Masterarbeit

Domänenmodellierung im Textuellen Fallbasierten Schließen

Kerstin Bach

Universität Hildesheim
Fachbereich III: Informations- und Kommunikationswissenschaften
Institut für Informatik – AG Intelligente Informationssysteme
Betreuung: Prof. Dr. Klaus-Dieter Althoff
Dipl.-Inform. Alexandre Hanft

Inhaltsverzeichnis

1 Einleitung

Knowledge is like a garden;
If it is not cultivated, it cannot be harvested.
(Afrikanisches Sprichwort)

Erfahrungen sind wichtige Bestandteile für die Erstellung und Weiterverwendung von Produkten – jedoch liegen die meisten Erfahrungen bei den Menschen, die sie gemacht haben und sind für andere nicht zugänglich. Um Erfahrungen auch für andere bereitzustellen, wurden in den 1980er Jahren Expertensysteme entwickelt. Zuerst sollten diese Systeme den menschlichen Experten ersetzen – heute weiß man, dass dies nicht möglich ist. Allerdings können Erfahrungen erfasst und bei Bedarf bereitgestellt werden, so dass Mitarbeiter in einem Unternehmen von dem Wissen der anderen Mitarbeiter profitieren können. Erfolgreiche Erfahrungsmanagementsysteme sind in den alltäglichen Ablauf der Mitarbeiter eingebunden [AlthoffEtAl2006].
Darüber hinaus gibt es aber Dokumente in denen Informationen bereits enthalten sind, die zum Aufbau eines Erfahrungsmanagementsystems genutzt werden können wie beispielsweise FAQs oder Produktbeschreibungen. Solche Dokumente unterliegen meist einer einfachen Struktur (Überschrift, Beschreibung etc.) und sind in den Textpassagen frei formuliert. Um die semantische Bedeutung dieser Passagen zu erfassen werden spezielle Methoden angewandt. Im Erfahrungsmanagement erfolgreich eingesetzte Systeme sind zum Beispiel Anwendungen des Fallbasierten Schließens [AlthoffEtAl2006].
Eine Art des Fallbasierten Schließens befasst sich speziell mit der Verarbeitung von Freitexten: das Textuelle Fallbasierte Schließen. Für diese Technik stellt die vorliegende Arbeit Konzepte vor, wie unstrukturierte Texte erfasst und weiterverarbeitet werden können. Die Vorverarbeitung von Texten geht davon aus, dass Dokumente, deren Anwendungsbereich und Vokabular bekannt ist, besser verwertet werden können.
Diese Arbeit legt den Schwerpunkt auf das unbekannte Vokabular in einer für das Fallbasierte System neuen Domäne, welches durch die Integration heterogener Wissensquellen erweitert werden soll. Eine weitere Zielsetzung liegt darin, die Erweiterung eines vorhandenen Vokabulars so weit wie möglich zu automatisieren, damit lediglich Worte, die speziell in einem Anwendungsbereich vorkommen, manuell modelliert werden müssen. Im Rahmen dieser Masterarbeit wurde ein Werkzeug entwickelt, welches einem Modellierer helfen soll, eine unbekannte Domäne speziell für das Textuelle Fallbasierte Schließen zu erfassen.

Die Arbeit beginnt mit der Erläuterung des Fallbasierten Schließens. Zuerst wird das Konzept vorgestellt und im Anschluss daran werden mögliche Aufgaben für Fallbasierte Systeme erläutert. Das Kapitel stellt im Weiteren die drei grundlegenden Arten des Fallbasierten Schließens vor und geht am Ende auf indexbasierte Retrievalverfahren ein.
Im dritten Kapitel wird ein indexbasiertes Retrievalverfahren, das Retrieval mit Case Retrieval Netzen, vorgestellt, denn die im Rahmen dieser Arbeit unterstützte Domänenmodellierung hat zum Ziel, Domänenmodelle für diese Art des Fallbasierten Schließens zu erstellen.
Kapitel 4 stellt die Domänenmodellierung vor und definiert die Begriffe Domäne und Domänenmodellierung. Weiterhin werden in diesem Kapitel die Vorgehensweisen bei der Erstellung eines Domänenmodells in angrenzenden wissenschaftlichen Bereichen betrachtet, um abschließend eine Einordnung der Domänenmodellierung in das Textuelle Fallbasierten Schließen vorzunehmen.
Das fünfte Kapitel konzentriert sich auf die Umsetzung der Wissensakquisition als Vorbereitung für die Domänenmodellierung. Darin wird erläutert, wie unbekannte Texte erfasst werden können und was notwendig ist, um einen unbekannten Anwendungsbereich zu erschließen.
Im sechsten Kapitel werden die Oberfläche und Teile der Implementierung der Anwendung vorgestellt, die erfahrene Modellierer unterstützen sollen, ein Domänenmodell für eine neu zu erfassende Datenbank zu erzeugen.
Abschließend wird im siebten Kapitel die Evaluierung der im Rahmen der Arbeit erstellten Konzepte an einem konkreten Beispiel ausgewertet und das achte Kapitel fasst die Erkenntnisse der Arbeit zusammen, blickt auf verwendete Ansätze und gibt einen Ausblick, wie die Domänenmodellierung erweitert werden kann.

2 Fallbasiertes Schließen

Die Idee des fallbasierten Schließens (FBS, engl. Case-Based Reasoning, CBR) entstammt dem psychologischen Modell, dass Menschen neue Probleme lösen, indem sie gelöste Probleme aus der Vergangenheit (Erfahrungen) auf das aktuelle Problem anwenden [vgl. RiebeckSchank1989].
Die Erfahrungen werden im FBS als Fälle in Form eines Problems und seiner dazugehörigen Lösung erfasst. Darüber hinaus benutzt der Mensch Erklärungen und weiterreichende Informationen zur Lösungsfindung, die nicht unbedingt zu einem konkreten Fall gehören, sondern der Erfahrung beim Problemlösen entstammen. Dies ist das Grundmodell eines FBS-Systems in dem Fälle aus der Vergangenheit genutzt werden, um aktuelle Probleme zu lösen.
Erfahrungen bilden auch in anderen wissensbasierten Systemen die Grundlage für deren Aufbau, jedoch wird das Wissen in FBS-Systemen anders genutzt, so dass Richter diese Anwendungen als Fortentwicklung wissensbasierter Systeme ansieht [vgl. Richter2003 S.407f].
Die Grundlagen für die Entwicklung dieser Methode entstammen den Arbeiten von [Schank1982] und [Kolodner1993]. Beim Ansatz des Dynamischen Gedächtnisses von Schank wird ein psychologisches Modell über menschliches Problemlösen und Lernen zu Grunde gelegt nach dem Erinnern und Anpassen (Adaptieren) zentrale Prozesse beim Verstehen darstellen. Darüber hinaus nimmt Schank an, dass die Indexierung für das Erinnern wichtig ist und Verstehen dazu führt, dass im Gedächtnis eine Reorganisation vorgenommen wird. Als weiteren Aspekt geht er davon aus, dass für die Wissensverarbeitung wie auch für die Wissensspeicherung dieselben Gedächtnisstrukturen genutzt werden [vgl. RiebeckSchank1989].

Das FBS ist ein Teilgebiet der Künstlichen Intelligenz und die Systeme werden zu den wissensbasierten Systemen mit dem Fokus auf Problemlösen durch Erfahrung gezählt, wobei Techniken zur Repräsentation, Speicherung, Indexierung und Adaption von Erfahrungswissen angewandt werden [vgl. Bergmann2002 S. 18ff].

Dieses Kapitel erläutert den Grundaufbau eines FBS-Systems und gibt einen Überblick über die Arten von FBS-Systemen und deren mögliche Aufgaben.

2.1 CBR-Zyklus nach Aamodt und Plaza

Mitte der 1990er Jahre haben Aamodt und Plaza in [Aamodt1994] den abstraktesten CBR-Zyklus bestehend aus vier Prozessen vorgestellt. Wie Abbildung 1 zeigt, besteht der Zyklus aus den Schritten Retrieve, Reuse, Revise und Retain.

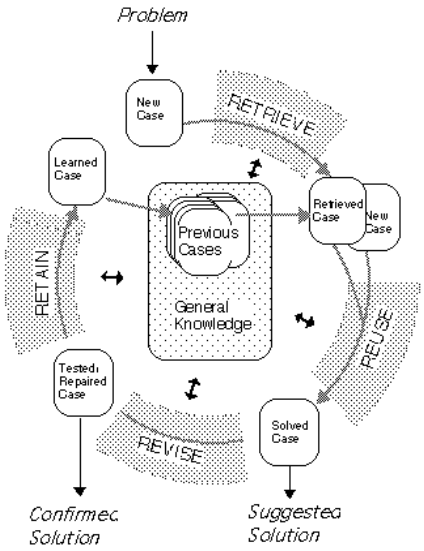


Abbildung 1: CBR-Zyklus nach Aamodt und Plaza [vgl. Aamodt1994]

Angestoßen wird der Zyklus mit einem neuen Problem (in Form einer Anfrage), welches mit Hilfe der Fallbasis gelöst werden soll. Die Anfrage beschreibt einen neuen Fall, für den im Rahmen der folgenden vier Prozesse eine Lösung gesucht wird. Die folgenden Erläuterungen wurden [Aamodt1994] und [Richter2003 S.411] entnommen.

Retrieve: Anhand der Anfrage selektiert das System einen geeigneten, problemlösenden Fall aus der Fallbasis. Der gewählte Fall muss nicht zwingend die vollständige Lösung beinhalten – er stellt jedoch den zur Anfrage ähnlichsten Fall dar.

Reuse: Der zurück gelieferte Fall wird entweder als Lösung übernommen oder für die Lösungsadaption durch Regeln weiter genutzt.

Revise: Unter dem dritten Schritt werden alle Verfahren zusammengefasst, die das Testen der gefundenen Lösung auf Korrektheit oder Tauglichkeit beschreiben. Es wird geprüft, ob der vorgeschlagene Fall zur Lösung des aktuellen Problems genutzt werden kann.

Retain: Nachdem im Revise-Schritt die neue Lösung bestätigt wurde, wird diese in der Retain-Phase in die Fallbasis eingefügt, so dass die Fallbasis am Ende des Zyklus um einen gelernten Fall (den neuen Fall) erweitert wird. Der neu erfasste Fall besteht aus der Problemstellung und dessen Lösung [vgl. Hanft2004 S.14f].

Für jede Anfrage wird der CBR-Zyklus durchlaufen. Allerdings beschreibt der Zyklus in Abbildung 1 nur die allgemeinen vier Schritte. In der konkreten Anwendung gliedern sich diese Schritte in weitere Teilschritte auf [vgl. Aamodt1994].

2.2 Wissenscontainer im Fallbasiertes Schließen

Den Begriff Wissenscontainer hat Richter 1995 in [Richter1995] geprägt, um die Arten von Wissen zu beschreiben, die im FBS benötigt werden. Er beschreibt sie als strukturelle Elemente, die selbst keine Aufgaben lösen, aber zur Lösung der Probleme genutzt werden [vgl. Richter2003 S.415f].
In fallbasierten Systemen gibt es vier Wissenscontainer:

1. Repräsentationssprache und Vokabular
2. Ähnlichkeitsmaß
3. Lösungstransformation
4. Fallbasis

In einem fallbasierten System kann jeder Container das gesamte Wissen enthalten, jedoch ist es in der Regel über alle Container verteilt. Richter unterteilt die Wissenscontainer weiter in kompiliertes (Vokabular, Ähnlichkeitsmaß und Lösungstransformation) und interpretiertes Wissen (Fallbasis). Das kompilierte Wissen liegt zur Laufzeit bereits vor und wird genutzt – es kann im Gegensatz zum interpretierten Wissen nicht verändert werden. Die Fallbasis dagegen verändert sich wie von Aamodt und Plaza [vgl. Abschnitt 2.1] beschrieben auch während der Laufzeit. Im Laufe der Zeit bleibt das Wissen nicht zwingend in einem Container, sondern kann in andere Wissenscontainer transferiert werden [vgl. Richter2003 S.407f].
Ein FBS-System kann nicht universell ohne domänenabhängige Vorarbeit aufgesetzt werden, denn je nach Anwendung unterscheiden sich die betrachteten Fälle. So zum Beispiel weicht ein Fall, der einen Defekt eines Flugzeugmotors beschreibt sehr von einem anderen Fall, der ein Produkt einer eCommerce[1]-Anwendung erfasst, ab, weil unter Anderem bei der Aufnahme von sicherheitsrelevanten Fällen kontrolliertes Vokabular[2] und mehrfache Prüfungen vorgeschrieben sind, wogegen eine Buchbeschreibung beliebig gestaltet sein kann.

