrahieren zu können.
Die Resultate die auf folgernder Abbildung 9.1 zu sehen sind, sind das Ergebnis aus 25 unbekannten Rechnungen der Firma Schottler (die nicht benutzt worden sind, um das HMM-System damit zu trainieren). Es wurden deshalb disjunkte Mengen von Rechnungsdaten aus dem Korpus benutzt, um das Verhalten des Systems auf unbekannte Dokumente zu beobachten. Schließlich entspricht dies auch realen Bedingungen.

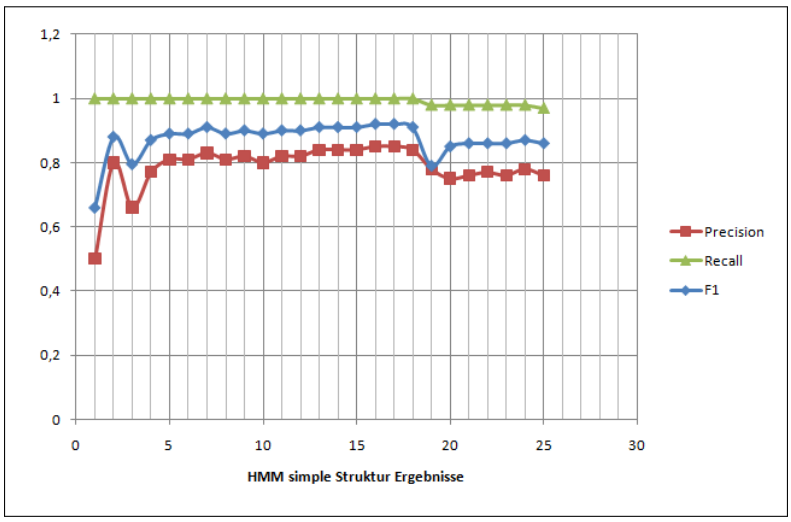


Abbildung 9.1: Resultate der Informationsextraktion, die durch ein HMM-System mit simpler Struktur durchgeführt worden sind

Man erkennt in Abbildung 9.1, dass die Werte für die Genauigkeit ab dem 5. Rechnungsdokument bis zum 19. Rechnungsdokument bei ca. 0.8 liegen. Es ist nur kurzzeitig eine Verringerung der Genauigkeit bei der 1. und 3. Rechnung zu verzeichnen. Ab der 20. Rechnung sinkt zwar der Wert für die Genauigkeit auf ca. 0.7, jedoch bleibt der Wert danach relativ konstant. Dort sieht man aber auch, dass bei der Extraktion der Produkte aus den Rechnungsdokumenten Fehler gemacht wurden, da dort die Genauigkeit zusätzlich zur Vollständigkeit abfällt. Der Wert für die Vollständigkeit ist ab dem 20. Rechnungsdokument gefallen, weil dort eine Produktbeschreibung existierte die im Verhältnis zu den anderen Produktbeschreibungen extrem lang war. Denn leider gab es im gesamten Trainingskorpus eine geringe Anzahl an Rechnungen, die eine derart lange Produktbeschreibung beherbergten. Denn das HMM-System kann nur so gut sein, wie es auch trainiert wird. Es kommt aber auch nicht immer auf die Größe des Trainingskorpus an, um die Güte des HMM-Systems zu verbessern. Denn bereits nach einer bestimmten Menge von Trainingsdaten erkennt das HMM-System die Muster die sich aus den Rechnungen ergeben. Wie man solche Muster in einem Trainingsdokument markiert wurde bereits in Kapitel 5 erklärt. Denn wie sich herausgestellt hat reicht bereits 1 Trainingsdokument aus, das eine Genauigkeit von 0.78 bei 10 unbekannten Rechnungsdokumenten erreicht werden kann. Jedoch muss dazu gesagt werden, dass zwar 33 von 42 Produkten erkannt worden sind, aber das System extrahierte dabei nicht alle Produktinformationen fehlerfrei beziehungsweise vollständig. So konnte es zum Beispiel sein, dass bei einem extrahierten Produkt, das ein "Waschtisch" sein könnte, die dazugehörigen Angaben wie zum Beispiel die "Farbe" des Waschtischs fehlten.

Nachdem das Hidden Markov Modell mit 20 Rechnungen trainiert worden ist und die Extraktion der Produkte erfolgreich ausgeführt worden ist, erfolgt danach folgende Ausgabe auf der Konsole:

Teil der Konsolenausgabe der AutomaticInformationExtractor-Klasse nach der Ausführung

```
Transition Matrix

Finish state is 0, start state is 1, (bkgrnd state is 2, * = target state).

       0    1    2    3    4    5   *6   *7   *8   *9   10   11   12
  0 1,00    -    -    -    -    -    -    -    -    -    -    -    -
  1    -    - 1,00    -    -    -    -    -    -    -    -    -    -
  2 0,01    - 0,95 0,05    -    -    -    -    -    -    -    -    -
  3    -    -    -    - 0,87 0,13    -    -    -    -    -    -    -
  4    -    -    -    -    = 1,00    -    -    -    -    -    -    -
  5    -    -    -    -    -    - 1,00    =    =    =    -    -    -
 *6    -    -    -    -    -    -    =    = 0,75 0,25    =    -    -
 *7    -    -    -    -    -    -    = 0,22 0,49 0,29    =    -    -
 *8    -    -    -    -    -    -    = 0,71 0,29    =    =    -    -
 *9    -    -    -    -    -    -    =    =    =    = 1,00    -    -
 10    -    -    -    -    -    -    -    -    -    -    - 0,87 0,13
 11    -    -    -    -    -    -    -    -    -    -    - 0,30 0,70
 12    -    - 1,00    -    -    -    -    -    -    -    -    -    -


****** State 0 ****** (FINISHSTATE) ****** No emissions ******

Transitions
-----------
--> 0: 1,000000


****** State 1 ****** (STARTSTATE) ****** No emissions ******

Transitions
-----------
--> 2: 1,000000
```

```
****** State 2 ****** (Background) ****** Type: UnseenEmitMap ******

Favorite words
--------------
,            0,067074    Wittlich     0,018293    Titel        0,012195
1            0,060976    BLZ          0,018293    montieren    0,006109
:            0,042683    D-54528      0,012195    1.229.38     0,006098
.            0,030488    10-12        0,012195    www.schottler-salmtal.de 0,006098
-            0,024391    Neuer        0,012195    Hot          0,006098
2            0,024391    0            0,012195    Posnr        0,006098
)            0,024391    Bahnhof      0,012195    Seite        0,006098
(            0,024391    GmbH         0,012195    Datum        0,006098
/            0,018293    Becker       0,012195    Line         0,006098
Schottler    0,018293    587          0,012195    28.06.2005   0,006098

P(seen) = 0,993976
P(xyz) = 0,000002  P(Xyz) = 0,000015  P(XY) = 0,000002  P(gh@bu.edu) = 0,000002
P(987) = 0,000011  P(9.8) = 0,000001  P(A8) = 0,000001  P(1975) = 0,000011

Transitions
-----------
--> 0: 0,006098
--> 2: 0,945121
--> 3: 0,048781

.........
.........
****** State 6 ****** Produkt ****** Type: UnseenEmitMap ******

All words
---------
Hansgrohe    0,125000
Kristallglas 0,125000
Eckventil    0,125000
Waschtischbefestigung 0,125000
Roehren-Geruchverschluss 0,125000
Wand         0,125000
Alape        0,125000
Mirror       0,125000
```

```
P(seen) = 0,900000
P(xyz) = 0,000001  P(Xyz) = 0,000016  P(XY) = 0,000001  P(gh@bu.edu) = 0,000001
P(987) = 0,000001  P(9.8) = 0,000001  P(A8) = 0,000001  P(1975) = 0,000001

Transitions
-----------
--> 6: 0,000000
--> 7: 0,000000
--> 8: 0,750000
--> 9: 0,250000
--> 10: 0,000000

.....
.....
```

In der Konsolenausgabe kann man deutlich die Zustandsübergangsmatrix der simplen HMM-Struktur erkennen. In dieser Matrix ist zu jedem Zustand die Übergangswahrscheinlichkeit zu einem anderen Zustand angegeben. Zusätzlich sind die Zielzustände (engl.: target states) mit einem Sternchen markiert. Zusätzlich wird noch separat zu jedem Zustand eine Liste mit den häufigsten Wörtern ausgegeben, die in diesem Zustand emittiert werden. Zustände die nicht mit einem Sternchen markiert sind stellen die so genannten "Hintergrundzustände" (engl.: background states) dar. Diese Zustände werden so trainiert, dass sie nur Wörter emittieren die nicht zu den Zielwörtern gehören (engl.: non-target tokens). In folgender Abbildung ist ein Teil des Graphen zu sehen, der sich aus der Zustandsübergangsmatrix ergeben hat.

Aufgrund fehlender bzw. niedriger Anzahl an Katalogdaten, die zur Verfügung standen, konnte für das HMM-System leider keine Vollständigkeits- und Genauigkeitswerte berechnet werden, da zu wenig Katalogdaten vorhanden waren. Es wurde lediglich getestet, ob es prinzipiell möglich ist, dass das HMM-System auch für Katalogdaten einsetzbar ist. Das Resultat ist, dass das HMM-System auch für Katalogdaten einsetzbar ist, es müssen aber dafür lediglich die Trainingsdaten dementsprechend bearbeitet werden (s. Entwurf Kapitel).

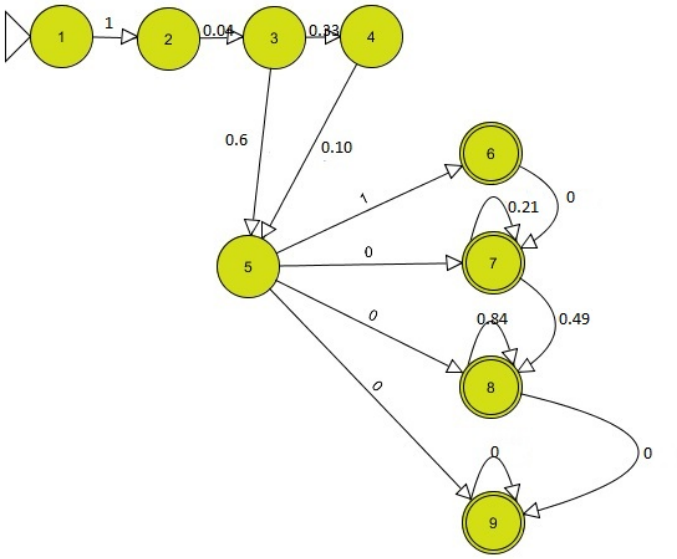


Abbildung 9.2: Teil der simplen Struktur, das zur Aufgabe hat, Produkte zu extrahieren

9.3.2 Auswertung HMM mit gelernter Struktur

Um die automatische Informationsextraktion mittels gelernter HMM-Struktur starten zu können, ist es vorher nötig eine neue HMM-Struktur zu lernen. Die HMM-Struktur wird einmalig gelernt und steht dann für weitere Tests zur Verfügung. Dazu ruft man als erstes die Methode *ExtractandWriteData* der *AutomaticInformationExtractor*-Klasse auf, um die neue Struktur nach dem Verfahren von Freitag & McCallum zu lernen. Die gelernte HMM-Struktur wird daraufhin zur weiteren Verwendung im jeweiligen Arbeitsverzeichnis unter dem Namen "Produkt.hmm" abgespeichert. Damit das System weiß wo sich die gelernte HMM-Struktur befindet, fügt man der Konfigurationsdatei in diesem Fall z. B. die Zeile "hmmFile=Produkt.hmm" hinzu. Der Name ergibt sich hierbei aus den so genannten "target fields". Nachdem die Struktur gelernt worden ist, wird nun die *QuickExtractAndWrite*-Methode aufgerufen mit dem Unterschied, das dieses mal eine 0 als Parameter übergeben wird. Das signalisiert nämlich das eine HMM-Struktur bereits vorliegt bzw. gelernt worden ist und nur lediglich in das System eingelesen werden muss. Der*AutomaticInformationExtractor*-Klasse wird dieses mal auch der Datenpfad zur Konfigurationsdatei übergeben. In

dieser Konfigurationsdatei steht jeweils der Datenpfad zum Testkorpus und Trainingskorpus. Der hierbei verwendete Testkorpus und Trainingskorpus entspricht genau demselben wie bei der Auswertung mit der simplen Struktur.
Nach Ausführung der oben genannten Schritte erhält man folgende Resultate wie man in Abbildung 9.3 sehen kann.