2.3 Aufgabenklassen des Fallbasiertes Schließen

Je nach Struktur der gegebenen Fälle und der konkreten Anwendung des Systems unterscheidet man im Fallbasierten Schließen zwischen den nachfolgenden Aufgaben, die von einem FBS-System gelöst werden können.

[1] Unter den Begriff des eCommerce (dt: Elektronischer Geschäftsverkehr) fallen alle Arten von Geschäften, die auf elektronischem Wege abgewickelt werden können. Dies umfasst die verschiedensten unternehmerischen Tätigkeiten, wie z.B. die Geschäftsanbahnung und -abwicklung, Werbung, Online-Banking aber auch Kundenservice [vgl. http://www.ecommerce-verbindungsstelle.de/].
[2] Ein kontrolliertes Vokabular ist eine Sammlung von Bezeichnungen (Wortschatz), die eindeutig Begriffen zugeordnet sind, so dass keine Homonyme auftreten. [Wikipedia2007]

2.3.1 Analytische Aufgaben

Der Schwerpunkt der Anwendungen aus diesem Bereich liegt in der Analyse einer vorliegenden Situation, denn um das gestellte Problem zu Lösen ist es notwenig die Situation in eine Klasse einzuordnen. Dafür ist im System eine Reihe von Klassen fest vorgegeben und je nach gestellter Aufgabe kommen weitere Problemschritte hinzu. FBS-Systeme für analytische Aufgaben bestehen aus Fällen, die zum einen eine Beschreibung der jeweiligen Situation beinhalten und im Zuge dessen eine Klassifizierung des Falles vornehmen. Der Schwerpunkt solcher Systeme liegt im Retrieval, da meist auf eine Lösung verzichtet werden kann. Denn die Lösung ist im zurück gelieferten Fall enthalten. Diese Art von fallbasierten Systemen findet in vielen Anwendungen des täglichen Lebens ihren Einsatz [vgl. Althoff2005, Foliensatz 2].
Beispiele dafür sind:

- Technische Diagnose:
  Die Wartung von Flugzeugtriebwerken anhand von strukturiertem Erfahrungswissen [vgl. StolpmannWess1998]
- Entscheidungsunterstützung:
  Unterstützung bei der Suche nach Produktinformationen im Internet
- Bewertung:
  Kreditwürdigkeitsprüfungen von Bankkunden oder Kostenschätzungen von Wohnhäusern
- Vorhersage:
  Vorhersage von Ausfallwahrscheinlichkeiten von technischen Teilen

2.3.2 Synthetische Aufgaben

Das Hauptaugenmerk bei der Lösung Synthetischer Aufgaben liegt darin, komplexe Lösungen aus einzelnen Bestandteilen zusammenzusetzen. Dabei wird in der Problembeschreibung meist die Anforderung an die Lösung spezifiziert. Typischerweise bestehen die Fälle aus der Problembeschreibung (Problemspezifikation) und der dazugehörigen Lösung bzw. des Lösungsweges. Im Gegensatz zu den analytischen Aufgaben liegt der Schwerpunkt in der Lösungsanpassung [vgl. Althoff2005, Foliensatz 2].
Anwendungsgebiete für synthetische Problemstellungen sind:

- Planung:
  Bestimmung einer komplexen zeitlichen Abfolge von Aktionen, die selbst aus vielen parametrisierbaren Eigenschaften bestehen. Ein Beispiel dafür ist die Produktionsplanung oder Personaleinsatzplanung.
- Konfiguration
  Zusammenstellung einer Lösung aus beliebig vielen vorgegeben Komponenten, die voneinander abhängen, wie beispielsweise die PC-Hardwarekonfiguration. Dabei sind die einzelnen Komponenten und ihre Abhängigkeit untereinander bekannt und das System stell eine gültige Konfiguration entsprechend der Anforderungen zusammen.

- Design
  Ähnlich wie bei der Konfiguration steht die Zusammenstellung einer Lösung aus beliebig vielen Variationen, die voneinander abhängen im Mittelpunkt. Eine Anwendung dafür ist der Systementwurf einer Anwendungssoftware.

2.4 Arten des Fallbasierten Schließen

Die Art des FBS-Systems hängt davon ab, wie die Fälle in der Fallbasis repräsentiert werden und was für eine Art von Wissen gewonnen werden soll. In den vergangenen Jahren wurden bei der Entwicklung von fallbasierten Systemen verschiedene Fallrepräsentationen implementiert, die in den meisten Anwendungen eine der folgenden Techniken für das fallbasierte Schließen verwendet: *Strukturelles Fallbasiertes Schließen*, *Dialogbasiertes Fallbasiertes Schließen* und *Textuelles Fallbasiertes Schließen.* Diese Techniken werden in Anlehnung an die Anführungen in [BergmannEtAl2003] nachfolgend vorgestellt.

2.4.1 Strukturelles Fallbasiertes Schließen

Im strukturellen FBS werden die Fälle als eine Menge vordefinierter Attribute repräsentiert. Je nach Anwendung sind die Attribute in einfachen Tabellen (flat tables), einem relationalen Modell oder Objektmodell organisiert. In einfachen Anwendungen reichen Tabellen aus, um das Wissen abzubilden, wogegen komplexere Ansätze zur Darstellung die anderen beiden Modelle benutzen. In einem relationalen Modell werden die einzelnen Tabellen durch einen Primärschlüssel indexiert. Weiterhin wird jeder Fall über einen eindeutigen Schlüssel identifiziert. Die Fallbeschreibungen liegen über mehrere Tabellen verteilt vor und sind über Relationen untereinander verbunden. Das Objektmodell besteht aus einzelnen Objekten, die wiederum in Unterobjekte aufgeteilt werden können. Jedes Objekt wird durch eine Menge von Attributen genauer beschrieben. Als Besonderheit des strukturellen fallbasierten Schließens hat jedes Attribut einen präzisen Typ, der vorab bekannt ist. Ein solcher Typ kann ein Symbol (bestehend aus einem oder mehreren vordefinierten Werten) einer Nominal-, Ordinal-, Intervall- oder Verhältnisskala sein. Um ein Symbol zu definieren reichen die Informationen aus einem Fall nicht aus, so dass es weiteren Hintergrundwissens bedarf.
Dieser Ansatz eignet sich, wenn die Problemstellungen als Attribut-Werte-Paare erfasst werden können und neben den eigentlichen Fällen noch zusätzliches Wissen zum Beispiel zur Beschreibung der Ähnlichkeit oder Lösungstransformation bereit steht. Aus diesem Grund muss vorab ein Domänenmodell [vgl. ManagoEtAl1994, Bergmann2002] definiert werden, was sicherstellt, dass neue Fälle qualitativ hochwertig in das System eingepflegt werden, wodurch der Wartungsaufwand der Fälle deutlich verringert wird. Das Domänenmodell legt die genannten *features* fest, mit denen die Fälle repräsentiert werden. Als *feature* bezeichnet Bergmann [Bergmann2002] eine Menge an Attributen, die durch das Domänenmodell aufgezeigt werden. Das Hintergrundwissen über mögliche Ausprägungen der Attribute wie auch der Zusammenhang zwischen den einzelnen Attributen ermöglicht es, die

Qualität des Retrievals signifikant zu verbessern, weil dieses Wissen dazu dient, weitere Ähnlichkeiten zwischen zwei Fällen zu bestimmen.
Zusammenfassend wird im strukturellen fallbasierten Schließen die Fallbasis aus Einträgen einer Datenbank erstellt, deren Felder standarisiert wurden. Neue Fälle werden nur in der Form in die Fallbasis aufgenommen, wie sie das Domänenmodell beschreibt. Die Anfrage an ein solches System erfolgt durch eine Menge von Attributwerten deren Ausprägungen vorgegeben sind und durch die Anfrage instanziiert werden. Für den Aufbau eines strukturellen fallbasierten Systems ist Hintergrundwissen zur Erstellung des Domänenmodells, zur Definition von Regeln sowie Berechnung der Ähnlichkeiten notwendig.

2.4.2 Dialogbasiertes Fallbasiertes Schließen

Die Idee des Dialogbasiertes Fallbasiertes Schließen [vgl. AhaBreslow1997, AhaEtAl2001] (engl. Conversational CBR) entstammt dem Prinzip, dass Wissen im Dialog zwischen Kunden und Dienstleistern besteht. Somit wird ein Fall durch eine Liste von Fragen repräsentiert und wobei sich diese Liste je nach Dialog verändern kann. Aus diesem Grund ist es nicht möglich ein Domänenmodell oder eine standardisierte Struktur für alle Fälle vorzugeben, denn dies könnte dazu führen, dass das System irrelevante Fragen stellt. Um dies zu verhindern werden die Fragen in baumähnlichen Strukturen organisiert, in denen mehrere Fragen eine bestimmte Menge an Attributen der Fallbasis repräsentieren. Die Fragen und deren Zuordnung werden vom Modellierer des FBS-Systems übernommen, so dass es auch seine Aufgabe ist, neue Fälle von Hand in die Struktur der Fallbasis (des Baums) einzupflegen. Zur Indexierung der Fälle muss der Modellierer festlegen, in welcher Reihenfolge Fragen gestellt werden.
Zusammenfassend besteht die Fallbasis aus einer Liste mit Fragen und Antworten, die von Fall zu Fall variieren können. Es gibt keine allgemeine Datenstruktur der Fallbasis. Die Anfrage erfolgt im Dialog mit dem Nutzer, indem dieser die vom System gestellten Fragen beantwortet. Das Resultat einer jeden Antwort hat entweder eine neue Frage zur Folge, die die Lösung einschränkt oder liefert basierend auf den vorangegangen Antworten ein Ergebnis (den gefundenen Fall). Zum Aufbau eines solchen Systems wird Hintergrundwissen zur Anordnung der Fragen in einer baumartigen Struktur benötigt sowie Antwortmöglichkeiten, die wiederum zur weiteren Navigation durch den Baum dienen.
Prädestiniert als Anwendung für ein Dialogbasiertes Fallbasiertes System sind Call Center Lösungen für die Unterstützung von Kunden, denn die Supportanfragen sind ähnlich und können gut in einer Baumstruktur abgebildet werden. Zudem sind die gestellten Fragen relativ einfach aber häufig, so dass durch die Systemunterstützung die Mitarbeiter den Kunden im First-Level-Support schnell weiterhelfen können. Weiterhin haben alle Mitarbeiter durch eine gemeinsame Fallbasis den gleichen Wissensstand und die Qualität der Beratung ist entsprechend gleichwertig. Falls ein Problem nicht gelöst werden kann, besteht in der Regel die Möglichkeit, es an den Second-Level-Support weiterzuleiten. Im Second-Level-Support neu erar-

beitete Lösungen werden zudem in die Fallbasis eingepflegt um das Wissen für den First-Level-Support nutzbar zu machen.

2.4.3 Textuelles Fallbasiertes Schließen

Die dritte Art ein fallbasiertes System zu erstellen ist das textuelle fallbasierte Schließen, dessen Ausgangpunkt so genannter Freitext ist, wie beispielsweise aus Produktbeschreibungen oder FAQ-Dokumenten. Somit liegt ein Fall im Freitextformat vor und zum Aufbau der Fallbasis bedarf es der Extraktion der Informationen aus dem Fall, um ihn für das Retrieval erfassbar zu machen. Freitextdokumente beschreiben nicht zwingend unstrukturierte Dokumente – so kann Freitext auch in einer Struktur (E-Mail, Produktbeschreibung mit festgelegten Abschnitten, Erläuterungen im Reisekatalog) vorliegen. Die einzelnen Textpassagen sind freie Formulierungen. Die Anfrage wird anhand differenzierender Schlagwörter mit den vorhandenen Fällen verglichen, mit dem Ziel den Fall mit dem ähnlichsten Inhalt zurückzuliefern. Das benötigte Hintergrundwissen ist ein Indexvokabular mit Informationen über die Beziehung zwischen den Worten.
Das textuelle fallbasierte Schließen steht im Fokus dieser Arbeit und wird ab Abschnitt 3 im Detail vorgestellt.

2.5 Retrieval im Fallbasierten Schließen

Nach der Repräsentation der Fälle in der Fallbasis ist das nächste auszuführende Schritt im fallbasierten System das Retrieval. Anhand der Anfrage sollen ähnliche Fälle in der Fallbasis gesucht und zurückgeliefert werden. Wie in Abschnitt 2.4 beschrieben gibt es verschiedene Arten des FBS und von der Art des Systems hängt auch das Retrieval ab, da es auf die Fallrepräsentation (vgl. Definition 3) abgestimmt sein muss. Dialogbasierte FBS-System verfolgen einen festgeschriebenen Auswahl-Ansatz wogegen das Retrieval im textuellen fallbasierten über erkannte Schlagworte erfolgt. Systeme, in denen der strukturelle Ansatz implementiert wurde, nutzen vorgegebene Algorithmen zur Bestimmung der Ähnlichkeit.
Gemein haben alle Ansätze, dass zuerst die lokalen Ähnlichkeiten und darauf basierend die globalen Ähnlichkeiten berechnet werden.
Die lokale Ähnlichkeit beschreibt die Beziehung zwischen zwei Fallbestandteilen. Im strukturellen FBS können dies zwei *features*, im textuellen FBS zwei Worte sein. Die globale Ähnlichkeit ergibt sich zum Beispiel aus der gewichteten Summe der lokalen Ähnlichkeiten [BergmannEtAl2003, S.18f].
Bei großen Fallbasen muss ein Index erstellt werden, damit im Retrieval ein effizienterer Zugriff auf die Fälle erfolgen kann. Es gibt verschieden Möglichkeiten einen Index für eine Fallbasis aufzubauen. Neben den in Abschnitt 3 vorgestellten Case-Retrieval-Netze gibt es im fallbasierten Schließen noch Fish & Shrink [Schaaf1998] sowie die kd-Bäume [FriedmanEtAl1977, Wess1995] als Indexorientierte Retrievalverfahren.

2.5.1 Retrieval mit kd-Bäumen

Beim Retrieval mit kd-Bäumen (kd: k-dimensional) sollen die Fälle möglichst homogen in einem Baum repräsentiert werden. Dazu wird die Fallbasis in immer kleinere Intervalle zerteilt und in einem binären Baum angeordnet. Ein Blattknoten repräsentiert eine Teilmenge der Fallbasis, die nicht weiter partitioniert werden soll. Falls die Einschränkung am erreichten Blattknoten kein Ergebnis liefert, kann im Gegensatz zu vergleichbaren Entscheidungsbaumverfahren über Backtracking zum vorherigen Knoten gesprungen und das Retrieval fortgesetzt werden. An jedem inneren Knoten wird die Fallbasis abhängig von den Werten des betrachteten Attributes weiter geteilt. Aus diesem Grund ist die Auswahl der Attribute beispielsweise nach dem größten Interquartilsabstand[3] oder der Entropie möglich. Für die Auswahl der Werte gibt es unter Anderem die Maximumsplittung und die Mediansplittung weitere Verfahren. Die Maximumsplittung sucht den größten Abstand zwischen zwei (metrischen) Werten und setzt dort die Splittung der Werte an. Die Mediansplittung ermittelt den Median[4] der Werte und setzt dort die Partitionierung an.

```
PROCEDURE CreateTree(FB): kd-tree
      IF |FB| ≤ b
            THEN RETURN Blattknoten mit Fallbasis FB
      ELSE
            A_i := wähle_Attribut(FB);
            v_i := wähle_Wert(FB,A_i)
      RETURN
            Baum mit der Wurzel, die mit A_i und v_i markiert ist, und
            die beiden Teilbäume
                  CreateTree({(x_1,...,x_k) ∈ FB | x_i ≤ v_i}) und
                  CreateTree({(x_1,...,x_k) ∈ FB | x_i > v_i}) enthält.
```

Abbildung 2: Algorithmus zum Aufbau eines kd-Baums nach [Wess1995]

Abbildung 2 zeigt den Algorithmus zum Aufbau eines kd-Baumes: Zuerst wird überprüft, ob alle Fälle der Fallbasis im kd-Baum enthalten sind, so dass jener Blattknoten (ein so genanntes Bucket) zurückgeliefert wird, der Ausgangspunkt für die Repräsentation der Fallbasis ist. Gibt es jedoch Fälle in der Fallbasis, die noch nicht im kd-Baum repräsentiert werden, so wird ein Attribut aus der Fallbasis ausgewählt (beispielsweise nach dem größten Interquartilsabstand) und die dazugehörigen Werte ermittelt. Für die Werte wird die Splittung wie zuvor beschrieben angesetzt und der Algorithmus liefert einen Teilbaum, genauer gesagt dessen Wurzel, zurück. Das Attribut und dessen Werte der Wurzel sind markiert und verweisen auf zwei Teilbäume, die durch die Splittung anhand des Attributes entstanden sind [Wess1994].

[3] Der Interquartilsabstand ist als der Abstand zwischen dem ersten und dem dritten Quartil definiert. Es ist zu beachten, dass der IQR genau 50 % der Daten innerhalb der Verteilung enthält. [vgl. http://www.statistics4u.info/fundstat_germ/cc_iqr.html]

[4] Der Median (oder Zentralwert) bezeichnet eine Grenze zwischen zwei Hälften. [vgl. Wikipedia2007]

```
PROCEDURE Retrieve(K: kd-baum)
      IF K ist Blattknoten
            THEN
            FOR jeden Fall F von K DO
                  IF sim(Q,F) > scq[m].similarity
                  THEN füge F in scq ein
      ELSE (* innerer Knoten *)
            Sei Ai das Attribut und vi der Wert, mit dem K
            markiert ist
            IF Q[Ai] ≤ vi
                  THEN Retrieve(K≤)
                        IF BOB-Test ist erfüllt
                        THEN Retrieve(K>)
            ELSE
                  Retrieve(K>)
                        IF BOB-Test ist erfüllt
                        THEN Retrieve(K≤)
                        IF BWB-Test ist erfüllt
                        THEN terminiere Retrieval mit scq
      ELSE RETURN
```

Abbildung 3: Algorithmus zum Retrieval in einem kd-Baum nach [Wess1995]

Abbildung 3 zeigt den Retrievalalgorithmus in einem kd-Baum. Beim Retrieval wird der Baum bis zu einem Blattknoten (Bucket) durchlaufen, in dem eine bestimmte Menge an Fällen abgelegt ist. Dabei ist es wichtig, dass die Ähnlichkeit nach einem vorgegebenen Ähnlichkeitsmaß bestimmt wurde, damit die für den aktuellen Fall ähnlichsten Fälle betrachtet werden können. Jeder betrachtete Fall wird in eine Queue aufgenommen, die abgearbeitet wird, indem die Fälle in so genannte Hyperkugeln platziert werden. Hyperkugeln haben in ihrem Zentrum die Query (Q) und im kreisförmigen Raum um den Fall liegen die ähnlichsten Fälle aus der Queue (vgl. Abbildung 4). Der Abstand im n-dimensionalen Raum zwischen der Query und jedem Fall entspricht der Ähnlichkeit des Falles zur Query gemäß dem gewählten Ähnlichkeitsmaß. Anhand der Hyperkugeln wird überprüft, ob der betrachtete Fall in Frage kommt. Dazu werden die BOB und BWB-Tests genutzt [Wess1994].

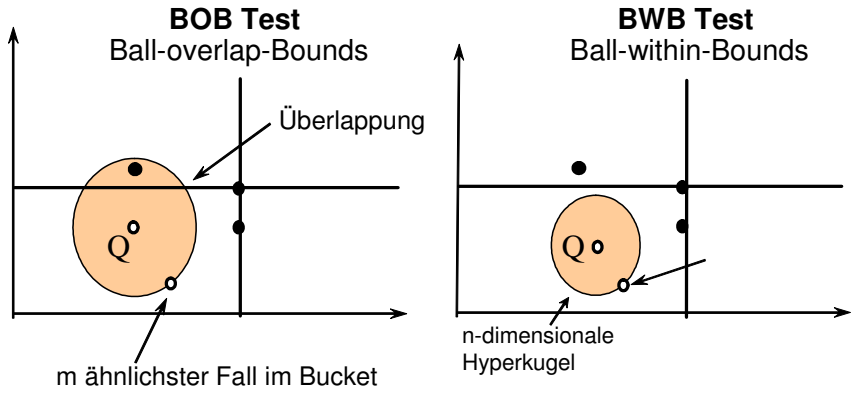


Abbildung 4: BOB und BWB-Tests nach [Wess1995]

Der BOB-Test (Ball-Overlaps-Bounds) überprüft, ob es ähnlichere Fälle zur Query als den bisher gefundenen Fall gibt (vgl. Abbildung 4, linke Darstellung). Eine Überlappung würde bedeuten, dass es ähnlichere als den bisher ähnlichsten gefundenen

Fall gibt. Der BWB-Test (Ball-Within-Bounds) stellt fest, ob sich die durch den bisher ähnlichsten Fall gezeichnete Hyperkugel in den festgesetzten Grenzen liegt (vgl. Abbildung 4, rechte Darstellung). Damit soll sichergestellt werden, dass alle relevanten Fälle betrachtet werden. Schlagen diese Tests fehl wird durch Backtracking der betrachtete Raum vergrößert, so dass alle relevanten Fälle berücksichtigt werden [Wess1994].