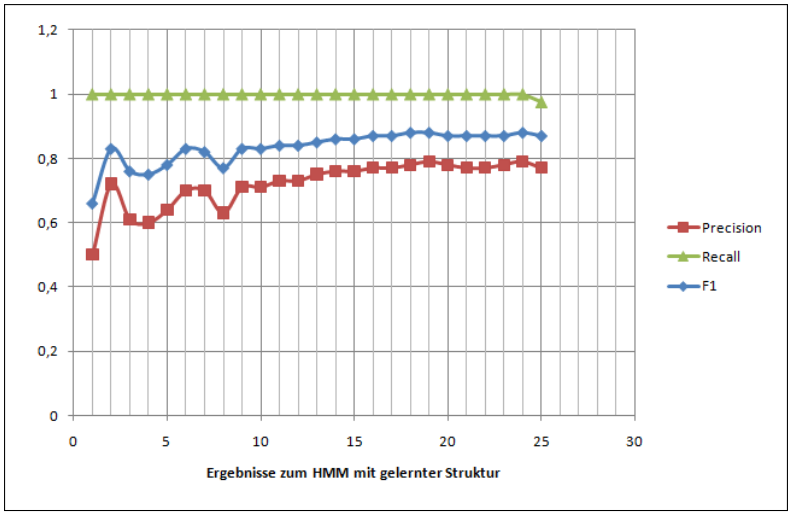


Abbildung 9.3: Resultate der Informationsextraktion, das durch ein HMM-System mit simpler Struktur durchgeführt worden ist

Man erkennt in Abbildung 9.3, dass die Werte für die Genauigkeit ab dem neunten Rechnungsdokument bis zum 25-ten Rechnungsdokument bei ca. 0.8 liegen. Es ist nur kurzzeitig eine Verringerung der Genauigkeit bei der ersten, zweiten, dritten, vierten und achten Rechnung zu verzeichnen. Dabei fällt dort der Genauigkeitswert auf 0.6. Der Wert für die Vollständigkeit ist anders als bei der simplen Struktur nicht ab dem 20. Rechnungsdokument gefallen, sondern fällt erst ab dem 25. Rechnungsdokument auf 0.97. Auch der F1 Maß bleibt ab dem 9. Rechnungsdokument relativ konstant bei ca. 0.9.
Es wurde festgestellt, dass das HMM-System mit gelernter Struktur die Informationen genauer extrahiert hat, als das HMM-System mit der simplen Struktur, welches bereits durch das System vorhanden war. Das heißt unter anderem, dass das System mithilfe der gelernten Struktur mehr Produkte extrahieren konnte, als das System mit der simplen Struktur. Aber auch wurde festgestellt, dass selbst bei denselben

extrahierten Produktinformationen, die die beiden Systeme extrahiert haben, die Informationen beim HMM-System mit der gelernten Struktur vollständiger waren. Demzufolge würde es bedeuten, dass zum Beispiel ein Produkt namens "Waschtisch" aus der Rechnung extrahiert worden ist, jedoch konnte es sein, dass bei dem HMM mit der simplen Struktur einige Informationen fehlten. Im Durchschnitt liegt der Wert für die Genauigkeit beim HMM-System mit simpler Struktur bei 78% und beim HMM-System mit gelernter Struktur bei 72%. Wie man anhand der Genauigkeit erkennen kann, verhält sich das System unterschiedlich in Bezug auf die Genauigkeit. Im F1-Score ist nicht so ein großer Unterschied zu verzeichnen. Denn dort liegt der F1 Wert für das HMM-System mit simpler Struktur bei 83.54% und beim HMM-System mit gelernter Struktur bei 83.36%.

Den F1 Wert betreffend unterscheiden sich die Ergebnisse der beiden Strukturen im wesentlichen nicht voneinander. Werden aber nun die Ergebnisse vom Entwickler des Systems Freitag & McCallum [17] hinzugezogen so ist ein etwas anderes Ergebnis bezüglich der beiden Strukturen zu verzeichnen. Aber man muss dazu sagen, dass dabei nicht die selben Dokumente zum Testen verwendet worden sind und sich das System dementsprechend anders verhält. Die Resultate die von Freitag & McCallum gemacht worden sind, sind folgender Tabelle 9.4 zu entnehmen. Dabei beschreiben die Überschriften der Tabelle die jeweils einzelnen Zielfelder (engl.: target fields) zu dem es jeweils eine HMM-Struktur gibt. Für jedes Zielfeld wird ein separates HMM trainiert. So gibt es beispielsweise in diesem Fall ein HMM mit gelernter Struktur für das Zielfeld "speaker","location" usw. Es standen hierbei zudem vier Dokumentenkorpora zur Verfügung, wo ein Korpus z. B. 485 Seminar-Ankündigungsdokumente einer Universität umfasste. Jedoch soll hierbei nicht weiter darauf eingegangen werden, da die Testergebnisse von Freitag und McCallum nur lediglich hierbei als ein Vergleich dienen sollten. Fakt ist, dass Freitag und McCallum hier [17] die Tests mit dem HMM-System für natürlich-sprachliche Texte auf englischer Sprache gemacht haben und man deshalb nicht direkt eine parallele mit dieser Arbeit ziehen kann.

	speaker	*location*	*acquired*	*dlramt*	*title*	*company*	*conf*	*deadline*	***Average***
Grown HMM	76.9	87.5	41.3	54.4	58.3	65.4	27.2	46.5	57.2
vs. SRV	+19.8	+16.0	+1.1	-1.6	—	—	—	—	+8.8
vs. Rapier	+23.9	+14.8	+12.5	+15.1	-11.7	+24.9	—	—	+13.3
vs. Simple HMM	+24.3	+5.6	+14.3	+5.6	+5.7	+11.1	+15.7	+6.7	+11.1
vs. Complex HMM	-2.1	+6.7	+7.5	-0.3	-0.3	+19.1	+0.0	-6.8	+3.0

Abbildung 9.4: Vergleich von F1 Werten zwischen HMM-System mit gelernter Struktur und anderen Struktur-Methoden. Die Zahlenwerte kennzeichnen um wie viel die Performanz gegenüber der jeweiligen Methode über bzw. unterlegen sind.

Teil der extrahierten Produktbeschreibungen aus einem Rechnungskorpus

```
1  Fussbefestigung an den Bodenradien Schallschutz gem ss DIN 4109
       / A1 ( in Kombination mit Kaldewei Schalld mmset )
       Allround
2  Duravit Konsolentraeger
3  Dombracht Montagesystem fuer 4-Loch-Wannen - randbatterie
4  Kaldewei Badewannen-Sonderfuss
5  Ausstattungsset fuer Ab - und Ueberlaufgarnitur mit Wasserzulauf
       , mit Drehrosette , Zulaufteil und Ventilkegel aus Messing
6  Funktionseinheit fuer Ab - und Ueberlaufgarnitur mit Drehgriff ,
       aus Kunststoff , Ventil aus Messing , mit unloesbarem ,
       gelenkigem Geruchverschluss , DIN 19545 , ohne Rosette und
       Stopfen
7  Abdeckplatte fuer Geberit UP-Spuelkasten mit Betaetigung von oben
       oder von vorne , mit 2-Mengen-Ausloesung , mit 2-
       Drueckerstangen
8  Funktionseinheit fuer Ab - und Ueberlaufgarnitur , mit
       Wasserzulauf durch den Ueberlauf - loerper aus Messing ,
       Zulaufteil mit Luftsprudler schwenkbar , auch fuer Wannen
       mit versenktem Ueberlauf , Anschluss 3 / 4 links oder
       rechts , Ventil aus Messing
9  Minderpreis wg . Kaldewei Classic Duo anstatt Vaioduo
10 Kaldewei Komfort-6-Eck-Badewanne 190x90cm , weiss
11 Hansgrohe Grundkoerper f r Absperrventil DN 15 mit Spindelventil
```

KAPITEL

ZEHN

Evaluierung CBR

10.1 Analyse der bestehenden Anforderungen

Die in der Implementierung umgesetzte Informationsextraktion, soll eine Extraktion von relevanten Daten des Dokuments gewährleisten. Relevante Informationen werden gemäß der schematischen Betrachtung des konzeptionellen Entwurfes definiert und entsprechen den Produkten und deren zugehörigen Produktbeschreibungen aus dem Mainframe des Systementwurfs (siehe Unterabschnitt 7.1.1). Bei Eingabe eines typischen Dokuments der Problemklasse, wird durch den Extraktionsvorgang eine Ausgabe generiert, die explizit die relevanten Informationen des eingegebenen Dokuments enthält. Die Ergebnisse werden anhand einer Reihe von Kriterien ausgewertet, welche die Güte der Extraktion durch eine Case-Based-Reasoning-Methodik auf der Probleminstanz 'Rechnung der Domäne Bauwesen' bestimmen.
Zum Test der Implementierung wird ein vom Projekt 'OnToBau' bereitgestellter repräsentativer Testkorpus der Domäne 'Bauwesen', welcher der Dokumentklasse 'Rechnung' zuzuordnen ist, verwendet. Der Testkorpus besteht aus einer Ansammlung gescannter Rechnungen der Firma "Schottler", zugehörige Textdateien dieser Dokumente sind durch die OCR-Software erstellte Resultate. Für eine genauere Betrachtung des Eingabekorpus kann auf das vorherige Kapitel, bezüglich der Analyse des Eingabekorpus (Abschnitt 7.1), verwiesen werden. Die Testphase wird primär unter optimalen Bedingungen durchgeführt, so sind die Testdaten des ersten Teils der

Evaluierung aufbereitet und damit fehlerfrei. Die Eingabedaten entstammen einer real existenten Menge von Dokumenten.

10.1.1 Evaluierungsmaße

Eine Bewertung der Evaluierung kann wie beim HMM mit Hilfe der üblichen Evaluierungsmaße des Information Retrieval vorgenommen werden. So liefern 'Precision' und 'Recall' allgemeine Auswertungen der Extraktionsergebnisse, die eigens dafür erstellten Kriterien, sollen eine Bewertung gemäß den Anforderungen unter Verwendung der Standardmaße ermöglichen. In einer Abwandlung zu den klassischen Definitionen der Retrievalmaße (Abschnitt 5.6), wird eine Anpassung zur Evaluierung einer Anwendung zur Extraktion von Informationen erreicht. Genauigkeit (engl.: precision) und Vollständigkeit (engl.: recall) werden wie folgt definiert:
Definition: Evaluierungsmaße der Anwendung 'Informationsextraktion mittels CBR'

- **Genauigkeit** P, ist ein Maß zur Angabe der Korrektheit der extrahierten Informationen.
- **Vollständigkeit** R, ist ein Maß, welches angibt, wie viele der tatsächlich zu extrahierenden Informationen gefunden wurden.

Die Berechnung der Maße erfolgt nach den bekannten Formeln, hierzu kann auf das Kapitel 9 zur Evaluierung des Hidden-Markov-Modells verwiesen werden.

Ein nahezu für eine Anwendung optimales Extraktionsverfahren maximiert die Werte für P und Q annähernd des positiven Extremwertes. Anhand der Berechnung der definierten Retrievalmaße wird die Bewertung des Verfahrens vorgenommen. Um eine genauere Einschätzung über die Erfüllung der gestellten Anforderungen an das System vornehmen zu können, kann eine Auflistung von Bewertungskriterien aufgestellt werden. Die unterschiedlichen Kriterien basieren auf den Maßen, Genauigkeit und Vollständigkeit.
Insofern kann ein Test des Verfahrens gemäß der aufgestellten Kriterien ausgewertet werden, indem die Implementierung anhand definierter Bewertungsaspekte getestet wird. Die Evaluierung ermöglicht so eine qualitative Bewertung und kann dementsprechend die Eignung des Systems zur Anwendung der Informationsextraktion auf einer strukturierten Rechnungseingabe bemessen.

10.1.2 Evaluierungskriterien

Die Evaluierunskriterien, die eine qualitative Auswertung des Verfahrens ermöglichen, teilen die Auswertung in drei evaluierenden Testphasen. Die Implementierung wird infolgedessen anhand drei grundsätzlicher Tests ausgewertet. Mit der durchzuführenden Testreihe im Gesamten werden folgende grundsätzliche Kriterien zur Bewertung einer Eignung abgedeckt. Die Ergebnisse der einzelnen Kriterien geben jeweils Ausschluss über ihre Genauigkeit und Vollständigkeit.
Evaluierungsskriterien: Informationsextraktion CBR-Invoice

- *Kriterium 1*: Test des Systems mit 20 initialen Fällen in der Fallbasis auf einem Testkorpus bestehend aus 25 repräsentativen domänenspezifischen Dokumenten.

- *Kriterium 2*: Testphase auf minimaler Fallbasis aus drei Fällen mit Test auf einen Testkorpus von zehn Dokumenten. Ziel ist die Abschätzung des Einflusses der initialen Fallbasis auf die Extraktion.

- *Kriterium 3*: Test auf zehn komplett unaufbereitete Eingaben, der Testkorpus besteht aus Dokumenten, die nicht den optimalen Testbedingungen entsprechen. Die Dokumente liegen in fehlerbehafteter und unregelmäßiger sequentieller Reihenfolge vor, so dass die Muster zur Extraktion sich nicht immer an den exakten Stellen befinden. Die Fallbasis hat zehn initiale Fälle.

Mit dem ersten Kriterium lässt sich die allgemeine Fähigkeit des Systems aus dieser Art von Dokumentkorpus Informationen zu extrahieren, testen. Dieses wird in der Evaluierung des Entwurfs ein Hauptkriterium sein und die Funktionalität des Systems. Damit kann eine Bewertung erfolgen, welche angibt in wie weit es gelungen ist, das Framework auf die Problemstellung anzupassen, um mit der Probleminstanz 'Rechnungsdokument' umgehen zu können. Das Kriterium erfüllt damit die Anforderung einer allgemeinen Durchführbarkeit eines Extraktionsprozesses des entworfenen Systems.

Eine minimale Anforderung an den Entwurf kann anhand des zweiten Kriteriums getestet werden. Hierbei kann folglich die aus 'Hauptkriterium 1' abgeleitete Eignung zur allgemeinen Extraktion verfeinert werden, indem von einer unter realistischen Bedingungen minimalen Fallbasis ausgegangen wird.
Ein Test hinsichtlich des letzten Kriteriums kann abschließend die Anforderung der Zuverlässigkeit des Systementwurfs aufzeigen. In diesem Fall kann die Wiederholbarkeit des Experiments nachgewiesen werden, indem eine Evaluation mit einem unaufbereiteten Testkorpus annähernd ähnliche Ergebnisse liefert.

Die Anforderung der Validität wird unter Beachtung des letzten Kriteriums ebenfalls in die Evaluierung, zumindest in Ansätzen, einbezogen. Eine Evaluierung kann somit den Nachweis liefern, dass neben den experimentellen Ergebnissen unter optimalen Bedingungen ebenso vergleichsweise reale Verhältnisse, durch unaufbereitete Eingaben, annähernd akzeptable Ergebnisse hinsichtlich der Genauigkeit und Vollständigkeit bringen. Wesentlich zu beachten ist jedoch, dass die Aussagekraft bezüglich der Validität abgeschwächt angenommen werden muss, da insgesamt ein nur sehr kleiner Dokumentenkorpus an Rechnungen vorliegt und dieser nur halbwegs reale Bedingungen für ein regelbasiertes Verfahren wie CBR garantiert.

Die Aufstellung der Kriterien ermöglicht die Durchführung und Auswertung einer 'Summativen Evaluation'[1], so kann das Ergebnis des Verfahrens zu einem späteren Zeitpunkt mit den Ergebnissen des zweiten Konzepts der Implementierung des Hidden-Markov Modells verglichen werden. Hier kann ein Vergleich der Systeme mit Auswertung auf Eignung durchgeführt werden.
Infolgedessen wird eine zusätzliche, für die Evaluierung von Information Retrieval Systemen typische Auswertung hinsichtlich der Effektivität mit Hilfe einer Kosten-Nutzen-Betrachtung, im Rahmen dieser Evaluierung nicht erfolgen. Diese hätte die Qualität der Lösung im Verhältnis von Aufwand und Ergebnis bewertet.

[1] auch Ergebnisevaluation: Ist ein Evaluationsansatz, der die Bewertung eines Erfüllungsgrades von vollständig entwickelten Systemen vornimmt

10.2 Durchführung

Das Programm wird wie in der Klassenbeschreibung zum CBR (Abschnitt 8.2) beschrieben ausgeführt, um Tests gemäß der aufgestellten Kriterien durchzuführen. Wichtig ist, dass die Testdokumente, exakt nach den Anforderungen des Frameworks zum Test eingebunden werden. Das Testdokument darf, an Anfang und Ende keine leere Zeile enthalten und muss der festgelegten Repräsentation entsprechen.

10.2.1 Test auf Lernfeatures

In einem ersten Testablauf wird gezeigt, dass die im Entwurf konzipierten 'Lernfeatures' funktionsfähig sind. Dazu werden jeweils manuell erstellte Testdokumente verwendet, die aufzeigen sollen, ob die Features des Systems bezüglich *Vervollständigung* eines Falles bzw. *Erkennung* eines unbekannten ersten Produktes, in der Rechnung einsetzbar sind.

Lernfeatures Um die Fähigkeit des Systems unbekannte Produkte eines Dokuments zu erkennen, werden Dokumente der folgenden Art erstellt:

1. unbekannter Eintrag: z. B. "*Badtisch*"
2. bekannter Eintrag: z. B. "Waschtischbefestigung"
3. bekannter Eintrag: z. B. "Eckventil"

Das System erkennt das Produkt "Bauschutt" als einen Waschtisch und baut diesen Fall in die Fallbasis ein.
Die Vervollständigung durch zusätzliche Produktangaben, kann mit der folgenden Produktart nachgewiesen werden:

1. bekannter erster Eintrag: z. B. "*Waschtisch*"
2. bekannter weiterer Eintrag: z. B. "Waschtischbefestigung"
3. bekannter weiterer Eintrag: z. B. "Eckventil"

Diese Art von Dokument wird die Angabe möglicher zusätzlicher Produkte im Produktfile 'Produkt.txt' testen. Das System greift auf Produkte die in der Fallbasis gespeichert sind zurück und gibt diese mit den extrahierten Daten aus.
Die Tests auf jeweilige Lernfeatures des Systems werden mit jeweils vier erstellten Dokumenten durchgeführt.

10.2.2 Test der Kriterien

Um das System der Implementierung einer CBR-Methodik nach dem erstem Kriterium zu prüfen, muss zur Testphase einmalig ein geeigneter "Startpunkt" für das System definiert werden. Dieser beschreibt den Status des Systems zu Anfang der Testphase. Um weiteres Wissen, im Sinne von extrahierten Informationen, akquirieren zu können, muss dem System ein gewisser Grad an Vorwissen gegeben werden. Das CBR-Framework wird mit einer Anzahl von Fällen in der Fallbasis initialisiert (Unterabschnitt 7.3.5), um die Methode des Fallbasierten Schließen ausführen zu können. Gerade beim Retrieval, dem Auffinden von Fällen, ist ein Rückgriff auf vorhandene Fälle zum Lösen des Problems notwendig.

Kriterium 1 Wie im konzeptionellen Entwurf beschrieben, wird eine Initialisierung des System vorgenommen. Dazu wird die initiale Fallbasis aus einer festgelegten Anzahl von 20 Fällen, dies ist eine Teilmenge des repräsentativen Textkorpus, aufgebaut. Dies geschieht nach bekanntem Prinzip unter Verwendung einer manuell definierten Matching-Liste (Unterabschnitt 7.3.3), die durch das System automatisiert eine initiale Fallbasis erstellt. So wird eine Anzahl von Dokumenten als Fälle in der Fallbasis zum Starten des Testprozesses abgelegt. Die verwendeten Dokumente aus der Menge des Textkorpus werden ähnlich dem 'Cross-Validation-Prinzip'[2], zwar nicht als Dokumente zum Training des Lernprozesses genutzt, sondern sind die Wissensgrundlage des Systems.
Der Vorgang der Initialisierung der Fallbasis ermöglicht darauf die Evaluierung der Implementierung. Alle folgenden Dokumente der Eingabe werden nacheinander zur Auswertung der Güte des Extraktionsvorgangs beitragen. So kann für jedes Doku-

[2]Der Textkorpus wird aufgeteilt in Trainings- und Testmenge

ment eine eigene Analyse der Extraktion vorgenommen werden, welche vermerkt, wie viele der relevanten und damit zu extrahierenden Informationen aus dem Testdokument erkannt und auch wirklich im Ausgabefile extrahiert werden. Ausgewertet wird anhand des Vergleiches der manuellen Feststellung von tatsächlichen Informationen eines Dokuments die extrahiert werden müssten und den resultierenden Extraktionsdaten der Ausgabe. Mit diesen Werten kann eine Evaluation gemäß der Kriterien bezüglich *Genauigkeit* und *Vollständigkeit* erfolgen. Die beiden Maße liefern eine Bewertung des Extraktionsvorgangs für die Anzahl von 25 Dokumenten. Das Ergebnis des gesamten Prozesses lässt sich in eine 'Recall-Precision-Graphik' eintragen.
Zu erwähnen ist, dass jedes weitere Testdokument die Fallbasis erweitert, und als gelernter Fall in die Fallbasis eingebaut wird. Diese Tatsache wird sich auf die Testergebnisse auswirken. Es ist anzunehmen, dass eine zunehmende Fallbasisgröße die Ergebnisse der Standardmaße beeinflussen wird.

Kriterium 2 Auf Grund dieser Vermutung kann ein Test auf das zweite Kriterium durchgeführt werden. Hierfür wird nach automatisiertem Prinzip die initiale Fallbasis mit einer minimalen Größe hergestellt. Ausgehend von einer minimalen Anzahl von drei Fällen in der Fallbasis, können durch eine kleine Testreihe, erneute Auswertungen gemäß der Retrievalmaße erfolgen. Ziel wird die Bestimmung der Werte und eine damit verbundene Einschätzung des Einflusses der Fallbasisgröße auf den Extraktionsvorgangs sein. Es gilt zu klären, welchen Faktor die Abdeckung der Fallbasis in dieser überlieferten Implementierung und Umsetzung auf den Extraktionsprozess im Ganzen hat.

Kriterium 3 Bei Durchführung der Testphase zum letzten Kriterium, kann eine absschließende Bewertung der allgemeinen Anwendbarkeit der Implementierung geliefert werden. Für diesen Zweck wird der Aufbau der initialen Fallbasis mit aufbereiteten Dokumenten, die Durchführung der darauffolgenden Testphase mit ausnahmslos unaufbereiteten Dokumenten, vollzogen. Es werden zehn Dokumente in die Fallbasis initialisiert, und der Test erfolgt auf zehn unaufbereitete möglicherweise fehlerbehaftete Rechnungen. Die Ergebnisse werden wiederum anhand der Gütemaße Genauigkeit und Vollständigkeit ausgegeben und zur Auswertung in ein Verhältnis

zu den Ergebnissen der "optimalen" Testverhältnisse der Testphase zu dem ersten Kriterium gesetzt. Die Auswertung dieses Testkriteriums liefert Aufschluss über die Notwendigkeit einer Dokumentaufbereitung und einer Korrektheit des Eingabekorpus. So kann eine Erkenntnis über die Eignung einer Übertragbarkeit des Systems auf ähnliche aber auch komplett divergent strukturierte Eingaben abgeleitet werden.

10.3 Ergebnisse und Auswertungen

Im Folgenden werden die Ergebnisse zu den verschiedenen Auswertungskriterien geliefert. Die Ergebnisse werden in diversen Precision-Recall-Graphen aufbereitet. Die Veränderung der Werte kann in Abhängigkeit zu der Anzahl von getesteten Dokumenten betrachtet werden. Eine qualitative Auswertung der Ergebnisdaten wird im Anschluss durchgeführt.

Ergebnisse der Lernfeatures Das System wurde mit vier erstellten Testdokumenten auf jeweils eines der Lernfeatures getestet. Beim Test *Erkennung* eines unbekannten Produktes, konnte bei drei von vier Testdokumenten ein richtiges erstes Produkt zugeordnet werden und der gelernte Fall in die Fallbasis abgelegt werden. Abbildung 10.1 zeigt ein solches Ergebnis im Produktfile des Systems.

```
Aus Extraktion gefundene Produkte:
  Waschtischbefestigung = Satz Waschtischbefestigung  M10x140mm  liefern
  Eckventil = Stck Eckventil  1/2  verchromt  selbstdichtend  mit  Rosette  liefern
Waschtisch= Stck Aloha  Badtisch  rechteckig  80x49cm  weiss  m.  Waschtischunterbau  mit  Auszug  liefern
Zusätzliche mögliche Produkte:
```

Abbildung 10.1: Erkennung im Produktfile

Eine Vervollständigung durch Produktangaben eines ähnlichen Falles, konnte bei allen 4 Testdokumenten erfolgen. Ein Dokument wurde fehlerfrei vervollständigt, bei den übrigen Dokumenten ergaben sich Fehler durch einzelne ausgelassene Produkte oder Duplikate in der Menge der zusätzlichen Produkte. Die Vervollständigung durch geeignete Produkte ist in Abbildung zu sehen.
Die hier gelieferten Ergebnisse werden nicht bezüglich der Gütemaße ausgewertet, da die Lernfeatures nur bedingt zu den Extraktionsergebnissen beitragen. In erster Linie

Abbildung 10.2: Vervollständigung im Produktfile

sollte die Funktionalität des Systems im Umgang mit noch unbekannten Produkten geprüft werden.

Kriterium 1 Nach der Testphase mit 25 aufbereiteten Rechnungsdokumenten, die nacheinander in das System eingegeben wurden, konnte für jedes Dokument einzeln seine Precision- und Recall-Werte bestimmt werden. Die F1-Meassure wurde per Formel (Abschnitt 9.1) bestimmt. So kann eine Graphik angegeben werden, in welcher für jedes Dokument die Werte der Extraktion eingetragen werden. Abbildung 10.3 zeigt die Ergebnisse der Extraktion von 25 Dokumenten mit einer Implementierung nach CBR-Ansatz.

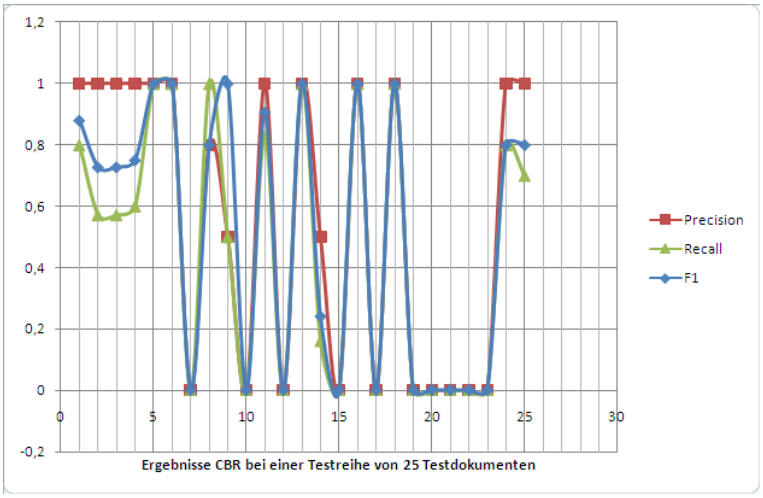


Abbildung 10.3: Ergebnisse CBR mit 25 Testdokumenten

Die durchschnittlichen Recall- und Precision-Werte des Systems sind Precision 0,55 und Recall 0,46. Wie zu erkennen ist, liefert die Extraktion auch Ergebnisse die an den Extremwerten liegen. In diesen Fällen wurden entweder alle relevanten Produkte des Produktes gefunden, oder in bestimmten Fällen keines der Produkte. Precision

und Recall liegen an diesen Stellen exakt auf 1 bzw. 0. Dies ist darauf zurückzuführen, dass zum einen nach Aufbau durch die initialen Matchingliste aus einer 20 Fall großen Fallbasis extrahiert wird, die einen Großteil der Produkte schon in Fällen gespeichert hat. Eine Erkennung der Informationen ist daher sehr wahrscheinlich und exakt. Zum anderen ist es so, dass im Testkorpus, da möglichst reale Bedingungen nachgebildet wurden, in einem Dokument kein Produkt zu extrahieren ist und dieses damit ausschließlich aus unrelevanten Informationen besteht. Eine weitere Erklärung eines "Null-Ergebnisses" bezüglich Recall und Precision, ist das vorkommen von Dokumenten, die ausnahmslos aus für die Fallbasis unbekannten Produkten bestehen. Für diese Problematik wurde das Lernfeature *Erkennung* erstellt, jedoch erwies sich die Verarbeitung von Dokumenten aus komplett unbekannten Produkten als anfällig. Hierauf wird in der anschließenden Auswertung genauer eingegangen. Des Weiteren gab es trotz Aufbereitung des Testkorpus ein Dokument an dem keine Extraktion durch das Systems möglich war. Als Folge davon brach das Programm ab. Dieses Ergebnis wurden als negativer "Aussreisser" mit zwei Nullwerten festgehalten. Bei 25 Testdokumenten konnte ein relativer Anteil an Dokumenten keines der gewünschten Informationen liefern.