2.5.2 Retrieval mit Fish & Shrink

Im Fish & Shrink-Ansatz [Schaaf1998] sind die Gewichtungen der Attribute einer Problemstellung nicht festgeschrieben, sondern werden durch die Anfrage selbst festgelegt werden. Dieser Ansatz orientiert sich an Anwendungen mit komplexen Ähnlichkeitsberechnungen [Bergmann2002, S. 200ff].
Die Fallrepräsentation ist in verschiedene Aspekte unterteilt, die eigene Ähnlichkeitsmaße definiert haben. Ein Aspekt kann eine komplexe Eigenschaft wie beispielsweise ein Röntgenbild oder ein Netz aus Attributen sein. Weiterhin wird angenommen, dass für die Ähnlichkeitsmaße die Dreiecksungleichung gilt. Die Idee besteht nun darin, dass in der Fallbasis bereits berechnete Aspektdistanzen zwischen den Fällen (nicht unbedingt zwischen allen Fällen) enthalten sind. Damit repräsentiert eine Anfrage zusätzlich eine Sicht (Gewichtsvektor) auf die Fallbasis [Richter2003, S. 426].
Das Retrieval beginnt mit der Bestimmung eines Distanzintervalls für jeden Fall der Fallbasis. Im nächsten Schritt wird ein Testfall gewählt, welcher die Sichtdistanz zwischen dem gewählten Fall und der Anfrage berechnet. Als nächstes wird für viele Fälle aus der Fallbasis das neue verkleinerte Distanzintervall unter Ausnutzung der Distanzungleichung bestimmt. Als letztes werden so lange Testfälle ausgewählt, bis das Distanzintervall klein genug ist und ein passender Fall gefunden werden kann [Richter2003, S. 426].
Ein klarer Vorteil von Fish & Shrink besteht darin, dass verschiedene Retrievalaufgaben durch die Flexibilität des Ähnlichkeitsmaßes berücksichtigt werden können. Außerdem ist das Verfahren effizient, da viele Distanzberechnungen vorab vorgenommen wurden und es als anytime-Algotithmus eingesetzt werden kann. Ein Nachteil sind die komplexen Berechnungen aufgrund der jeweils einzeln betrachteten Fälle. Bei großen Fallbasen und vielen Attributen pro Fall ist der Rechenaufwand sehr groß.

2.5.3 SQL-Approximation

Beim SQL-Approximationsansatz wird ein ähnlichkeitsbasiertes Retrieval durchgeführt indem SQL-Anfragen approximiert werden. Ursprünglich stammt die Idee von [KitanoShimazu1996]. Dafür müssen Falldaten nicht in einer neuen Struktur abgelegt oder ein Index angelegt werden, so dass sich dieser Ansatz für sehr große Fallbasen mit verschiedenartigen Anfragen eignet. Ähnliche Ansätze wurden bereits angewandt, um komplexe Anfragen an relationale Datenbanken zu stellen, die nicht allein von der Datenbank verarbeitet werden konnten. Ein weiterer Vorteil besteht

darin, dass der Ansatz auf beliebige Datenbanken anwendbar und vom Datenbanksystem unabhängig ist [vgl. Bergmann2002].
Es wird angenommen, dass Attribute als Attribute-Werte-Paare vorliegen und die lokalen Ähnlichkeiten durch monotone Ähnlichkeitsmaße beschrieben werden können. Weiterhin setzt der Ansatz voraus, dass die globale Ähnlichkeit ein gewichtetes Mittel ist, dies aber auf andere Amalgamierungsfunktionen erweitert werden kann.

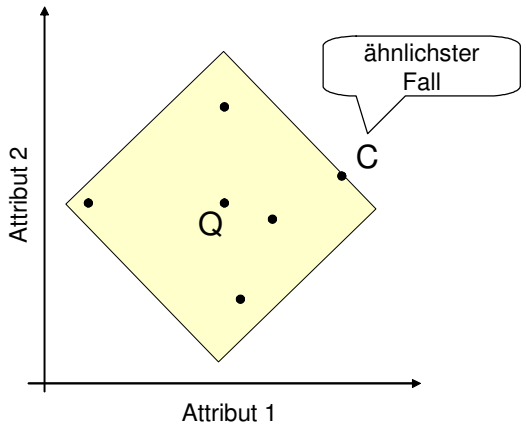


Abbildung 5: Similarity Rhombus, hier dargestellt für n=2 nach [Bergmann2002]

Wie Abbildung 5 zeigt, liegen die ähnlichsten Fälle in einem Hyper-Rhombus in dessen Mitte sich die Query (Q) befindet. Die zur Query ähnlichsten Fälle liegen nah im Zentrum des Rhombus, wogegen weniger ähnliche Fälle weiter entfernt liegen.
Stellt man eine einfache SQL-Anfrage an eine Datenbank in der Form

```
SELECT Attribut1, Attribut2 FROM table
WHERE ((Attribut1-toleranz) AND (Attribut1+toleranz))
AND ((Attribut2-toleranz) AND (Attribut2+toleranz));
```

so liegen die ausgewählten Fälle innerhalb eines Hyperrechteckes. Soll eine bestimmte Menge an Fällen gefunden werden, dann kann durch diese Art der Anfrage eine Lösung aus dem Lösungsraum (Hyperrechteck in Abbildung 6) fallen. Durch das Aufspannen des Hyperrechteckes an den Unter- und Obergrenzen, werden Fälle in den Lösungsraum aufgenommen, die an beiden Grenzen liegen (in den Ecken des Vierecks), wogegen andere Fällen, von denen ein Attribut mit der Query übereinstimmt und ein anderes die Grenze überschreitet, heraus fällt. Wie bereits in Abschnitt 2.5.1 beschrieben, ist der durch die Ähnlichkeit definierte Lösungsraum eine Hyperkugel. Dafür wird in einem zweistufigen Retrieval zuerst eine Vorauswahl durch ein SQL-Retrieval aus einer Datenbank vorgenommen, deren Ergebnis Hyperrechtecke sind. Die beiden Retrievalstufen sind in einer Schleife eingebettet, so dass die Vorauswahl inkrementell erweitert wird, wodurch eine Vermeidung von α-Fehlern möglich ist. Bei einem Lösungsraum in Form eines Hyperrhombus werden die zuletzt beschrieben weiter betrachtet, weil sie der Anfrage räumlich näher und demnach auch ähnlicher sind. Aus diesem Grund wird die Query Relaxation zur SQL-Approximation angewandt.

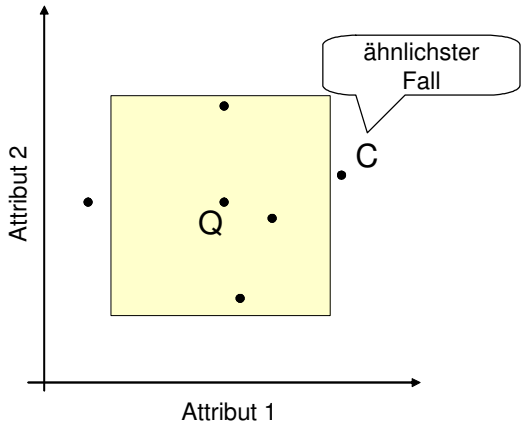


Abbildung 6: SQL-Hyperrechteck, hier dargestellt für n=2 nach [Bergmann2002]

Bei der Query Relaxation werden iterativ Anfragen gestellt, bei denen sich der Raum der betrachteten Fälle inkrementell erweitert, so dass die Form des Hyperrhombus durch die Vergrößerung der Hyperrechtecke approximiert wird (siehe Abbildung 7).

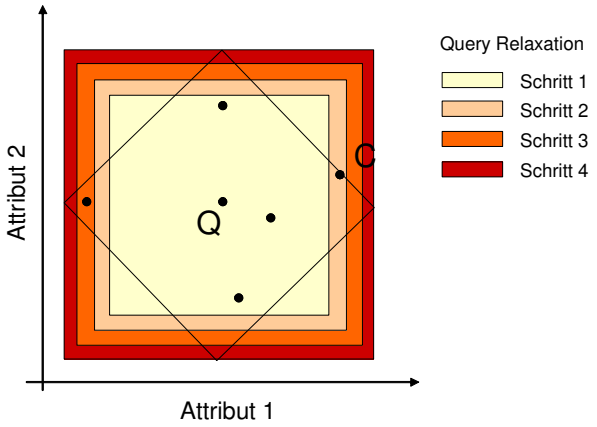


Abbildung 7: Retrieval Ringe und Hyperrhombus nach [Bergmann2002]

Bei der inkrementellen Erweiterung des Hyperrechteckes besteht die Herausforderung darin, dass ein richtiges Maß an Erweiterung gefunden wird. In einem Schritt dürfen nicht zu viele Fälle hinzugenommen werden (also ein zu großer Schritt gemacht werden), aber auch nicht zu wenige (also einen zu kleinen Schritt), da dies eine Verschlechterung der Retrievalperformance zur Folge hat. Ein zu großer Schritt kann zudem dazu führen, dass zu viele Fälle betrachtet werden müssen und ein zu kleiner Schritt, dass eventuell keine neuen Fälle hinzukommen.
Ein Terminierungskriterium kann sein, dass die Anfrage immer eine komplette Liste an Fällen zurückgibt, d.h. mit Sicherheit der ähnlichste Fall in der Lösungsmenge enthalten ist. Es kann aber auch das Similarity Rhombus (siehe Abbildung 5) verwendet werden, da bekannt ist, dass der ähnlichste Fall auf einer Kante des Rhombus liegt. Denn wenn der Rhombus komplett im Rechteck liegt, muss auch der ähnlichste Fall in der Ergebnismenge liegen [vgl. Bergmann2002, S.212f].

Neben dem Vorteil dynamische Fallbasen verarbeiten zu können, kann dieser Ansatz als anytime-Algorithmus eingesetzt werden. Als Nachteil ist aufzuzeigen, dass die Effizienz von den Attributdimensionen abhängen und die Fälle gleichmäßig im Lösungsraum verteilt sind. Die dritte von Bergmann und Schumacher [BergmannSchumacher2000] aufgezeigte Kritik an der SQL-Approximation ist die Begrenzung auf die Attribut-Werte-Darstellung der Fälle, denn jedes Attribut muss auf eine Spalte in einer Tabelle projiziert werden [vgl. Bergmann2002, S.212f].

3 Textuelles Fallbasiertes Schließen in Case Retrieval Netzen

3.1 Grundlegende Begriffe im Textuellen Fallbasierten Schließen mit CRNs

Dieser Abschnitt widmet sich den Grundlagen und Definitionen im TFBS. Die nachfolgenden Definitionen wurden der Diplomarbeit von Mirjam Minor [Kunze1998] entnommen.
Systemintern erfolgt die Darstellung von Fällen und Anfragen durch Informationseinheiten (im nachfolgenden IE genannt). Wie auch in [Bergmann2002] definiert sich ein IE folgendermaßen:

Definition 1: Informationseinheit (IE)

> Eine Informationseinheit ist als atomares Element das kleinste modellierte Element zur Wissensrepräsentation. E ist die Menge aller (möglichen) Informationseinheiten in einer Domäne.

Wie bereits in [Kunze1998] und [Hanft2004] diskutiert, ist der Begriff *Fall* nicht eindeutig definiert, da er sowohl für Beschreibungen (z. B. Problem und Lösung), die unabhängig von dem konkreten fallbasierten System existieren, als auch für deren systeminterne Form innerhalb einer Fallbasis verwendet wird.
Im Weiteren Verlauf dieser Arbeit die systeminterne Form fortan in Abweichung zu [Kunze1998] als *Fallrepräsentation* bezeichnet, während im allgemeinen Kontext *Fall* verwendet wird:

Definition 2: Fall

> Ein Fall ist eine gemachte Erfahrung aus einer bestimmten Domäne in der Form, in welcher der Akteur mit ihr arbeitet.

Definition 3: Fallrepräsentation

> Eine Fallrepräsentation c spiegelt das Wissen eines Falls wider, sie besteht aus einer Menge von IEs: $c \subseteq E$.

Definition 4: Fallbasis

> Die Menge aller Fallrepräsentationen in der Fallbasis wird mit C bezeichnet, wobei gilt: $C \subseteq \wp(E)$.

Analog verhält es sich für den Begriff *Anfrage* (engl. Query), welcher auf der einen Seite die Problembeschreibung, wie ein Benutzer des Systems sie formuliert, und auf der anderen Seite die systeminterne Darstellung dieser Anfrage bezeichnet. Daher wird zur Unterscheidung die zweite Bedeutung als Anfragerepräsentation bezeichnet [vgl. Hanft2004, S.13].

Definition 5: Anfrage

> Eine Anfrage ist eine Problemstellung an ein fallbasiertes System, in der Form wie sie der Benutzer definiert.