Auswertung Festzuhalten ist, dass sofern Produkte im Korpus erkannt werden, diese auch insgesamt mit relativ hohen Vollständigkeits- und Präzisionswerten Informationen gewonnen werden können.
Aus den Ergebnissen sind folgende Regelmäßigkeiten abzuleiten: Sofern im System wenige von vielen möglichen Produkte aus einem Dokument erkannt werden, ist die Wahrscheinlichkeit groß, dass diese Erkannten auch falsch erkannt werden. Sofern Null Produkte erkannt wurden, sind diese folglich auch komplett nicht korrekt extrahiert. Entsprechend sind die Werte für Genauigkeit und Vollständigkeit gleich und entsprechen Null. Eine wichtige Erkenntnis der Testreihe ist, dass bei der Extraktion ab einem Recall-Wert von 0,5714 das System 100 prozentige Präzisionswerte erreicht. Werden Produkte aus Dokumenten mit Werten über 50 Prozent, also mit einer "halben Vollständigkeit" extrahiert, so werden diese auch nahezu ausnahmslos korrekt gewonnen.

Kriterium 2 Der Test auf 'Kriterium 2' erfolgt von einer minimalen Fallbasis. Die zehn zufällig gewählten Testdokumente werden nacheinander getestet, so dass die Fallbasis mit jedem getesteten und danach eingebauten Fall erweitert wird. Es ist zu erwarten, dass sich die Ergebnisse für zumindest ein Maß bei zunehmender Fallbasisgröße verbessern, da das System mit jedem gelernten Fall seine Performanz verbessert.

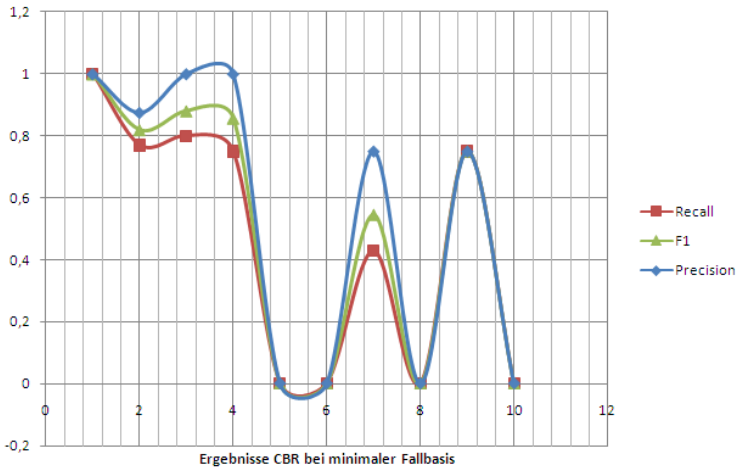


Abbildung 10.4: Ergebnisse CBR mit einer minimalen Fallbasis

Ausgehend von einer minimalen Fallbasis, werden bei einem Testkorpus von zehn Dokumenten ein durchschnittlicher Präzisionswert von 0,53 erreicht, während der durchschnittliche Vollständigkeitswert bei 0,4492 liegt.
Zu Anfang ist zu erkennen, dass die ersten vier Dokumente mit jeweils hohen Werten über 0,7 fast vollständig und mit einer hohen Präzision erkannt werden. Durch weitere zum Teil komplett nicht extraktionsfähige Dokumente werden die zunächst guten Ergebnisse abgeschwächt, bis sich die Werte des Systems wieder in der Nähe der Durchschnittswerten von Precision und Recall einstellen. So treten wie schon beim Test auf 25 Testdokumente extreme "Auschläge" des Systems aufgrund nicht erkennbarer Dokumente auf. Precision und Recall gehen dann auf den Wert Null.

Auswertung Wie aus den Ergebnissen zu schließen ist, hat die Größe der Fallbasis bei dem hier durchgeführten Testumfang keinen allzu großen Einfluss auf die Ergebnisse der Extraktion. Das Vorhandensein einer Fallbasis, die einen weiten Eingabebereich abdeckt (engl.: coverage) ist zwar im Allgemeinen, wie auch bei dieser konkreten Umsetzung vorteilhaft, jedoch ist dies bei einem begrenzten Testkorpus aus Rechnungsdokumenten sehr geringfügig feststellbar.

Kriterium 3 Das System konnte acht von zehn unaufbereitete Testdokumente, ohne Fehler, die zu einem Abbruch des Prozesses führen, verarbeiten. Zwei der Dokumente haben dazu geführt, dass das System bei dieser Art von Eingabe nicht terminiert, diese Ereignisse wurden ebenfalls als Einträge mit Werten 0 für Genauigkeit und Vollständigkeit vermerkt. Zu erkennen ist, dass bei etwas mehr als der Hälfte der Dokumente Informationen nach den im Folgenden gelieferten Ergebnissen extrahiert werden konnten. Bei exakt drei Dokumenten konnten keine relevanten Produkte trotz Verarbeitung gefunden werden. Hierbei handelte es sich wiederum um zwei Dokumente die vollständig aus unbekannten Produktangaben bestanden haben.
Auf Grund der vorliegenden Testergebnisse konnte eine Average-Precision von 0,435 berechnet werden. Der Average-Recall liegt bei 0,41. Diese Werte sind geringfügig unter den der aufbereiteten Dokumente.

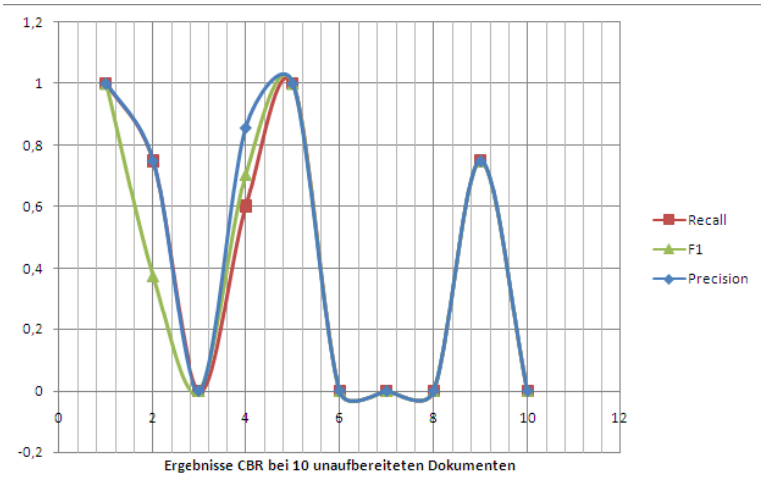


Abbildung 10.5: Ergebnisse CBR mit 10 unaufbereiteten Testdokumenten

Auswertung Das System liefert, wie auch schon bei dem Test auf 25 Dokumente annehmbare Ergebnisse auf unbekannte Dokumente. Die Ergebnisse liegen bei durchgeführter Extraktion in dem Bereich der Werte aus der großen Testreihe. Werden Vollständigkeitswerte von mindestens 0,6 erreicht, so kann ein mindestens gleich guter Präzisionswert bei der Extraktion von unaufbereiteten Dokumenten erzielt werden. Sofern Produkte aus der Fallbasis im Dokument aufgefunden werden, können diese in den meisten Fällen mit einer Precision von mindestens 0,75 extrahiert werden. Einzig die Problematik der komplett unbekannten Produktangaben und Dokumente, die zu Abbruch des Prozesses führen, bewirkten eine Verschlechterung der Ergebnisse bezüglich Genauigkeit und Vollständigkeit.

10.4 Bewertung der Ergebnisse

Dem System zum CBR-Ansatz unter Verwendung des Frameworks *FreeCBR* mit seiner Schnittstelle *CbrInvoice* konnte eine Eignung zur Extraktion von Produkten und Produktinformationen der speziellen Probleminstanz "Rechnung" nachgewiesen werden. Auswertungen gemäß der aufgestellten Eignungskriterien ergaben, dass das System Auswertungsergebnisse im typischen Bereich von Retrival- oder Extraktionsverfahren liefern kann. Eine besondere Eignung stellte sich im Umgang von Dokumenten die bekannte Produkte liefern, also in der Fallbasis vorliegen, heraus. Sowohl

der Test auf unaufbereitete Dokumente, als auch der Umgang mit einer minimal kleinen Fallbasis ergab, dass das System auch unter diesen "Ausnahmebedingungen" zur Extraktion aus Rechnungen durchaus geeignet ist (vgl. Abschnitt 10.3). Wie der Auswertung zu entnehmen ist, unterscheiden sich die Werte zu Genauigkeit und Vollständigkeit unter diesen Bedingungen, im Vergleich zum Test unter optimalen Bedingungen nur um wenige Prozentpunkte.
Wie aus den Ergebnissen zu allen Kriterien zu entnehmen ist, rufen Dokumente die aus vollständig für die Fallbasis unbekannten Produkten bestehen, Ungenauigkeiten im System hervor. Dies ist anhand der Präzisionswerte und Vollständigkeitswerte bei diesen Eingabetypen exakt ablesbar. Diese Problematik verschlechtert die allgemeine Performanz der Extraktion. Es könnte bei einer Behandlung dieses Problems im Sinne einer Modifikation im System eine Verbesserung der Extraktionsfähigkeit erzielt werden, die sich auf die durchschnittlichen Gütemaßwerte auswirkt.

Modifikation zur Problembehandlung Eine Möglichkeit zur Behandlung dieses Problems, um eine Verbesserung der Genauigkeit und Vollständigkeit herbeizuführen, wäre die Einführung eines weiteren dritten *Lernfeatures*. Dieses Feature entspräche einer Ausnahmebehandlung von Dokumenten mit vollständig unbekannten Produkten, die nicht durch die bestehenden Features gelöst werden konnten. Das System würde wie gewohnt nach Erkennung der im Konzept beschriebenen Muster des Dokuments (Unterabschnitt 7.1.1), bisher undefinierte Produkte oder konkrete Produktnamen die dem Muster folgen, extrahieren. Diese unbekannten Einträge, bisher nicht vorliegender Produkte, werden in gewisser Weise "blind" extrahiert, da nicht auf Vorhandensein eines Falles in der Fallbasis geprüft wird. Eine Extraktion dieser Art wird mit einer hohen Wahrscheinlichkeit auch richtige zuvor noch unbekannte Produkte bzw. deren Bezeichnung aus dem Dokument gewinnen. Die als unbekannt extrahierten Produkte liefern einen neuen Fall gemäß des konzeptionellen Formalismus eines Falles und sofern dieser in die Fallbasis abgelegt wird, wurde ein vollständig unbekanntes Dokument gelernt und liegt zur Problemlösung nachfolgender Anfragen vor.
Eine Extraktion nach Ausnahmebehandlung wird Extraktionsergebnisse zu dem ne-

gativen Extremwert Null beheben. Die Gütemaße liefern von nun an theoretisch keine "Nullwertergebnisse" und je mehr Fälle dieser Art dem Sytstem in der Fallbasis vorliegen, desto wahrscheinlicher ist ein Auffinden eines nunmehr nicht mehr unbekannten Dokuments.

Ein weiteres Problem, welches sich aus der Betrachtung der Ergebnisse nach den Testdurchläufen ergeben hat, ist die erkennbare limitierte Aussagekraft bzw. wenig repräsentative Eignung der vorliegenden Testdaten aus dem Textkorpus.

Problem: Limitierte Testdaten Im repräsentativen Dokumentenkorpus befindet sich eine gewisse Anzahl unbrauchbarer Dokumente. Diese sind zum Teil fehlerhaft, so dass nach Verarbeitung durch die OCR-Software beispielsweise in mehreren Fällen keine Produktinformationen ausgegeben wurden. Diese Dokumente beinhalten ausschließlich unrelevante Informationen, die zur Extraktion nicht geeignet sind. Des Weiteren liegen in der Menge des Textkorpus unter anderem Dokumente vor, die Duplikate im Sinne der Dokumentart "Angebot" und "Rechnung" eines gleichen Vorgangs sind. Daraus resultiert, dass in zwei Testdokumenten exakt gleiche Produkteinträge geliefert werden. Gerade bei einer großen Testmenge aus 25 Dokumenten macht sich das Fehlen alternativer Testvorlagen bemerkbar. Bei den Tests mussten nahezu alle vorliegenden Testdaten verwendet werden. Die wenigen geeigneten Testdaten sind zusätzlich nach Betrachtung der Kontexte, die diese beschreiben, extrem unterschiedlich. Es ergeben sich in der Menge des Testkorpus nur sehr wenige Produktzusammenhänge, in denen einzelne Dokumente nur ganz bestimmte Kontexte behandeln.
Eine weitere Einschränkung, die den repräsentativen Charakter des Testdokuments abschwächt ist die Tatsache, dass insgesamt sehr wenig Produktangaben in den Dokumenten selbst enthalten sind. Die kleine Anzahl der zu extrahierenden Produkte, führt in vielen Fällen dazu, dass unverhältnismäßig oft alle Produkte oder kein einziges Dokument aufgefunden werden. So ist festzustellen, dass sich die Präzisionswerte und Vollständigkeitswerte häufig an den Extremwerten Null und Eins befinden. Die Spanne bei der Extraktion zwischen vollständig extrahierten Dokumenten und gar nicht extrahierten Dokumenten ist sehr gering.

10.4.1 Fazit

Das umgesetzte System ist auf eine konkrete Problemstellung ausgelegt. So konnten bei der Realisierung des Systems bestimmte Regelmäßigkeiten des Eingabedokuments (Unterabschnitt 7.1.1), wie auftretende Muster welche die Extraktion vereinfachen, ausgenutzt werden. Diese Vereinfachung hat eine nur sehr begrenzte Eignung und damit Anpassungsfähigkeit auf andere Probleminstanzen, das heißt Dokumente anderer Struktur, zur Folge.
Die Lernfähigkeit des Systems, ist ohne Modifikation des hier vorgeschlagenen dritten Lernfeatures, relativ gering. Diese Feststellung ist anhand der Ergebnisse zu allen Kriterien abzuleiten. So ist keine ersichtliche Verbesserung bei Betrachtung der Ergebnisse bei Anstieg der Dokumentanzahl bei den eher kleinen und schwach repräsentativen Testreihen festzuhalten. Das System lernt zwar mit jedem neu hinzugefügten Fall, jedoch macht sich dies nur sehr unwesentlich in den Ergebnissen bemerkbar. Dies kann auf die begrenzte Aussagekraft der wenig repräsentativen Testdaten zurückgeführt werden. Die Testphasen wurden entweder mit sehr ähnlichen Dokumenten, die bereits in bekannten Produkten enthalten waren oder mit gänzlich vom Kontext nahezu unbekannten Dokumenten durchgeführt.
Die CBR-Methodik hat bei der hier durchgeführten Umsetzung nur bedingten Einfluss auf die Ergebnisse des Extraktionsvorgangs. Zu vermuten ist, dass mit Einbindung des dritten Lernfeatures zum Handling der vollständig unbekannten Produktangaben, eine signifikante Verbesserung zu erkennen ist. Der Umgang mit der Fallbasis undefinierter Produkte, stellte sich im Nachhinein als sehr grundlegend für die Performanz des Systems heraus. Aus zeitlichen Gründen konnte eine nachträgliche Modifikation im Rahmen dieser Arbeit nicht erfolgen, da eine Einbindung in das bestehende Framework *FreeCBR* komplizierter war als zunächt angenommen.
Abschließend kann festgehalten werden, dass die allgemeinen Extraktionsergebnisse insgesamt zufriedenstellend sind, was die Ergebnisse zu allen drei Kriterien gemäß ihrer durchschnittlichen Gütemaße belegen. So konnte sogar die Eignung auf komplett unaufbereitete Eingabedaten nachgewiesen werden.

KAPITEL

ELF

Ontologiepopulation

Die Informationsextraktion ist eine Technik, die einem Ontologieexperten während der Population der Ontologie und des Prozess der Instandhaltung (engl.: maintenance) behilflich sein kann. Die Extraktion von Informationen kann als Aufgabe angesehen werden, vordefinierte Entitäten aus Text zu ziehen, um vordefinierte "Slots" in Klassen zu befüllen [8]. Der Vorgang der manuellen Instantiierung von Ontologien stellt sich als arbeitsintensiv und zeitraubend dar [2].

Zielsetzung In diesem Kapitel wird die Methodik der Anreicherung einer bestehenden Ontologie durch Instanzen, die aus den gewonnenen Daten der maschinellen Extraktionsvorgänge des HMM und des CBR stammen, beschrieben. Die umgesetzte Methodik wird sich an typischen Prinzipien der 'Ontologiepopulation' orientieren und den Vorgang der Instantiierung einer A-Box der Domänenontologie theoretisch durchführen.

Für diesen Zweck wird zunächst die graphische Beschreibung einer dem Projekt 'OnToBau' entstammenden Domänenontologie vorgenommen. Die Ontologie ist in der Ontologiebeschreibungssprache 'OWL' definiert, und beinhaltet aktuelle Entitäten des Bereiches Bauwesen die aus dem Projekt erstellt wurden. Aus der Domänenontologie wird ein beispielhafter Ausschnitt dieser Ontologie hervorgehoben, welcher in einer Darstellung aus ontologiespezifischen Konzepten, Attributen und Relationen

graphisch mit Protégé[1] aufbereitet wird. Ziel ist es, einen aktuellen Zustand und exemplarisch die vordefinierte Struktur der Ontologie in einem konkretisierten Abschnitt aufzuzeigen. Mit der Präsentation eines Teilabschnitts der Ontologie soll ein schematischer "Fixpunkt" geliefert werden, welcher verdeutlicht, an welcher Stelle die Methode der Ontologiepopulation ansetzen wird. Insbesondere wird in Definition einer 'T-Box' eine konkrete Beispielinstanz der Ontologie angegeben.

Im Weiteren soll ein kurzer Rückgriff auf die Ergebnisse des Extraktionsprozesses zur weiteren Verdeutlichung des angestrebten Vorganges erfolgen. Mit der Anschauung der aus dem maschinellen Lernen gewonnen Daten, das heißt den Ergebnissen in Form von extrahierten Produkten und deren zugehörige Produktangaben, soll das Vorhaben einer Anreicherung von Instanzen in der Ontologie durch eine geeignete Methodik, die auf diese Daten angewandt wird, aufgezeigt werden. Betrachtet wird damit ein Ausgangspunkt des Prozesses und die extrahierten Informationen liefern der Methode der Ontologiepopulation ihre Eingabe.

Ein wesentlicher Aspekt wird die theoretische Beschreibung geeigneter Methoden zur Ontologiepopulation sein. Eine kurze Einführung in die allgemeine Thematik der 'Ontologiepopulation' und ein grundsätzliches Prinzip der Wissenserweiterung konnte in Abschnitt 3.7 'Ontologielernen' gegeben werden. An dieser Stelle werden konkrete Entwurfsansätze, die unter Berücksichtigung der gegenwärtigen Problemstellung, also der vorliegenden Daten konzipiert werden, geliefert. Diese Methoden setzen den "Anreicherungprozess" des Wissens in der definierten Ontologie mit den extrahierten Daten aus den Lernverfahren um. Die Ansätze basieren auf klassischen Verfahren der Bereiche 'Named-Entity Recognition' und 'Ontologiepopulation'.

Zuletzt wird anhand einer exemplarischen Instantiierung der Daten in die Ontologie, eine der geeigneten Methoden angewandt. So wird eine Population der betreffenden Konzepte und Attribute in der Ontologie erfolgen. Das Ergebnis dieses Vorgangs wird abschließend zur Definition der exemplarischen 'A-Box', welche die extrahierten und wissenserweiternden Information in formalisierter Notation enthält, genutzt.

[1] http://protege.stanford.edu/

11.1 Die Domänenontologie

Um an dem Beispiel einer konkreten Ontologie Stellen an denen Instantiierungen durchgeführt werden können, aufzuzeigen, kann exemplarisch die Domänenontologie des Projektes 'OntoBau' graphisch ausgegeben werden. So kann eine Betrachtung des Aufbaus und der Struktur einer Ontologie vorgenommen werden. Insbesondere wird eine Anschauung einzelner Konzepte und Relationen, in denen diese in der Ontologie stehen ermöglicht. Die graphische Präsentation der Domäne durch die Ontologie, kann explizit Instanzen die aus der Domäne Bauwesen stammen, aufzeigen.
Die Ontologie des Projektes 'OntoBau' wurde in der Ontologiebeschreibungssprache 'OWL' (Unterabschnitt 2.2.3) erstellt, die graphische Ausgabe erfolgt in Protégé.
Die erste Abbildung 11.1 zeigt einen Ausschnitt der Ontologie zu Konzepten des Bereichs *Produkte*. Das Konzept 'Produkt' stellt taxonomische Relationen zu konkreteren Produktentitäten, in Form von speziellen Produktbereichen bereit. Diese Bereiche können auf einem niedrigeren Abstraktionslevel durch Relationen zu einzelnen, Produkten die dem Konzept untergeordnet sind, konkretisiert werden.

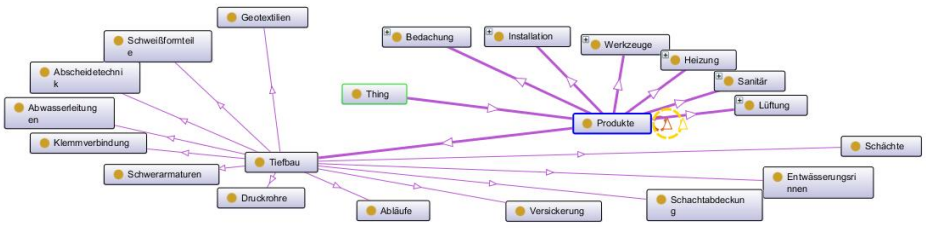


Abbildung 11.1: Ausschnitt der Ontologie 'OntoBau' mit Produktkonzepten

Das beschriebene Konzept *Produkt* kann in einer zweiten Darstellung in einer übersichtlicheren Anordnung (Abbildung 11.2) ausgegeben werden.

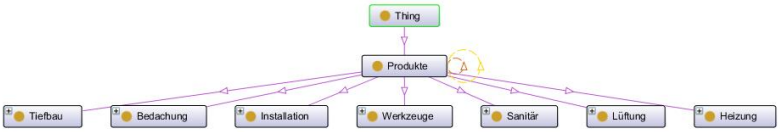


Abbildung 11.2: Weitere Anordnung des Konzepts 'Produkt'

Auf Abbildung 11.3 ist eine Instantiierung mehrerer Produktkonzepte abgebildet. Die Produkte werden jeweils mit einem expliziten Produktnamen instantiiert.

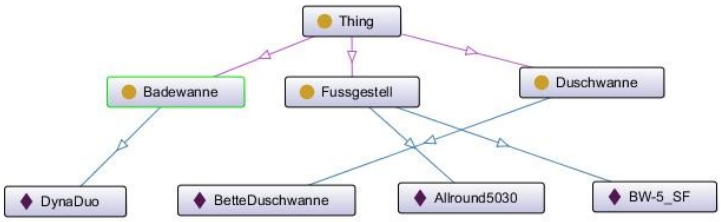


Abbildung 11.3: Instantiierung von Produktkonzepten

T-Box Um ein Beispiel eines konzeptionellen T-Box-Statements (Unterabschnitt 2.2.4) zu liefern, kann eine einfache Klasse in *OWL* angegeben werden. Die Klasse beschreibt auf einfachste Weise ein Konzept der Ontologie, welches mit Instanzen aus dem Ergebnisstring befüllt werden kann. Die erstellte exemplarische Klasse zeigt den angedeuteten Ansatz [33] einer Konzeptdefinition.

T-Box-Statement (OWL)

```
<owl:Class rdf:ID="Produkt"/>

<owl:Class rdf:ID="Waschtisch">
    <rdf:subclassOf rdf:resource="#Produkt"/>
</owl:Class>

...

<owl:DatatypeProperty rdf:ID="Abmessungen">
    <rdfs:domain rdf:resource="#Waschtisch"/>
    <rdfs:range rdf:resource="#&xsd;integer"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Farbe">
    <rdfs:domain rdf:resource="#Waschtisch"/>
    <rdfs:range rdf:resource="#xsd;string"/>
</owl:DatatypeProperty>
...
```

11.2 Ergebnisse des Extraktionsvorgangs

Die extrahierten Informationen beider Lernverfahren liegen als Ausgabefile vor. Der Ausgabefile ist eine einfache Textdatei, die mit allen gefundenen Daten eines Dokuments befüllt wurde. Jeder dieser Files besteht aus einer Ansammlung von zeilenweise angeordneten *'Ergebnisstrings'*. Die Strings beinhalten jeweils einen extrahierten Produktnamen und die für das zugeordnete Produkt typischen Produktinformationen. Diese Daten liegen in annähernd strukturierter Form vor. Abbildung 11.4 zeigt eine typische Ausgabedatei des CBR-Verfahrens. Die Zeile eines Ergebnisstrings lie-

```
Aus Extraktion gefundene Produkte:
  Waschtisch= Stck Alape  Waschtisch  rechteckig  80x49cm  weiss  m.  mit  Auszug  liefern
  Eckventil = Satz Eckventil  M10x140mm  liefern
  Kristallglas = Stck Hansgrohe  Kristallglas  Chrom  mit  Ablaufgarnitur  liefern
  Geruchverschluss = Stck Röhren  Geruchverschluss  1  1/4Mx1  1/4\  Messing  chrom  liefern

Mögliche zusätzlich Produkte:
  Waschtischbefestigung = Satz Waschtischbefestigung  M10x140mm  liefern
  Spiegelset = Stck Mirror  tec  Profi  KlicK/08  Spiegelset  bis  0.8  qm  für  Feuchträume  liefern
  Deckenleuchte = Stck Wand  und  Deckenleuchte  satiniert  liefern
```

Abbildung 11.4: Ausgabefile mit Ergebnisstrings

fert an erster Stelle das eigentliche Produkt mit seinem Produktnamen, die folgenden Stellen werden durch zusätzliche produktspezifische Angaben dieses Produktes belegt. Für die Befüllung der Ontologie bedeutet dies, dass an führender Position ein Konzept vorzufinden ist, welches von den nachfolgenden Angaben, damit also den Attributen des Konzepts, beschrieben wird. Die extrahierten Entitäten aus einem String entsprechen folglich einer möglichen Instanz der Ontologie.