Allerdings kann die Form der Anfrage bereits durch die Struktur der Benutzeroberfläche vorgegeben sein. So kann beispielsweise die Eingabe über eine Weboberfläche dazu beitragen, dass bestimmte Attribute nur vorgegebene Werte annehmen dürfen, wogegen für andere Attribute weitere Texteingaben akzeptiert werden [vgl. Hanft2004, S.13f].

Definition 6: Anfragerepräsentation

> Eine Anfragerepräsentation *q* spiegelt das Wissen einer Anfrage wider, das zur Spezifizierung der Problemstellung an ein fallbasiertes System (vom Nutzer) angegeben wird und zu dessen Lösung verwendet werden kann. Sie besteht aus einer Menge von IEs: $q \subseteq E$. Die Menge aller Anfragerepräsentationen über der Fallbasis wird mit Q bezeichnet, wobei gilt: $Q \subseteq \wp(E)$.

Nachdem die Fallbasis angelegt und die Anfrage repräsentiert ist, wird der erste Schritt im CBR-Zyklus nach Aamodt und Plaza [vgl. Abschnitt 2.1], das Retrieval angestoßen. Dazu bedarf es Ähnlichkeitsfunktionen, die z.B. in [Kunze1998] wie folgt definiert sind:

Definition 7: Lokale Ähnlichkeitsfunktion

> Eine lokale Ähnlichkeitsfunktion sim_{loc} bezeichnet den Grad der Ähnlichkeit eines IEs zu einem anderen IE:
>
> $$sim_{loc} : E \times E \rightarrow \Re$$

Definition 8: Globale Ähnlichkeitsfunktion

> Eine Ähnlichkeitsfunktion $sim : Q \times C \rightarrow \Re$ berechnet den Grad der Ähnlichkeit einer Fallrepräsentation zu einer Anfragerepräsentation q, so dass gilt: $sim(q, c_i) > sim(q, c_j) \leftrightarrow c_i \; ist \; \ddot{a}hnlicher \; zu \; q \; als \; c_j$
>
> Sie ist z. B. definiert als:
>
> $$sim(Query, Case) = \sum_{e_k \in Query} \sum_{e_i \in Case} sim(e_k, e_i).$$

3.1.1 Bestandteile des CRN

Das Case Retrieval Netz (CRN, engl. Case Retrieval Net) ist ein gerichteter Graph mit zwei Arten von Knoten und gewichteten Kantentypen. Die Knoten werden in IE-Knoten und Fallknoten (Falldeskriptoren zur Identifikation von Fällen) unterschieden. Jeder Graph besitzt dazu zwei verschiedene Arten gewichteter Kantentypen. Bidirektionale Ähnlichkeitskanten, die je nach Typ (der IEs) die gewichtete Ähnlichkeit zwischen zwei IEs ausdrücken sowie Relevanzkanten von IEs zu den Fallknoten.

Letztere bestimmen die Relevanz einer IE für einen Fall, wie es Abbildung 8 zeigt. Die Propagierungsfunktionen der IE- bzw. Fallknoten lauten π_{IE_i} bzw. π_{C_i} .

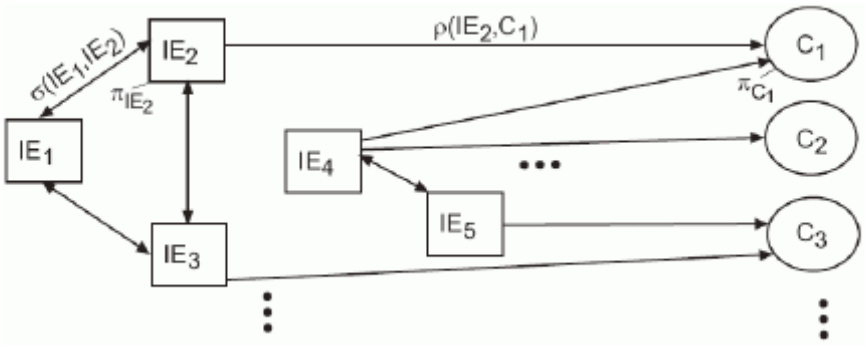


Abbildung 8: Case Retrieval Netz [vgl. Bergmann2002, S. 207]

Eine Fallrepräsentation setzt sich im CRN aus einem Fallknoten und all jenen IE-Knoten zusammen, die den Fallknoten darstellen. Demnach ist eine Fallrepräsentation ein Teilgraph im CRN.
Daraus resultierend ergibt sich folgende Definition des CRN [vgl. Kunze1998, S.7]:

Definition 9: Case Retrieval Netz

Ein Case Retrieval Netz (CRN) ist definiert als Netz $N = [E, C, \sigma, \rho, \Pi]$ mit

- E als endlicher Knotenmenge von Informationseinheiten (IEs),
- C als endlicher Knotenmenge von Falldeskriptoren,
- σ als Ähnlichkeitsfunktion $\sigma : E \times E \rightarrow \Re$,
  welche die Ähnlichkeit $\sigma(e_i, e_k)$ zwischen den IEs e_i und e_k ausdrückt
- ρ als Relevanzfunktion $\rho : E \times C \rightarrow \Re$,
  welche die Relevanz $\rho(e, c)$ zwischen IE e und Fall c ausdrückt und
- Π als endlicher Menge von Propagierungsfunktionen π_n für jeden Knoten $n \in E \cup C \; mit \; \pi_n : \Re^E \rightarrow \Re$.

Die Ähnlichkeiten zwischen den IEs als auch die IEs selbst reflektieren dabei die Eigenschaften der Domäne [Richter2003, S. 419]. Die Bestimmung der Ähnlichkeit wird in Abschnitt 3.2.4 vorgestellt.

3.1.2 Indexierung im CRN

Wie in [Kunze1998] beschrieben, werden zum Aufbau einer Fallbasis die Fallrepräsentationen auf Mengen von IEs abgebildet. Damit die gegebenen Texte im CRN dargestellt werden können, müssen sie durch Indexterme repräsentiert werden.

Die nachfolgenden Definitionen sind ebenfalls der Diplomarbeit von Mirjam Minor [Kunze1998] entnommen.

Definition 10: Indexterm

Ein Indexterm ist ein Begriff, der eine Menge von Strings repräsentiert, die dasselbe semantische Konzept beschreiben, z. B. grammatikalische Formen, Abkürzungen oder verschiedene Schreibweisen.

Definition 11: Indexvokabular

Das Indexvokabular eines fallbasierten Systems für Texte ist die Menge aller bekannten Strings, die auf Indexterme abgebildet werden können.

Im textuellen fallbasierten Schließen werden weiterhin für die Repräsentation der Texte folgende zwei Formen von IEs unterschieden:

Definition 12: Text-IE

Ein Text-IE ist ein IE, das einen Indexterm repräsentiert.

Definition 13: Attribut-IE

Ein Attribut-IE ist ein IE, das ein Attribut-Werte-Paar repräsentiert.

Jedem Index-Term ist genau ein IE zugeordnet und die Text-IEs sind in verschiedene Kategorien (z. B. „domänenspezifischer Begriff oder „allgemeiner Begriff“) unterteilt:

Definition 14: IE-Kategorie

Eine IE-Kategorie ist eine Markierung eines Text-IEs, welche die semantische Zugehörigkeit des IEs zu einer Klasse von Begriffen ausdrückt.

Definition 15: Indexlexikon

Ein Indexlexikon enthält die Menge aller Indexterme, die eindeutigen Abbildungsvorschriften des Indexvokabulars auf Indexterme und die eindeutige Zuordnung jedes Indexterms zu einer IE-Kategorie.

3.1.3 Datenstruktur des Indexlexikons

Wie in Abschnitt 3.1.2 aufgeführt, besteht das Indexlexikon aus mehreren Komponenten, damit das Indexvokabular auf die Indexterme und ihre zugehörigen Klassen abgebildet werden kann. Das nachfolgende Datenbankschema beschreibt den Aufbau des Indexlexikons [vgl. Abbildung 9, S. 27].

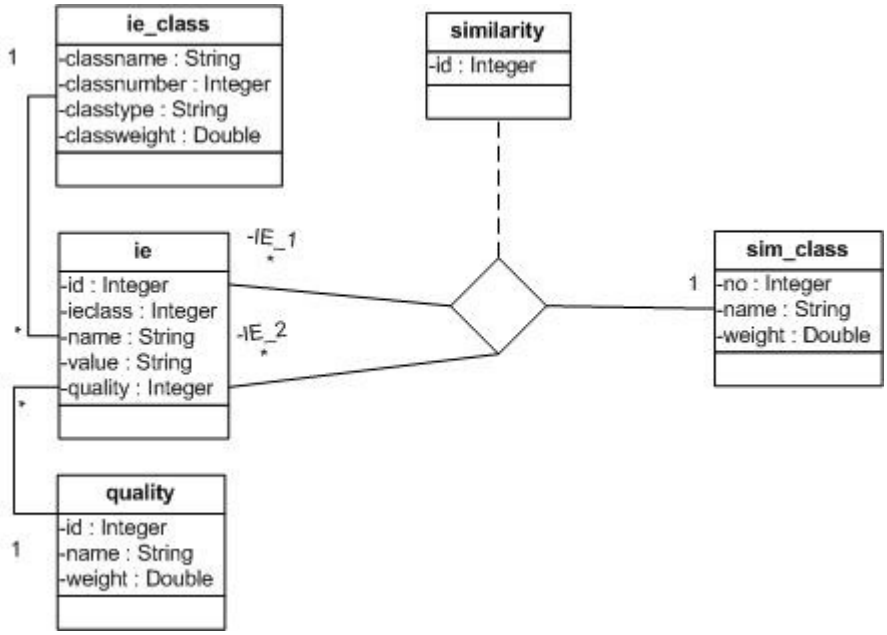


Abbildung 9: Datenmodell des Indexlexikons

Die Struktur des Indexlexikon wurde in Anlehnung an die Diplomarbeit von Hilbig [vgl. Hilbig2003] erstellt und besteht aus vier Klassen und einer Assoziationsklasse. Die zentrale Klasse *ie* besitzt die Attribute:

id für die eindeutige Identifizierung der IE und dient als Primärschlüssel

ieclass als Referenz auf die dazugehörige IE-Klassen (Klasse: ieclass)

name als Bezeichner für die IE

value als Ausprägung der IE

quality als Referenz auf die dazugehörige Klasse zu Bestimmung der Qualität des Attributs

Jedes *IE* kann eindeutig einer *IE-Klasse* zugeordnet werden, wogegen es mehrere IEs geben kann, die einer Klasse angehören. Ebenso verhält sich die Assoziation zwischen *ie* und *quality*. Das Attribut *quality* der Klasse *ie* referenziert auf genau eine *quality*-Klasse wenngleich beliebig viele *ie*-Objekte ein und dieselbe q*uality*-Klasse haben können. Die semantische Bedeutung der beiden Klassen sind in den Abschnitten 5.5.1.1 und 9.1 (IE-Klassen) sowie Abschnitt 5.5.4 (Qualität) beschrieben.

Neben der Erfassung der IEs im Indexlexikon beschreibt das Modell die tenäre Beziehung zwischen der Ähnlichkeit zweier IEs und deren Ähnlichkeitsklasse. Dafür besitzt die Assoziation die Rollen *IE_1* und *IE_2*, die einen Wert des Attributs *value*

aus der Klasse *ie* annehmen können. Jedes IE-Paar besitzt eine eindeutige Identifizierung durch die *id* der Assoziationsklasse *similarity* und die genaue Zuordnung zu einer Ähnlichkeitsklasse (*sim_class*), die die Gewichtung der Beziehung zwischen den beiden IEs beinhaltet.