11.3 Methodologie: Ontologiepopulation

Die Prozeduren des maschinellen Lernens, in Umsetzung der Verfahren des 'Hidden-Markov-Modells' (Kapitel 6) und des 'Case-Based Reasoning'(Kapitel 7), wurden in dieser Arbeit zur Erkennung des Wissens (engl.: knowledge discovery) aus den domänenspezifischen Dokumenten in Form von repräsentativen Dokumenten der Klasse 'Rechnung' eingesetzt. Um den Schritt der Wissensanreicherung (engl.: knowledge enhancement) bzw. der Anreicherung des Wissens (engl.: knowledge enrichment) in Anwendung der 'Ontologiepopulation', durch neue Instanzen der bestehenden Konzepte durchzuführen, wird eine "maßgeschneiderte" Methodik verlangt.

Prinzipiell können beim sogenannten 'Enhancing' einer Ontologie unterschiedlichste Ansätze, basierend auf Methoden der Informationsextraktion, schon direkt zur Population einer Ontologie genutzt werden. Eine Vielzahl der Ansätze nutzt das System zur Informationsextraktion, um Konzeptinstanzen in einem annotierten Korpus zu lokalisieren. Typische wissenserweiternde Techniken arbeiten auf lexikalischer Ebene mit Prinzipien, welche die verschiedensten Lexikalisierungen[2] eines Begriffs ausnutzen, um spezielles Wissen anzureichern. Zusätzlich können diese Ansätze unterschiedlichste Ressourcen, wie Wörterbücher oder bestehende Ontologien verwenden, die den Anreicherungsprozess unterstützen.[44]
Ein Ansatz auf Basis einer klassischen, syntaktisch linguistischen Verarbeitung ist in Anbetracht der Art von Informationen aus der Domäne weniger sinnvoll anwendbar, da es sich bei den Begrifflichkeiten der Menge von Entitäten aus dem Bereich 'Bauwesen' hauptsächlich um Eigennamen, speziell also Produkte für die sich kaum weitere Lexikalisierungen ergeben, handelt.

Konkrete Problemstellung Liegen die Daten zur Anreicherung des Wissens, dieses Wissen ist annähernd gleich zu setzten mit Instanzen der Konzepte einer Ontologie, bereits aus einem ersten Extraktionsvorgangs vor, so kann eine geeignetere Methodik gefunden werden. Der Ergebnis-File der Extraktion besteht wie in den Ergebnissen der maschinellen Lernverfahren beschrieben, aus einer Ansammlung von "Ergebnisstrings". Jede Zeile dieser Strings enthält damit die extrahierten Informationen aus dem jeweiligen Dokument für einen Kontext eines Produktes, in der Reihenfolge Produkt in führender Position und untergeordnete Produktinformationen nachfolgend. Ein Beispiel kann dies verdeutlichen:

Beispiel 'Ergebnisstring'

- **allgemeine Struktur**: erste Position *Produkt*, nachfolgende Positionen *Produktinformationen* (Abmessungen, Farbe, Zubehör etc.)

- **konkrete Instanz**: "Alape Waschtisch 40x120 weiss verchromt"

[2] auch lexikalische Synonyme

Ziel wird also sein, eine Methode zu finden, welche die nahezu optimalen Voraussetzungen des strukturierten Ergebnisstrings ausnutzt, um ein möglichst unüberwachtes Verfahren entwerfen zu können. Ein Verfahren kann demnach ohne den Aufwand einer erneuten Annotation der Extraktionsergebnisse angewandt werden, da kein weiteres (maschinelles) Lernverfahren zum Lernen, von z. B. unstrukturierten natürlichsprachlichen Eingaben, verlangt wird. Nach Kenntnis der Probleminstanz in Form des Ergebnisstrings, wird ein klassisches 'Preprocessing' auf Basis von syntaktischen Techniken des 'NLP' nicht angewandt.

11.3.1 Ansatz: Named Entity Classification

Um eine erste prinzipielle Idee für eine geeignete Methodik liefern zu können und die Verarbeitung eines "Ergebnisstrings" aufzugreifen, eignet sich zunächst die Betrachtung eines Ansatzes aus dem Bereich des 'Named Entity Recognition' (NER). Das Named Entity Recognition ist eine Anwendung der Informationsextraktion und verfolgt das Ziel, relevante Bestandteile eines Textdokumentes zu lokalisieren um diese in vordefinierte Kategorien zu klassifizieren. Die Bestandteile werden auch als 'Entitäten' bezeichnet und sind spezifische Ausprägungen eines Konzeptes einer bestimmten Anwendung. Die Entitäten werden Kategorien zugewiesen und sofern diese ebenfalls eindeutig, im Sinne einer Konkretisierung zum Beispiel eines Eigennamens identifizierbar sind, entspricht diese einer 'Named Entity'.
Die Methodik des NER ist dem Vorgang der Klassifikation von Instanzen in Konzepten einer Ontologie ähnlich. Collins und Singer geben in ihrer Arbeit 'Unsupervised Models for Named Entity Classification' [10] einen Ansatz für unüberwachte Modelle zur *Klassifikation* von 'Named Entities'.

Unüberwachte Named Entity Classification Die Methodik eines unüberwachten Verfahrens bietet sich in Anbetracht der vorliegenden Daten der Extraktion an, so kann ein manueller Labelvorgang aus Gründen des hohen Aufwands möglichst vermieden werden. Die extrahierten Daten werden bestenfalls ohne manuellen Eingriff in eine bestehende Ontologie eingebaut.
In Folgendem Abschnitt wird Bezug auf den Ansatz der 'unüberwachten Modelle zur

Klassifikation' von Collins [10] genommen.
Um den Aufwand des Erstellens einer großen Anzahl von Regeln zur Klassifikation, welche die Domäne abdecken, und das manuelle "Labeln" großer Mengen von Trainingsbeispielen um einen Klassifikator (engl.: classifier) zu trainieren, minimieren zu können, wird ein Ansatz geliefert, der mit "unlabeled" Daten umgehen kann. Grundlegende Annahme ist, dass zum einen die Schreibweise (engl.: spelling) einer Bezeichnung und zum anderen der Kontext, in dem dieser auftritt, hinreichende Kriterien liefert, um die Kategorie einer "named-entity" Instanz zu bestimmen. So kann aus ungelabelten Beispielen zur Klassifikation eine Funktion gelernt werden, die aus einem Eingabestring (Eigenname) seinen Typen dementsprechend eine Kategorie ableitet. Die Kategorien könnten zum Beispiel *Produktname*, *Maße*, *Farbe* sein. Ein gutes Klassifikationssystem identifiziert "Waschtisch" als Produktname, "40x120" als Maß und "weiss" als Farbe. Zur Klassifikation benutzt der Ansatz Regeln, die einen Eingabestring untersuchen nach:

- der Schreibweise;
- dem Kontext, in dem dieser steht.

Regeln zur Untersuchung der Schreibweise können z. B. aus einem einfachen "Look-up" bestehen, welcher typische Eigennamen in einer Liste vorliegen hat, oder einer Regel entsprechen, die innerhalb des Strings nach Worten sucht, welche die betrachteten Eingabe eindeutig identifizieren. Ein einfachstes Beispiel zu dieser Regel ist die Voranstellung eines Titels oder einer Anrede, welche einen darauffolgenden String als *Person* bestimmt. Anpassungen der Regeln auf die Domäne erfolgen in der Modifikation der Anwendung (Unterabschnitt 11.3.1).
Eine kontextuelle Regel betrachtet die Worte, die den betrachteten String umgeben und können aus dem Kontext eines vorangehenden Wortes auf die Kategorie schließen.
Mit einer geeigneten Kombination von wenigen Regeln lässt sich ein akkurater Klassifikator für ungelabelte Daten entwerfen, der bis auf den Entwurf der Regeln ohne "Überwachung" (engl.: supervision) auskommt. Die Methode basiert auf Redundanz innerhalb der ungelabelten Daten. Sogenannte Features zu Schreibweise und Kontext

bestimmen den Typ eines Strings, um diesen in eine Kategorie zuweisen zu können. Die Modelle werden auf Texte, aber auch auf einzelne Eingabestrings angewandt. Sofern eine Klassifikation auf (natürlichsprachlichen) Text erfolgt, muss in einem vorangehenden Schritt eine Extraktion nach linguistisch grammatikalischen Kriterien auf Basis von Nominalphrasen erfolgen, um Wortsequenzen mit Eigennamen in Named-Entity Beispiele zu überführen. In einer zweiten Phase wird eine Extraktion von Features vorgenommen. Eigens definierte Features ermöglichen eine detailliertere Untersuchung nach den Schreibweise- und Kontextregeln, so kann einer Entität ihre Features mitangegeben werden. Das Auffinden der *(Schreibweise, Kontext) Paaren* in den "geparsten" Daten ermöglicht also die Extraktion einer Anzahl von Features. Diese werden folglich benutzt, um jeweils Beispiele für den Lernalgorithmus repräsentieren zu können. In [10] werden zwei Ansätze unüberwachter Lernalgorithmen geliefert, die auf den hier vorgestellten Prinzipien arbeiten. Der vorgestellte Entwurf zur Population durch Instanzen in diesem Kapitel wird einen ebenfalls überwachten Ansatz anwenden.

Anwendungsspezifische Modifikationen Die Eignung für die konkrete Anwendung auf den Ausgabefile kann durch eine einfache Erkenntnis begründet werden. Der Klassifier für Entitäten kann sowohl in einer Zeile als auch für den gesamten Ergebnisstring, also in Spalte der führenden Konzepte, dies sind die Produkte, auf einen Kontext, untersucht werden. Die extrahierten Produkte eines gesamten Dokuments stehen in der Regel ebenfalls in einem gemeinsamen Kontext, da ein Rechnungsdokument oft eine Zusammenstellung von Produkten eines gemeinsamen Bereichs ist. Des Weiteren werden spezielle Regeln für die Untersuchung des Ergebnisstrings definiert, die es ermöglichen typische Schreibweisen der Produktangaben zu erkennen. In der Domäne Bauwesen können dies Muster sein, welche beispielsweise 'Buchstaben-Zahlen Kombinationen', oder Maßangaben aufgrund der Repräsentation "Zahl x Zahl" erkennen.

Die vorgestellte *unsupervised* Methodik klassifiziert auf einfache Weise die Daten des Ergebnisstrings. Den zuvor unbekannten Entitäten konnte durch Klassifizierung ein Konzept zugewiesen werden. Eine geeignete Methode zur Population der Onto-

logie mit "klassifizierten" Instanzen, die aus dem Ausgabestring stammen, wird im Folgenden beschrieben. Der gelieferte Ansatz befüllt die bestehende Ontologie mit den vorliegenden Instanzen und baut diese in die vordefinierten Konzepte an die entsprechende Stelle ein.

11.3.2 Problemdefinition Ontologiepopulation

Die Problemstellung der Population einer Ontologie kann durch eine einfache Formalisierung nach '*Celjuska* und *Vargas-Vera* [8]' beschrieben werden.
Die Population einer gegebene Ontologie O wird durch Instanzen, die möglichst automatisiert aus Text extrahiert werden, erfolgen. Die Aufgabe eines Systems zur Population ist zunächst die Identifikation von wichtigen Entitäten in einem Dokument. Die Entitäten können als 'Slot Values' $v_1, v_2, ...; v_{Ni}$ bezeichnet werden und sind als unbefüllte Stellen in einer Klasse aufzufassen. Slot Werte können jeweils einer bestimmten Klasse C_i zugeordnet werden. Es soll die Konstruktion einer Instanz $I_i = (v_i, ..., v_{Ni})$ für eine Klasse $C_i \in \{C_1, C_2, ..., C_M\}$ in der gegebenen Ontologie O durchgeführt werden.
Ein geeignetes Verfahren bestimmt, ob die konstruierte Instanz I_i gültig ist, und ob diese in eine Klasse C_i eingefügt werden kann. Die Ontologie O besteht sinnvollerweise aus einer Anzahl von Klassen $C_1, ..., C_M$, wobei jede Klasse C_i mit Slots $s_i, s_2, ..., s_{Ni}$ die mit $v_i, v_2, ..., v_{Ni}$ instantiiert werden konnen, definiert wird.
Die Slots sind demnach als die Attribute einer Klasse anzusehen und geben detailliertere Informationen über die Instanz, die konstruiert werden soll. Ein System hat die Aufgabe, diese Attribute $v_i, ..., v_{Ni}$ in einem Dokument oder im konkreten Fall in dem Ergebnissstring zu identifizieren um eine Instanz zu erstellen, die eine entsprechende Klasse C_i in der Ontologie O befüllt.

11.3.3 Konzeptionelles System zur Ontologiepopulation

In dem hier überlieferten konzeptionellen Desing zur Population einer Ontologie durch Instanzen wird ein grundsätzlicher Ansatz aufgegriffen. Die theoretische Umsetzung des System orientiert sich an einem Ansatz, welcher "Kontextfeatures" (Unterabschnitt 11.3.4) verwendet. Die Features entstammen aus einem Korpus und

definieren einen Kontext, welcher mit einer semantischen Klasse in Verbindung gebracht wird [44]. Das System ist ausschließlich für die effiziente Befüllung der Klassen bzw. Konzepte verantwortlich, es bedarf keiner weiteren Lernverfahren oder syntaktischen Modelle um Entitäten aus dem Korpus zu identifizieren. Die Entitäten liegen bereits im Ergebnisstring in strukturierter, und nach Anwendung des allgemeinen NEC-Prinzips, in klassifizierter Form vor.

Der Prozess der 'Named Entity Classification' (Unterabschnitt 11.3.1) wird als ein "pre-processing" ähnlicher Vorgang, eingesetzt, so kann der Ergebnisstring einer Vorverarbeitung unterzogen werden. Die aus der Extraktion vorliegenden Informationen werden mit einer ersten unüberwachten Methodik semantisch klassifiziert. Daher wird eine aufwendige, da manuell annotiert, wie sie bei einer Vielzahl von semiautomatischen Systemen [8] verlangt wird, um das Lernen durch Lernverfahren einzuleiten, entfallen.

11.3.4 Umsetzung der Befüllung: 'Class-Word Ansatz'

Das 'Class-Word' Prinzip nach *Cimiano* und *Völker* [9] ist, wie gezeigt werden soll, ein sinnvoll anwendbarer Ansatz. Der "isolierte" und schon klassifizierte Ergebnisstring liefert nahezu optimale Voraussetzungen für das unüberwachte Verfahren, da dieser in direkter, für die Methodik verwertbarer Form vorliegt. Zunächst kann eine weitere allgemeine Formalisierung des Begriffs "Population" vorgenommen werden, um die Beschreibung der hier gelieferte Anwendung vorzubereiten. Entitäten werden im Zusammenhang dieses Ansatzes als Terme angesehen.

Szenario Ontologiepopulation Gegeben ist eine Menge von Termen $T = t_1, t_2, ..., t_n$ und eine Sammlung von Dokumenten D. Terme aus T sind in den Dokumenten enthalten. Eine Menge von vordefinieren Klassen $C = c_1, c_2, ..., c_m$ weisen Konzepte einer Klasse aus. Jeder Term t_i muss einer geeigneten Klasse in C zugewiesen werden.[40]
Eine zusätzliche Annahme der überlieferten Formalisierung aus [40] ist:

- Klassen aus C sind korrelativ[3] disjunkt
- Jeder Term wird genau einer Klasse zugewiesen.

Der unüberwachte Ansatz zur Population einer Ontologie basiert auf Ähnlichkeit sogenannter *'Vektor-Features'* zwischen jenen Konzepten c und einem Term t der klassifiziert wird. So kann ermittelt werden, wie sehr ein konkretes Produkt eine entsprechende Instanz einer Produktklasse ist. Die Methode findet dabei die 'Feature-Vektor Ähnlichkeit' zwischen dem vorliegenden Produkt und der in der Ontologie definierten Produktklasse. Jede Instanz der Testmenge T wird einer der Klassen aus der Menge C zugeteilt.

Kontextvektoren Im Gegensatz zum eigentlichen Verfahren Class-Word, bei dem die Features aus einem Korpus gesammelt werden, kann bei Anwendung des Ansatzes auf den Ergebnisstring, die Menge der Features, die im Kontext für die direkte Erstellung des Vektors aus den Features stehen, verwendet werden. Der aus den Features erstellte Vektor ermöglicht die Klassifizierung in eine geeignete Klasse. Die Vektoren werden auch als Kontextvektoren v_t, v_c bezeichnet, da sie die Kontexte der Terme und Konzepte abbilden. So sind die einzelnen Elemente gewichtete Kontextfeatures aus dem Korpus des Terms t (z. B. "Waschtisch") oder des Konzeptwortes c (z. B. "Produkt"). Der Korpus der Terme ist folglich der jeweilige Ergebnisstring. Der Korpus des Konzeptwortes ist das Konzept der Domänenontologie.

[3] voneinander, wechselseitig

Algorithmus zur Klassifikation

Unter Verwendung der Kontextvektoren kann ein unüberwachter Algorithmus zur Klassifikation angewandt werden, welcher die verschiedenen Terme in die entsprechenden Konzepte "*mappt*".

Unsupervised Algorithmus zur Ontologiepopulation [40]

```
1 classify (T, C, Corpus)
2       foreach (t in T) do{
3             v_t = getContextVector(t, Corpus);}
4       foreach (c in C) do{
5             v_c = getContextVector(c, Corpus);}
6       foreach (t in T) do{
7             classes[t] = argmax_{c∈C} sim(v_t, v_c);}
8       return classes[];
9 end classify
```

Die gelieferte Prozedur klassifiziert einen Term anhand seiner Features. Features können z. B. gemäß der Relevanz und Wichtigkeit in ihrer Klasse gewichtet werden. Da mit den Angaben des Ergebnisstrings ein Kontext eines Produktes abgebildet wird, können die Elemente des Vektors auch als *semantische Features* angesehen werden. Die Ähnlichkeit der Kontextvektoren wird bestimmt, indem eine maximierende Ähnlichkeitsfunktion auf die gewichteten Vektoren angewandt wird. In der Anwendung auf einen Ergebnisstring sind die Produkte mit ihren erweiternden Produktbeschreibungen wie Maße, Farbe und Zubehör, also die Features des Vektors. Ein Vektoren wird dementsprechend aus NEC-klassifizierten Entitäten des Ergebnisstrings aufgebaut und mit Hilfe des Algorithmus den Klassen gemäß ihrer Ähnlichkeit der Features zugeordnet. Die Ontologie wird unter der Annahme, dass Produkte den Konzepten und Produktangaben den Attributen zugewiesen werden, mit Instanzen befüllt. Mit der Instantiierung werden die konkreten Ergebnisse des Extraktionsvorgangs aus HMM und CBR in die Ontologie eingebaut.

11.4 Instantiierung einer A-Box

Eine Befüllung durch Ergebnisse der Extraktion, die in Repräsentation des Kontextvektors vorliegen, erfolgt, indem die entsprechende Klasse der bestehenden Ontologie durch den 'Class-Word Ansatz' aufgefunden wird. Liegt eine entsprechende Klasse vor, so wird die Instanz des Ergebnisstrings an die passende Stelle des Konzeptes eingefügt. Ein vordefiniertes T-Box-Statement kann demnach mit konkreten Instanzen belegt werden.
Folglich wird jeder klassifizierte Eintrag des Ergebnisstrings als eigenes Individuum 'ID', d. h. Instanz einer Klasse in OWL, angesehen. Nach Auffinden des geeigneten Konzepts, in OWL der "Klasse", werden diese "IDs" an die betreffende Stelle des Datenwertes (DatatypeProperty) gefügt. Ein konzeptionelles T-Box-Statement wird mit einem konkreten Wert belegt. Eine Instanz in Ausprägung eines A-Box-Statements wurde in das Konzept eingefügt.

A-Box-Statement (OWL)

```
1
2       <owl:DatatypeProperty rdf:ID="Abmessungen">
3           <rdfs:domain  rdf:resource="#Waschtisch"/>
4           <rdfs:range  rdf:resource="#&xsd;integer"/>
5       </owl:DatatypeProperty>
6       <owl:DatatypeProperty rdf:ID="Farbe">
7           <rdfs:domain  rdf:resource="#Waschtisch"/>
8           <rdfs:range  rdf:resource="#xsd;string"/>
9       </owl:DatatypeProperty>
10      ...
11
12      <Abmessungen rdf:ID="120"/>
13      <Farbe rdf:ID="weiss"/>
```

KAPITEL

ZWÖLF

Fazit und Vergleich der Systeme

In dieser Ausarbeitung wurde die Einsatzfähigkeit maschineller Lernverfahren zur Informationsextraktion aus repräsentativen Textdokumenten einer bestimmten Domäne getestet. Ein domänenspezifischer Textkorpus, welcher einer Menge eingescannter und vorverarbeiteter Rechnungen entspricht, wurde diesbezüglich auf typische Eigenschaften und Regelmäßigkeiten des Dokumentes untersucht. Durch die Erkenntnisse aus dieser Analyse, die im Kontext der jeweilig eingesetzten Verfahren bewertet wurden, konnten geeignete interne Repräsentationen der Probleminstanz 'Rechnungsdokument' erarbeitet werden. Eine Anpassung bestehender Java-Frameworks, welche die beiden im unterschiedlichen Ansatz mit eigenen Verfahren umsetzen, konnte später durchgeführt werden.

Mit Hilfe des zuvor erworbenen theoretischen Wissens der jeweiligen Verfahren, sowie speziellen Ideen bzw. weiteren Erkenntnissen aus schon existierenden spezifischen Anwendungen, konnten Entwürfe von Systemen erstellt werden, welche die automatische Informationsextraktion aus Textdokumenten theoretisch umsetzen. Anhand dieser prinzipiell unterschiedlichen Entwürfe, wurden zwei voneinander unabhängige Systemausführungen implementiert. Diese setzen zum einen die Informationsextraktion mittels 'Hidden-Markov-Modell' um, zum anderen konnten relevante Informationen durch eine Case-Based Reasoning Verfahren extrahiert werden.

Die beiden Entwürfe wurden mittels zwei 'Opensource Java-Frameworks', die bereitgestellt worden sind, umgesetzt. Ein kurzes Fazit, im Sinne eines abschließenden

Vergleichs der beiden Umsetzungen kann erfolgen. Die Abschlussbetrachtung wird sich sowohl aus den Daten der Evaluationsergebnisse der beiden Implementierungen, als auch aus den Erkenntnisse im Umgang bzw. der Umsetzung der Systeme, zusammensetzen. Insbesondere werden erkannte Probleme und Schwierigkeiten, sowie besondere Eignungsfähigkeiten der Systeme erwähnt.

12.1 Vergleich der Systeme

Die Genauigkeits- und Vollständigkeitsmaße können im Allgemeinen verwendet werden, um Systeme, die einen identischen '*Task*' durchführen, miteinander zu vergleichen. Eine eindeutige Aussage über eine bessere Eignung eines der beiden Systeme aufgrund besserer Evaluierungsergebnisse, kann allerdings nur dann erfolgen, wenn sowohl 'Precisionwert' als auch 'Recallwert' eines Systems besser sind, als die Werte des anderen Systems. In allgemeinen Testauswertungen ist eine gewisse Gegenläufigkeit der Maße abzulesen. Die Werte der Ergebnisse liegen zwischen beiden Extremwerten. So können die Resultate, je nach Ergebnis des jeweiligen Maßes, zwar Auskunft über unterschiedliche Eignungen der Systeme geben, nicht jedoch zu einer eindeutigen Erkenntnis bezüglich einer besseren Eignung für eine spezielle Anwendung. Zu einer allgemeinen Abschätzung der besseren Eignung in Bezug auf eine Anwendung, müssen beide Werte zu 'Precision' und 'Recall' über den Werten des anderen Systems liegen.[14]

Evaluierungsdaten In den Evaluierungen der Systeme konnte nachgewiesen werden, dass die Umsetzungen zur automatischen Informationsextraktion mittels Hidden-Markov-Modell und Case-Based Reasoning mit den hier gelieferten Systemen prinzipiell geeignet ist. Die Ergebnisse der "großen" Testreihen mit jeweils 25 Testdokumenten zeigten, dass beim HMM-System im Durchschnitt bei optimaler Konfiguration eine Genauigkeit von 78% und eine Vollständigkeit von bis zu 99% im besten Fall erreicht werden kann. Beim CBR-System konnte durchschnittlich eine Genauigkeit von 55% und Vollständigkeit von 46% ermittelt werden.
Eine Eignung bezüglich der gegebenen Problemstellung fällt damit eindeutig zu Gunsten des hier umgesetzten HMM-Systems aus, welches bei beiden Gütemaßen höhere

Prozentzahlen erreichen konnte. In erster Linie ist dies auf das komplettere Framework zurück zu führen, welches im Falle des HMM exakt auf die Problematik der Extraktion aus Rechnungsdokumenten angepasst werden konnte.

Insgesamt konnte festgestellt werden, dass beide Verfahren gegenläufige Ergebnisse in Bezug auf die Retrievalmaße liefern. So ist bei Betrachtung der Auswertungen zu erkennen, dass die 'CBR-Umsetzung', sofern eine Extraktion erfolgt ist, in der Regel hohe Prozentwerte (ab 75% bei einer großen Anzahl von Dokumenten) aufweisen konnte. Das bedeutet, die Informationen konnten, falls sie erkannt wurden, mit einer hohen Genauigkeit identifiziert werden. Der 'Recallwert' lag im Fall der Extraktion von Informationen regelmäßig unter dem der Genauigkeit. Die HMM-Implementierung dagegen konnte nahezu konstante Prozentwerte in der Vollständigkeit des Extraktionsvorgangs eines Dokuments erreichen. Die Werte zum Recall lagen generell bei über 90% und nahe dem Extremwert. Die Genauigkeit verbesserte sich mit Anzahl der getesteten Dokumente, was auf die stochastische Methodik des HMM zurück zu führen ist. Mit steigender Anzahl von Testdokumenten verbessert sich also die Lernfähigkeit des Modells und damit die Genauigkeit der Extraktion.