3.2 Aufbau des Case Retrieval Netzes

Zum Aufbau des CRN werden alle für den Retrieval-Prozess relevanten Informationen auf IEs abgebildet.
Wie bereits in Abschnitt 3.1.2 erläutert werden im CRN IEs in Text-IEs und Attribut-IEs unterschieden. Ein Text-IE stellt einen Abschnitt in einem Satz oder einer Texteinheit dar. Da die Text-IEs aus natürlichsprachlichen Formulierungen extrahiert werden ist es wichtig, den Text in repräsentative Sinneinheiten zu zerteilen, um alle relevanten Informationen herauszufiltern. Attribut-IEs werden explizit als Attribut-Werte-Paar (engl. Attribute-Value Pair, AVP) erfasst und über die Menge aller bekannten Attribut-Werte-Paare auf Attribut-IEs abgebildet.
Die Fälle werden im CRN durch einen Fallknoten und einer Menge von IE-Knoten repräsentiert [vgl. Abschnitt 3.1.1 für die Bestandteile des CRN].
Weiterhin kann es im CRN einen Fallbezeichner, dem Titel des Falles geben, der eine Beschreibung darstellt. Somit besteht ein Fall innerhalb einer Fallbasis aus einem Fallbezeichner, einer oder mehreren Mengen von Text-IEs und einer Menge von Attribut-IEs.

3.2.1 Vokabular

Die Grundlage für die Erkennung von Informationseinheiten und Strukturen eines Textes ist das vorhandene Vokabular, denn es beschreibt die so genannte *Sprache*, die in der Domäne verstanden und genutzt werden kann. Das Vokabular an sich beinhaltet keinerlei Wissen, da es sich um eine Sammlung von Worten handelt [vgl. Bergmann2002].
Somit ergibt sich laut Bergmann folgende Definition des Vokabulars:

Definition 16: Vokabular

> Das Vokabular definiert die Informationseinheiten und Strukturen, die genutzt werden, um Erfahrungen und wieder verwendbares Wissen zu repräsentieren [Bergmann2002].

Die Semantik entsteht erst durch die Benutzung des Vokabulars und dessen Verwendung zur Bestimmung von Ähnlichkeiten und Findung von Lösungen.

3.2.2 Extraktion von Fallinformationen: AVP

Im strukturellen und dialogbasierten FBS (vgl. Abschnitt 2.4) sind die Fälle stark strukturiert und bestehen ausschließlich aus einer Menge von Attribut-Werte-Tupeln.

Im TFBS sind Attribut-Werte-Tupel eine Art der IE und werden wie folgt definiert [Hilbig2003]:

Definition 17: Attribut-Werte-Tupel

Ein Attribut-Werte-Tupel $e_{[a,v_a]}$ besteht aus einem Attribut-Werte-Namen a, der ein Element aus der Menge *AV Name*, die alle möglichen Zeichenketten enthält, ist, und einem Wert v_a aus dem Definitionsbereich Dom_a: $e_{[a,v_a]} \in \{a\} \times Dom_a$

Der Definitionsbereich Dom_a kann grundsätzlich beliebig gewählt werden und jedes Werte-Tupel kann eine beliebige Anzahl von Elementen besitzen. Die Elemente eines Werte-Tupels können aus unterschiedlichen Definitionsbereichen stammen. Dabei müssen die Definitionsbereiche den Datentypen, die die Datenbank bereitstellt, entsprechen, damit die Tupel gespeichert werden können.
Für die Abbildung von Informationen auf AVPs muss vorab bekannt sein welche Informationseinheiten potentielle AVPs enthalten. Weiterhin müssen die Ausprägungen des Attributs vorab definiert werden, damit die Berechnung von Ähnlichkeitswerten zwischen zwei AVPs möglich ist. Die Ausprägungen der AVPs können beliebig modelliert werden und verschiedene Datentypen annehmen. Für die Bestimmung der Ähnlichkeiten zwischen IEs muss für jeden Typ AVP eine Berechnungsfunktion hinterlegt sein [vgl. Hilbig2003, S.14f]. Die Berechnung der Ähnlichkeiten wird in Abschnitt 3.2.4 vorgestellt.

3.2.3 Extraktion von Fallinformationen: Text-IE

Die Ausgangssituation für die Extraktion von Fallinformationen ist ein Textabschnitt in dem IEs und AVPs enthalten sind. Die Grundlage für die Extraktion bildet das Fallformat, denn es gibt an, wie die Informationen abgelegt sind. Es unterscheidet zwischen Retrieval- und Informationsattributen. Retrievalattribute werden beim Aufbau des CRNs berücksichtigt, d.h. potentielle IEs müssen aus ihnen extrahiert werden. Informationsattribute dagegen dienen der Unterstützung des Falles und können dem Nutzer als Ergebnis neben den Retrievalattributen als Lösung präsentiert werden. Im Gegensatz zu Retrievalattributen werden Informationsattribute nicht zum Retrieval herangezogen.
Damit IEs aus einem Text extrahiert werden können, muss dieser vorab geparst werden. Dabei geht das System folgendermaßen vor:

1. Der Text wird Wort für Wort eingelesen
2. Suche nach der längsten Wortfolge, die auf einen String des Indexvokabulars passt.
3. Der gefundene Indexterm wird als IE zu diesem Fall aufgenommen und stellt im CRN die zusammengehörige Wortfolge dar.

Bei der Suche nach einem adäquaten Indexterm wird nach [Rijsbergen1979] folgendermaßen vorgegangen:

1. Alle Stoppwörter herausnehmen
2. Anhand einer rudimentären Grammatik zusammengehörige Terme zusammenführen.
3. Nachschlagen der möglichen Indexterme in einer Liste mit vorhandenen, bereits verwendeten Indextermen.

Es ist das Ziel, so viele Worte wie möglich zu erkennen, daher hat sich die Technik bewährt als Indexterme Worte in ihrer grammatikalischen Grundformen zu verwenden und ihre Flexionen als Ausprägungen anzuführen. Dadurch werden alle Wortformen als „semantisch gleichwertig“ betrachtet.
Darüber hinaus kann die Liste potentieller Indexterme durch Worte erweitert werden, die häufig in Anfragen vorkommen um somit können die Wörterbücher einer Domäne zu erweitern.
Eine Herausforderung besteht jedoch darin, dass ein initial möglichst umfassendes Wörterbuch vorhanden ist, um möglichst viele Fälle auf IEs im CRN abzubilden. Dazu bedient sich das System linguistischer Methoden, die große Mengen von Texten indexieren und eine Liste von Indextermen extrahieren [Kunze 1998, S.9f].
Dabei dienen beispielsweise Part-of-Speech Tagger dazu, ein Indexvokabular bestehend aus klassifizierten Verben, Adjektiven, Adverbien und Substantiven zu erstellen oder bei der Grundformenreduktion zu helfen [vgl. Minor2006, S. 108ff]. Im Deutschen ist die Grundformenreduktion jedoch deutlich schwieriger, da bei Beugungen oftmals der Wortstamm geändert wird [vgl. Minor2006, S. 73]. Aus diesem Grund besteht das in der vorliegenden Anwendung vorhandene Indexvokabular nicht nur aus Grundformen sondern auch aus ihren Beugungen [vgl. 5.5.2].

3.2.4 Bestimmung der Ähnlichkeiten

Wie bereits in Abschnitt 3.1 beschrieben gibt es im CRN lokale und globale Ähnlichkeitsmaße. Attribute und lokale Ähnlichkeitsmaße reflektieren die Eigenschaften einer Domäne. Die Werte der lokalen Ähnlichkeitsmaße sind vom Problem unabhängig, wogegen die der globalen Ähnlichkeitsmaße die Wichtigkeit eines IEs für die Anfrage repräsentieren.
Im CRN werden die lokalen Ähnlichkeiten zwischen Indextermen durch Ähnlichkeitskanten zwischen IE-Knoten dargestellt. Die globale Ähnlichkeit wird durch einen *spreading activation* Mechanismus aufgrund der Aktivierung der entsprechenden Falldeskriptoren berechnet. Welcher Falldeskriptor aktiviert wird, legt die Anfrage fest [vgl. Kunze1998 S.18ff].
Für die Berechnung der Ähnlichkeit, die zwischen zwei AVPs besteht ist die Modellierung der Ähnlichkeiten zwischen ihren Ausprägungen notwendig. Bestehen die AVPs aus Wertepaaren mit lediglich einem Tupel, so stellt der Attributname mit dem zugehörigen Wert das IE dar und der Ähnlichkeitswert zwischen den IEs wird anhand der Ähnlichkeitsfunktion σ festgelegt. Bei größeren Definitionsbereichen,

wie beispielsweise Preis- oder Zeitangaben soll nicht für jedes Element ein Indexterm erstellt werden. Aus diesem Grund muss für solche AVPs eine Berechnungsknotenfunktion existieren, die zwischen einem Attribut-Werte-Tupel der Anfrage und einem Attribut-Werte-Tupel eines Falles einen Ähnlichkeitswert berechnet. Dabei müssen die beiden Attribut-Werte-Tupel den gleichen Attribut-Werte-Namen besitzen [vgl. Hilbig2003, S.14f].
[Hilbig2003] definiert und erläutert die Berechnungsknotenfunktion wie folgt:

Definition 18: Berechnungsknotenfunktion:

Sei Dom_a der Definitionsbereich des Attribut-Werte-Namens $a \in AV\ Name$. Dann ist die Berechnungsknotenfunktion σ_a definiert als

$$\sigma_a : Dom_a \times Dom_a \to \Re .$$

Sei $AV\ Name^*$ die Menge aller Attribut-Werte-Namen, die eine Berechnungsknotenfunktion besitzen. Dabei gilt $AV\ Name^* \subseteq AV\ Name$. Diese Unterscheidung ist notwendig, da auch Attribut-Werte-Namen existieren können, bei denen ein Ähnlichkeitswert zwischen zwei Attribut-Werte-Tupeln durch σ berechnet wird. Die Menge E_a, mit $a \in AV\ Name^*$, ist festgelegt als $E_a = \{ e_{[a,v_a]} \mid v_a \in Dom_a \}$. Sie enthält alle möglichen Attribut-Werte-Tupel des Attribut-Werte-Namens a. E_A sei definiert durch $E_A = \bigcup_{a \in AVName^*} E_A$.

Die Auswirkungen auf die Berechnung der lokalen Propagierungsfunktion und die veränderte Berechnung von σ_a von ausführlich in [Hilbig2003 und LenzBurkhard1996] erläutert.
Die Berechnung der Ähnlichkeitsfunktion erfolgt gemäß der Definition der lokalen Ähnlichkeitsfunktion (Definition 7, S. 23), indem die Wörter, die in Texten vorkommen auf die Indexterme abgebildet werden. Die Ähnlichkeitswerte zwischen den IEs werden entsprechend der Ähnlichkeitsfunktion zwischen den Indextermen bestimmt:

$\sigma : I \times I \to \Re$, wobei *I* die Menge der Indexterme darstellt.

In den meisten Anwendungen werden statt der Ähnlichkeitsfunktion σ *Ähnlichkeitsklassen* verwendet, die jeweils einen Ähnlichkeitswert beinhalten. Die Ähnlichkeitsklasse wird zwischen Paaren von Indextermen festgelegt und somit wird sie genutzt, um σ auf $\Re$ abzubilden.

3.3 Retrieval im Case Retrieval Net

Das Retrieval ist der erste Schritt im 4-R-Zyklus nach Aamodt und Plaza (vgl. Abschnitt 2.1) und dient der Fallauswahl auf Grundlage der Ähnlichkeit. Demnach muss die Bestimmung der Ähnlichkeit immer auf einen speziellen Aspekt oder Zweck bezogen werden. Der Ausgangspunkt des Retrievals ist die Problemstellung für die eine Lösung gefunden werden soll. Dabei ist es notwendig, dass die Ähnlich-

keit die Nützlichkeit gut approximiert [Richter2003]. Abhängig von den Attributtypen muss ein Ähnlichkeitsmaß gewählt werden, welches zur Domäne passt.
Wie zuvor beschrieben bestehen zwischen den IEs in einem CRN Ähnlichkeiten und beim Stellen einer Anfrage werden jene IEs aktiviert, die entweder in der Anfrage vorkommen oder zu den IEs der Anfrage eine Verbindung haben. Die Aktivierung wird durch das Netz bis hin zu den Fallknoten durchpropagiert, so dass jener Fall mit der höchsten Aktivierung als Lösung zurückgeliefert wird. Die Propagierung erfolgt zweistufig – zuerst die Ähnlichkeiten, danach die Relevanz.

Aktivierung
Eine neue Anfrage Q an ein CRN wird wie ein neuer Fall aufgefasst, dass heißt die Anfrage muss dem Fallformat entsprechen, nach dem die Fälle im CRN aufgefasst werden. Aus der Anfrage werden alle möglichen IEs extrahiert und im CRN jene IEs aktiviert, die in der Anfrage vorkommen. Die Bestimmung der Aktivierung α_0 erfolgt durch:

$$\alpha_0(e) = \begin{cases} 1 & falls\ IE\ e\ in\ der\ Query\ vorkommt \\ 0 & sonst \end{cases}$$

e ist hierbei die ein IE aus der Menge der IEs, die das CRN aufbauen.
Die aktivierten IEs im CRN spiegeln demnach die Fallbeschreibung der Anfrage wieder.

Ähnlichkeitspropagierung
Die Ähnlichkeitspropagierung α_1 erfolgt für jeden IE-Knoten, der eine Ähnlichkeitskante zu einem aktivierten IE hat und wird folgendermaßen bestimmt:

$$\alpha_1(e) = \pi_e\ (\sigma(e_1, e) * \alpha_0(e_1),\ \ldots,\ \sigma(e_S, e) * \alpha_0(e_S))$$

Die IEs e_1 bis e_S sind die aktivierten IEs aus dem gegebenen CRN.

Relevanzpropagierung
Für die Relevanzpropagierung werden die Relevanzfunktionen ρ, für alle Fallknoten zu denen eine Relevanzkante von einem aktivierten IE führt, berechnet:

$$\alpha_2(c) = \pi_c\ (\rho(e_1, e) * \alpha_1(e_1),\ \ldots,\ \rho(e_S, e) * \alpha_1(e_S))$$

Das Ergebnis der Aktivierung und Propagierung ist eine nach Relevanz geordnete Liste von Fällen, die der Query am ähnlichsten sind. Die zurück gelieferten Fälle werden wie in Abschnitt 2.1 beschrieben weiterverarbeitet und als Ergebnis der Anfrage für die nachfolgenden Schritte bereitgestellt.

4 Domänenmodellierung

Dieser Abschnitt behandelt die verschiedenen Vorgehensweisen bei der Domänenmodellierung in verwandten Gebieten und zeigt anhand von Beispielen die anzuwendenden Schritte für eine erfolgreiche Domänenmodellierung.
Bei der Entwicklung von Anwendungssystemen ist es zum einen wichtig den Einsatzbereich der Anwendung zu kennen, um es in die vorhandenen Arbeitsprozesse zu integrieren und zum anderen Wissen über die Domäne bereitzuhalten, dass über den Inhalt der Fälle hinausgeht.
In der Literatur ist der Begriff Domäne teilweise sehr weit gefasst, so dass er entweder zur Beschreibung der Anwendungsbereiche (application domain) selbst oder zur Zusammenfassung der Anwendungsbereiche und der dafür entwickelten Softwaresysteme genutzt wird.
In [PietroDíaz1990] wird die Domäne als Anwendungsgebiet für die ein Softwaresystem entwickelt wurde definiert, wogegen die Definition in [Berard1993] darüber hinaus die Domäne als Sammlung von aktuellen und zukünftigen Softwaresystemen, die gemeinsame Charakteristiken haben, zur Beschreibung einer Domäne zählt. Weiterhin kann laut [Berard1993] eine Domäne eine begrenzte Menge von Charakteristiken sein, die eine Gruppe von Problemen, die von Anwendungssoftware gelöst werden soll, exakt und vollständig beschreibt.
Basierend auf diesen Definitionen wird in [Krabbel2000] geschlussfolgert, dass der Begriff Domäne sehr unterschiedlich angewandt wird und nur wenige Kriterien aufgezeigt werden, die die Eigenschaften einer Domäne bestimmen. Konkretere Eigenschaften als

- Domänen sind ein abgetrenntes, kohäsives Ganzes,
- Domänen sind personenabhängig,
- Domänen sind vertikal oder horizontal,
- Domänen bauen aufeinander auf,

wurden nicht genannt [Kabbel2000].
Ähnlich verschiedene Ansätze gibt es bei der Definition der Domänenmodellierung, denn je nach Auffassung des Domänenbegriffs beschreibt das Domänenmodell

- die Problemklasse im Anwendungsbereich in Form von Konzepten und Taxonomien, die dann bis hin zur Spezifikation von Softwaresystemen in der Domäne verwendet werden [Arango1994]. Oder
- die Abstraktionen und Klassifizierung von Gemeinsamkeiten, die bei den untersuchten Softwaresystemen in dieser Domäne festgestellt wurden.

Aufgrund der verschiedenen Auffassungen der Begriffe Domäne und Domänenmodellierung soll in den nachfolgenden Abschnitten ein kurzer Überblick über Domänenmodellierungen in verwandten Disziplinen gegeben werden.

4.1 Anwendungsbereiche der Domänenmodellierung

4.1.1 Expertensysteme

Als Expertensysteme werden Anwendungen im Bereich der Klassifikation, Diagnose, Konstruktion und Planung bezeichnet, die zur Lösung praktischer Aufgaben entwickelt wurden. Dabei sollten die Gedankengänge und Erfahrungen von menschlichen Experten formalisiert werden, um die Art der Problemlösung zu reproduzieren. Die Entwicklung solcher Systeme begann Mitte der 1970er Jahre und stellt somit die Grundlage aller Weiterentwicklungen von wissensbasierten Systemen dar.
Die Erfahrung von Experten wird auch als Expertenwissen bezeichnet, welches im System repräsentiert werden muss. Für die Darstellung des Expertenwissens musste der Anwendungsbereich erfasst und formalisiert werden. Eine Wissensbasis im Expertensystem besteht aus bereichsbezogenem Wissen, fallspezifischem Wissen sowie Zwischen- und Endergebnissen aus der Problemlösungskomponente [Steinmüller2005].
Die Anwendungsbereiche für Expertensysteme sind nicht klar definiert und abgegrenzt und das darin enthaltene Wissen ist meist unsicher, unvollständig und zeitabhängig, so dass die Formalisierung nicht trivial ist und Methoden entwickelt wurden, das Wissen abzubilden. Die dafür entwickelten Methoden werden heute weiterhin wissensbasierten Systemen eingesetzt [Steinmüller2005].
Da Expertensysteme für konkrete Aufgaben entwickelt wurden, musste der Aufgabenbereich beschrieben werden, was heute die Aufgabe der Domänenmodellierung ist.

4.1.2 Software Engineering

Im Software Engineering, genauer im Requirement Engineering, ist die Domänenmodellierung der erste Teil der Objektorientierten Analyse (OOA) in dem die Anforderungen der Nutzer eines Systems erfasst und beschrieben werden. Das Ergebnis der Domänenmodellierung ist ein textuelles Pflichtenheft mit allen Fakten, die im zu implementierenden System berücksichtigt werden müssen. Auf dem Pflichtenheft basiert der zweite Teil der Objektorientierten Analyse, das Design, in dem die fachlichen Zusammenhänge in objektorientierten Konzepten beschrieben werden. [vgl. CoadYourdon1994].
In [EisenecknerCzarnecki2000] wird der Begriff Domäne als Wissens- und Anwendungsbereich beschrieben, der von Stakeholdern festgelegt und folgendermaßen charakterisiert werden kann:

- Sammlung von Konzepten und Begrifflichkeiten, die von Anwendern des Fachgebiets interpretiert werden
- Eingrenzen der Sichtweise bezüglich der optimalen Abdeckung der Anforderungen, welche durch die Interessenvertreter entwickelt werden
- Wissen für die komponentenbasierte Systementwicklung

In der Produktlinienentwicklung ist die Analyse und Modellierung einer Domäne ein zentraler Baustein des Domänenengineering, welches sich in Domänenanalyse, Ar-

chitekturentwicklung und Domänenimplementierung unterteilt. Die Domänenanalyse wurde von Arango als “Prozess zur Identifikation, Akquisition und Evaluation von wieder verwendbaren Informationen, die im Zuge der Entwicklung von Systemen für Klassen von Applikationen oder Problemdomänen wieder verwendbar sind“ beschrieben [vgl. Arango1994]. Die Domäne wird in diesem Zusammenhang als eine Menge von Konzepten aus einem Wissensbereich mit domänenspezifischen Problemen, Lösungen und Begriffen. Aus diesem Grund muss es das Ziel der Domänenanalyse sein, dass ein Modell erstellt wird, welches den Anwendungsbereich des Systems so ausgiebig wie möglich in seinen Wissenscontainern widerspiegelt. Kang et al. schlägt folgende Phasen zur Erstellung eines Domänenmodells vor [Kang1990]:

1. Kontextanalyse:
   Die Kontextanalyse beschreibt die Grenzen und den Umfang einer Domäne sowie die Umgebung, in der sich die zu beschreibende Domäne befindet. Dazu gehören im Weiteren die Kommunikationsschnittstellen zum Datenaustausch, Beschränkungen und Informationsflüsse.
2. Domänenmodellierung:
   Die Domänenmodellierung bildet die Kernphase, denn die Beschaffenheit der Domäne wird untersucht und ihre Struktur beschrieben, so dass als Ergebnis dieser Phase ein Modell entsteht, welches als Grundlage für die Strukturierung eines Systems dient.
3. Architekturmodellierung:
   Im Rahmen der Architekturmodellierung werden Lösungen entsprechend des in Phase zwei entwickelten Domänenmodells entworfen.

Im Gegensatz zu der in dieser Arbeit vorgestellten Domänenmodellierung beschränkt sich die Domänenmodellierung im Software Engineering auf den Entwurf der Software. Dennoch bestehen Gemeinsamkeiten bei der Analyse der vorliegenden Dokumente und dem Ziel, ein Modell zu schaffen, welches die Struktur der Daten beschreibt und somit maschinell verarbeitbar macht.