Für beide Ansätze konnte die Einsatzfähigkeit auf die Problemstellung nachgewiesen werden. Für das Hidden-Markov Modell konnte außerdem die Anpassungsfähigkeit auf neue Probleminstanzen, mit dem zusätzlichen Test auf Produktinformationen aus einem Katalog, gezeigt werden.

Das Case-Based-Reasoning-System, in der hier gelieferten Umsetzung, zeigte Robustheit des Programms, auf unaufbereitete nicht optimale Eingaben. Dies belegte der erbrachte Test auf 'Kriterium 3' in der Evaluierung zum CBR. Erstaunlicherweise konnte eine Eingabe auf komplett unaufbereitete Dokumente Ergebnisse liefern, die im Bereich des Test unter optimalen Bedingungen lagen.
Die Wiederverwendbarkeit des HMM auf neue Dokumenttypen ermöglicht, dass keine Umstrukturierung der Implementierung verlangt wird. Das CBR-System ist nicht für andere Dokumenttypen wie z. B. Kataloge konzipiert und erfordert eine Umstrukturierung bzw. Anpassung der Implementierung.

Insgesamt kann dem HMM in dieser Form eine bessere Eignung zur textuellen Verarbeitung von Dokumenten attestiert werden. Die komplizierte Angleichung der Retrievalmethoden aus dem Konzept und dem bestehenden Framework des CBR erschwerten die Umsetzung des CBR-Systems. Um einen besseren Ablauf des Prozesses zu garantieren, mussten Vereinfachungen wie das Ausnutzen der Muster im Dokument eingebaut werden.

Die Grundsysteme Ein großer Vorteil des HMM-Systems sind die Optimierungseinstellungen des Verfahrens, die durch die frameworkeigenen Konfigurationen und anwendbaren Programmfeatures des sehr komplexen Systems bereitgestellt werden. Hingegen ergab sich bei der CBR-Umsetzung die Schwierigkeit, ein weniger komplettes Systems vorzufinden, welches im Vergleich zum HMM ausschließlich Basismethoden und ein Grundgerüst eines CBR-Systems enthalten hat. Dabei stellte sich die eigentliche Umsetzung und Anpassung des konzeptionell erstellten Verfahrens auf das gegebene Framework als schwieriger heraus. Infolgedessen ergab sich ein größerer Anteil an eigener Designentscheidungen, da das System eher einfach und auf andere Problemstellung ausgelegt wurde.

Dem HMM-System konnte mit dem Test auf Informationen aus Produktkatalogen die Übertragbarkeit auf neue Problemstellungen nachgewiesen werden. Damit das System für neue Problematiken eingesetzt werden kann, ist, wie in Unterabschnitt 6.3.1 beschrieben, eine Vorverarbeitung des Dokumentenkorpus nötig. Hierfür mussten sowohl die Trainingsdokumente bearbeitet werden, als auch die eigentlichen Testdokumente, aus denen die Informationen extrahiert werden sollten.
Das System kann unabhängig vom Dokumentenkorpus eingesetzt werden, sofern die Dokumente problementsprechend vorverarbeitet werden. In der hier gelieferten Umsetzung erfolgte die Vorverarbeitung der Dokumente manuell. Zur Vorverarbeitung gehörte unter anderem die Analyse der Dokumente, um bestimmte Regelmäßigkeiten und Eigenschaften des Dokuments auszunutzen. Diese Eigenschaften wurden dazu verwendet, die Trainingsdaten manuell, mit den dazugehörigen 'target Fields' zu annotieren. Eine Modifikationsmöglichkeit wäre, die "target Fields" zukünftig automatisch annotieren zu lassen. Dies könnte umgesetzt werden, sofern bekannt ist, welche

Wörter explizit auf ein "target Field" hindeuten. So könnte der Aufwand der Vorverarbeitung entfallen. Im Gegensatz zum aktuellen System extrahiert das System hierbei nicht nur die Informationen zu den Produkten, sondern zeigt dem Benutzer an, welchem Konzept die jeweils extrahierten Informationen zuzuordnen sind. Diese Vorgehensweise könnte die anschließende Population der Ontologie ebenfalls vereinfachen, da dieses Prinzip Konzepte der Ontologie anzeigen kann.

Eine Erweiterung zur Modifikation des CBR-Systems wurde bereits in Abschnitt 10.4 geliefert. Diese basierte auf der Idee einer Ausnahmebehandlung von Dokumenten, die vollständig aus unbekannten Produkten bestehen und aus denen keine geeignete Repräsentationen eines Falles erstellt werden konnte. Des Weiteren wurde eine dynamische Erstellung der initialen Fallbasis aus stetig erweiterten Matchinglisten, die mit einem möglichst umfangreichen, geschloßenen Vokabular aus aktuellsten Begriffen der Domäne versorgt wird, vorgeschlagen. Diese aktuellen Daten können aus einer Datenbank, die regelmäßige "Updates" erfährt, oder direkt aus der Domänenontologie stammen und damit das System mit den neuesten, noch unbekannten Informationen versorgen.

ABBILDUNGSVERZEICHNIS

LITERATURVERZEICHNIS

[1] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, March 1994.

[2] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel Shadbolt. Web based Knowledge Extraction and Consolidation for Automatic Ontology Instantiation. In *2 nd Int. Conf. Knowledge Capture (KCap'03), Workshop on Knowledge Markup and Semantic Annotation*, 2003.

[3] James F. Allen. Lectures. Website, zuletzt abgerufen am 25.05.2011. `http://www.cs.rochester.edu/u/james/CSC248/Lec4.pdf`;.

[4] Altova. Was ist das Semantic Web ? Website, zuletzt abgerufen am 25.05.2011. `http://www.altova.com/de/semantic_web.html`;.

[5] Ralph Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications*, volume 2432 of *Lecture Notes in Computer Science*. Springer, 2002.

[6] Ralph Bergmann, Janet L. Kolodner, and Enric Plaza. Representation in case-based reasoning. *Knowledge Eng. Review*, 20(3):209–213, 2005.

[7] Horst Bunke and Terry Caelli. *Hidden Markov Models: Applications in Computer Vision*. World Scientific Publishing Co. Pte. Ltd., P O Box 128, Farrer Road, Singapore 912805, 2001.

[8] David Celjuska and Dr. Maria Vargas-Vera. Ontosophie: A Semi-Automatic System for Ontology Population from Text. In *International Conference on Natural Language Processing (ICON).*, 2004.

[9] Philipp Cimiano and Johanna Voelker. Towards large-scale, open-domain and ontology-based named entity classification. Website, zuletzt abgerufen am 25.05.2011.
`http://people.aifb.kit.edu/pci/Publications/ranlp05.pdf`;.

[10] Michael Collins and Yoram Singer. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

[11] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[12] R. Lopez de Mantaras et al. Retrieval, reuse,revision, and retention in case-based reasoning. *The Knowledge Engineering Review*, 00:0(1-2):39–59, 2005.

[13] Karsten Draba. Anwendung von Ontologien. Website, zuletzt abgerufen am 25.05.2011.
`http://www.dbis.informatik.hu-berlin.de/dbisold/lehre/WS0203/SemWeb/artikel/11/Draba_Anwendung-von-Ontologien-Text2.pdf`.

[14] Reginald Ferber. Information Retrieval - Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web. 2003.

[15] Gernot A. Fink. *Mustererkennung mit Markov-Modellen.* B.G. Teubner Verlag, 2003.

[16] Dayne Freitag and Andrew Kachites Mccallum. Information Extraction with HMMs and Shrinkage. In *In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.

[17] Dayne Freitag and Andrew Kachites Mccallum. Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the*

Seventeenth National Conference on Artificial Intelligence, pages 584–589. AAAI Press, 2000.

[18] Norbert Fuhr. Wissensrepräsentation für Texte. Website, zuletzt abgerufen am 25.05.2011.
`http://www.is.informatik.uni-duisburg.de/courses/ir_ss10/folien/irfk4-print.pdf`.

[19] Hatem Hamza, Yolande Belaïd, and Abdel Belaïd. Case-Based Reasoning for Invoice Analysis and Recognition. In *ICCBR*, pages 404–418, 2007.

[20] Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *In Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[21] Wolfgang Hesse. *Modellierung in der Softwaretechnik: eine Bestandsaufnahme.* Informatik Spektrum, 2008.

[22] Janet Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, Inc., 2929 Campus Drive, Suite 260, San Mateo CA 94403, 1993.

[23] Janet L. Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7:243–280, 1983.

[24] Martin Kost. Grundlagen des Semantic Web. zuletzt abgerufen am 25.05.2011.

`http://www.is.informatik.uni-duisburg.de/courses/db_ws04/folien/owl.pdf`.

[25] David B. Leake and David C. Wilson. When Experience is Wrong: Examining CBR for Changing Tasks and Environments. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 218–232. Springer Verlag, 1999.

[26] Mario Lenz, Brigitte Bartsch-Spörl, Hans-Dieter Burkhard, and Stefan Wess, editors. *Case-Based Reasoning Technology, From Foundations to Applications*, volume 1400 of *Lecture Notes in Computer Science*. Springer, 1998.

[27] Mario Lenz, André Hübner, and Mirjam Kunze. *Textual CBR*, chapter 5, pages 115–137. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1998.

[28] Mario Lenz, André Hübner, and Mirjam Kunze. Textual CBR. In *Case-Based Reasoning Technology*, pages 115–138, 1998.

[29] Alexander Maedche and Steffen Staab. Ontology Learning. In *HANDBOOK ON ONTOLOGIES*, pages 173–189. Springer.

[30] Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web. Website, zuletzt abgerufen am 25.05.2011. `http://www.aifb.uni-karlsruhe.de/WBS`.

[31] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification. In *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pages 41–48. AAAI Press, 1998.

[32] Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. Review and Restore for Case Base Maintenance. *Computational Intelligence: Special Issue on Maintaining Case-Based Reasoning Systems*, 17(2):214–234, 2001.

[33] Jens Rohler. Web Ontology Language OWL. Seminararbeit, Lehrstuhl KI, Universität Frankfurt, Frankfurt.

[34] R. C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, MA, 1982.

[35] Markus Schwinn. Automatische Informationsextraktion aus Dokumenten zum Aufbau eines Wissensarchivs als Grundlage für eine automatisierte Angebotserstellung. Masterarbeit, FH Trier, 2010.

[36] Mehrnoush Shamsfard and Ahmad Adollahzadeh Barforoush. The state of the art in ontology learning: a framework for comparison. *Knowl. Eng. Rev.*, 18:293–316, December 2003.

[37] S. Staab and R. Studer. *Handbook on Ontologies.* Springer Verlag, Berlin, Heidelberg, 2004.

[38] Armin Stahl. Lernen von Wissensintensiven Ähnlichkeitsmaßen im Cased-Based Reasoning. *Künstliche Intelligenz*, 3:69–71, April 2004. ISSN 0933-1875, arendtap Verlag, Bremen.

[39] Benno Stein, Michael Suermann, Hans Kleine Büning, Hans-Werner Kelbassa, Andreas Reckmann, Rainer Tellmann, Carsten Thurau, and Hans-Gerd Wiegard. Fallbasiertes Schließen – Grundlagen und Anwendungen für Konstruktions- und Entwurfsaufgaben. Notes in Computer Science tr-ri-07-279, University of Paderborn, Germany, Computer Science Institute, January 2007.

[40] Hristo Tanev Tanev. Weakly Supervised Approaches for Ontology Population. In *Proceedings of EACL-2006*, pages 3–7, 2006.

[41] Madeleine Theile. Active Hidden Markov Models for Information Extraction. Website, zuletzt abgerufen am 25.05.2011. `http://www-ai.cs.uni-dortmund.de/LEHRE/SEMINARE/INFORMATIONSEXTRAKTION/theile.pdf`.

[42] E. Tulving. *Organization Memory.* Academic Press, 1972. Chapter: Episodic and semantic memory.

[43] Reinhold Taucher und Marina Lenger. Ontologie Population und Ontologie Lernen. Website, zuletzt abgerufen am 25.05.2011. `http://my-svn.assembla.com/svn/seminarOnto/seminararbeit.pdf`;.

[44] Alexandros G. Valarakos, Georgios Paliouras, Vangelis Karkaletsis, and George A. Vouros. Enhancing Ontological Knowledge Through Ontology Population and Enrichment. In *EKAW'04*, pages 144–156, 2004.

[45] Paola Velardi, Roberto Navigli, Alessandro Cucchiarelli, and Francesca Neri. Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. Website, zuletzt abgerufen am 25.05.2011. `http://www.dsi.uniroma1.it/~velardi/IOS-FAIA.pdf`;.

[46] René Witte and Jutta Mülle, editors. *Text Mining: Wissensgewinnung aus natürlichsprachigen Dokumenten*, Interner Bericht 2006-5. Universität Karlsruhe,

Fakultät für Informatik, Institut für Programmstrukturen und Datenorganisation (IPD), 2006. ISSN 1432-7864, http://digbib.ubka.uni-karlsruhe.de/volltexte/1000005161.

[47] Holger Wunsch. Der Baum-Welch Algorithmus für Hidden Markov Models, ein Spezialfall des EM-Algorithmus. Website, zuletzt abgerufen am 25.05.2011. http://www.sfs.uni-tuebingen.de/resources/em.pdf;.

[48] Nancy R. Zhang. Hidden Markov Models for Information Extraction, 2001.

[49] Lina Zhou. Ontology learning: state of the art and open issues. *Information Technology and Management*, 8(3):241–252, 2007.