4.1.3 Die Bedeutung der Domäne bei der Entwicklung von Multiagentensystemen

Agenten sind Computersysteme, die dazu fähig sind eigenständige Handlungen im Sinn ihres Benutzers durchzuführen [Althoff2006]. Sie können Aufgaben selbstständig ausführen, reaktiv und proaktiv auf veränderte Umwelteinflüsse handeln, interaktiv kommunizieren und sind hinreichend mobil. Es gibt verschiedene Agententypen, wie beispielsweise in [Schiemann2007] aufgezeigt:

- Reiz-Reaktions-Agent (Subsumptionsarchitektur)
- Rationaler Agent (korrekt, Leistungsfunktionsmaximierung)
- Modellbasierter Agent (inneres Abbild der hypothetischen Welt)
- Zielbasierter Agent (abstrahiert von einfachen Regeln zur Strategie)
- Lernender Agent (lernt, bewertet welche Aktion am günstigsten ist)
- Logische Agenten (Wissensbasis, Inferenzfähigkeit)

Multiagentensysteme bestehen aus mehrere Agenten, die untereinander kommunizieren. Dafür ist es notwendig, dass es Schnittstellen gibt, zwischen denen der Informationsaustausch unter den Agenten vorgenommen werden kann. Sie müssen demnach zusammenarbeiten, sich aufeinander abstimmen und miteinander verhandeln können [Althoff2006].
Aus den oben aufgezeigten Agententypen benötigt der modellbasierte Agent ein Abbild der hypothetischen Welt, welches als Wissensbasis bestehend aus einem internen Domänenmodell, das den Anwendungsbereich wiedergibt, den Zielen des Agenten sowie der chronologischen Entwicklung der Umgebung (das so genannte Wissensmodell) aufgefasst wird [vgl. BiundoStephan2006].
Das Domänenmodell besteht aus:

- Konzepten und Konzepthierarchien
- Objekten, Eigenschaften, Zusammenhängen
- Aktionen, Informationen über Ausführbarkeit und Wirkung
- Gesetzmäßigkeiten

Die Konstruktion der Wissensbasis beginnt mit dem Entwurf des Anwendungsbereiches in dem der Agent arbeiten wird. Anhand der daraus resultierenden Anforderungen wird ein Repräsentationsformalismus festgelegt und Inferenzmechanismen werden ausgewählt und bereitgestellt. Als nächstes wird Wissen zum festgelegten Anwendungsbereich gesammelt und in Ontologien und/oder Regeln abgebildet um es dem Agenten zur Verfügung zu stellen [vgl. BiundoStephan2006].
Die Abbildung der Domäne ist demnach ein zentraler Baustein im Aufbau einer Wissensbasis, die dem Agenten ermöglicht Umwelteinflüsse zu erfassen und auf sie zu reagieren.

4.1.4 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) beschreibt einen Prozess der (semi-) automatischen Extraktion von gültigem, bisher unbekanntem und potentiell nützlichem Wissen aus Datenbanken. Es ist eine Schnittstellendisziplin aus Maschinellem Lernen, Datenbanksystemen und Statistik.

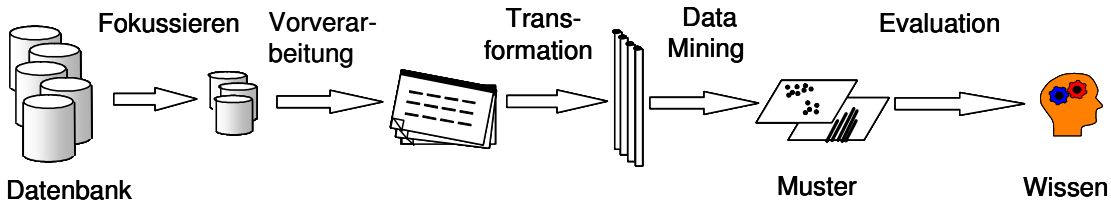


Abbildung 10: KDD nach [FayyadEtAl1996]

Abbildung 10 zeigt das allgemein anerkannte Schema der Wissensextraktion aus Datenbanken nach [FayyadEtAl1996], in dem KDD folgendermaßen definiert wurde:

Definition 19: Knowledge Discovery in Databases nach [FayyadEtAl1996]

> "Wissensentdeckung in Datenbanken ist der nichttriviale Prozess der Identifizierung gültiger, neuartiger, potentiell nützlicher und verständlicher Muster in (großen) Datenbeständen."

Im ersten Schritt muss das Ziel des KDD definiert werden und entsprechend diesem Ziel werden die Daten aus dem Operativsystem beschafft.
Die selektierten Daten werden aus unterschiedlichen Quellen integriert, auf Konsistenz geprüft und gegebenenfalls vervollständigt. Dabei soll vor allem das Hintergrundrauschen minimiert und/oder fehlende Daten zur selektierten Datenmenge hinzugefügt werden. Anschließend werden die Daten reduziert und durch Projektionstechniken werden die in der Regel hochdimensionalen Datensätze in leichter handhabbare, niederdimensionale Daten transformiert. Dabei werden beispielsweise numerische Attribute auf Wertebereiche in Intervalle mit gleicher Länge aufgeteilt, so dass der Informationsgewinn bezüglich der Klassenzugehörigkeit maximiert wird. Weiterhin werden durch die Aggregation von Datensätzen oder durch Verknüpfen mehrerer Attribute neue Attribute erzeugt. Dadurch können andere Attribute überflüssig werden, die manuell oder automatisch von der Weiterverarbeitung ausgeschlossen werden, denn zu viele Attribute führen zu Ineffizienz und evtl. Ineffektivität des Data Mining [vgl. Oellien2002].
Das eigentlich Data Mining beginnt erst nach der Datenaufbereitung mit der Analyse der generierten Datensätze, um in den Daten verborgene Gesetzmäßigkeiten zu extrahieren und diese in Form von Vorhersagen zu nutzen. Data Mining Aufgaben sind unter Anderem Clustering, Klassifikation, Generalisierung oder Extraktion von Assoziationsregeln. Für die richtige Auswahl und mögliche Aggregation der Daten ist es demnach wichtig, in welchem Anwendungsbereich das Data Mining vorgenommen werden soll. Die Bedeutung der Domäne ist zudem essentiell, wenn die erkannten Muster ausgewertet und Wissen aufgebaut werden soll. Zur Klassifikation können wie beispielsweise in YALE[5], einem Data Mining Tool der Universität Dortmund, folgende Techniken genutzt werden (Auszugsweise[6]) [vgl. Mierswa2006]:

- Maschinelles Lernen:
  - Support Vector Machines (LibSVM, SMO, mySVM, ...)
  - Decision Tree and Rule Learners (ID3, C4.5, PART, PRISM, ...)
  - Lazy Learners (Nearest Neighbors, K*, LBR, ...)
  - Bayesian Learners (Naive Bayes, Bayes Net, AODE, ...)
- Datenvorverarbeitung:
  - Discretization (Binning, Frequency, ...)
  - Normalization (Interval, Standardization, z-Transformation, ...)
  - Dimensionality Reduction (PCA, Kernel-PCA, ...)

[5] YALE = **Y**et **A**nother **L**earning **E**nvironment, Website: http://yale.sf.net/ und http://rapid-i.com/
[6] vollständig: www-ai.cs.uni-dortmund.de/SOFTWARE/YALE/intro.html

4.1.5 Domänenmodellierung für Webanwendungen

Bei der Konzeption von Webanwendung ist es wichtig, neben den Zielen der eigentlichen Webanwendung auch das Web-Umfeld zu analysieren um Gemeinsamkeiten und Unterschiede angrenzender Produkte (Anwendungen) zu kennen und zu beachten. Das Domänenmodell soll alle möglichen Informationen strukturieren und bereitstellen, die für die Entwicklung der Anwendung notwendig sind. Darüber hinaus müssen die Randbedingungen, die funktionalen wie auch nichtfunktionalen Abhängigkeiten erfasst werden. Es ist eine explizite Repräsentation der Eigenschaften von Systemen einer Systemfamilie oder Komponenten von Systemen innerhalb einer Domäne [vgl. FZI2001, S.7ff].
Das Ergebnis der Domänenanalyse ist laut [EisenecknerCzarnecki2000] ein Domänenmodell bestehend aus:

- Die ***Domänendefinition*** definiert und verfeinert den Bereich der Domäne und charakterisiert die Begriffe und Systemteile innerhalb der Domäne.
- Das ***Lexikon/Glossar*** definiert das domänenspezifische Vokabular und legt die Bedeutung grundlegender Begriffe fest.
- ***Modelle*** beschreiben die Begriffe der Domäne, ihr Verhalten und ihre Interaktion untereinander in einem Modellierungsformalismus, angereichert durch Text. Die Art und der Umfang der Modelle unterscheiden sich je nach Verfahren.

Webanwendungen basieren größtenteils auf Client-Server-Architekturen, so dass sie aus verschiedenen Sichten analysiert und modelliert werden können [vgl. Kruchten1995]:

- Die ***logische Sicht*** mit den Beschreibungen der Objekten und Begriffen der Anwendung.
- Die ***physikalische Sicht*** mit der Fokussierung auf den Zusammenhang zwischen Software und Hardware.
- Die ***Entwicklungssicht*** beschreibt die Komponenten des Systems und ihre statische Organisation.
- Die ***Prozess-Sicht*** beschreibt das Zusammenwirken der Komponenten.
- Als weitere ***Sicht*** beschreiben ***Szenarien*** die Anwendung des Systems in einem konkreten Kontext.

Die Sichten abstrahieren von den konkreten Problemstellungen und dienen dazu die Architektur für das zu entwickelnde System zu spezifizieren. Bei der Betrachtung der Sichten auf logischer bzw. konzeptioneller Ebene wird zwischen horizontaler und vertikaler Dimension unterschieden.

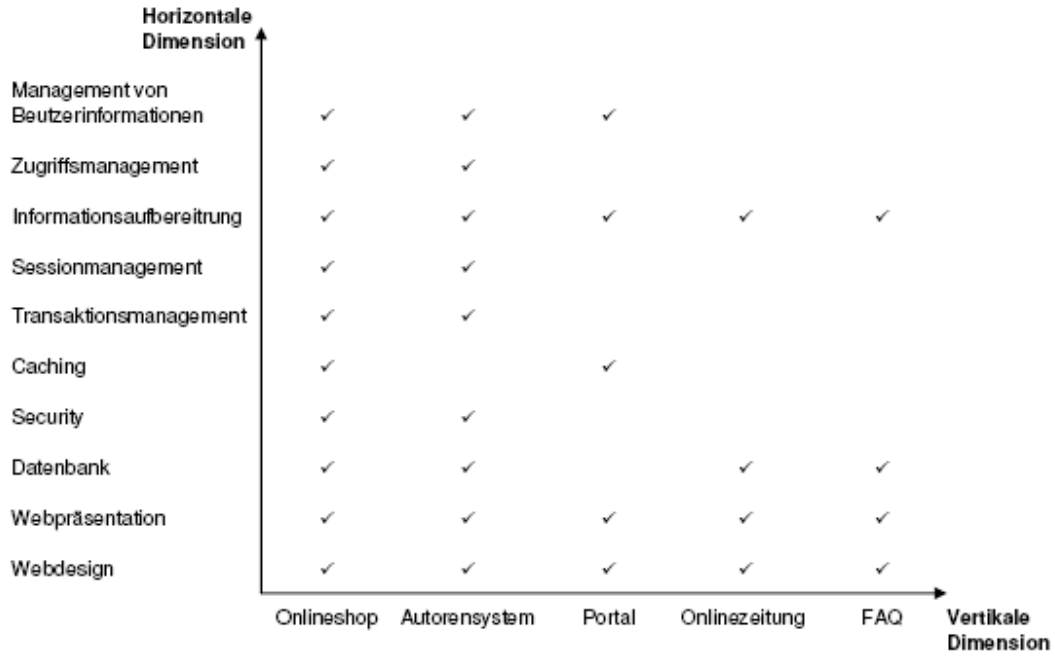


Abbildung 11: Horizontale und vertikale Dimensionen der Domänen nach [FZI2001, S. 15].

Die horizontale Dimension beschreibt die Querschnittstechnologien, die genutzt werden, um die Anwendung zu implementieren wogegen die vertikale Dimension die Produkte und Anwendungen beschreibt. Horizontale und vertikale Dimensionen können getrennt voneinander betrachtet werden, jedoch sind je nach Anwendungen (vertikale Dimension) die horizontalen Dimensionen verschieden priorisiert. In Abbildung 11 wird für Beispielanwendungen (Abszisse) gezeigt, welche Querschnittstechnologie (Ordinate) für deren Umsetzung implementiert werden müssen [vgl. FZI2001, S.15, S. 40ff].
Die Auffassung der Domänenmodellierung für Webanwendung unterscheidet sich von der Domänenmodellierung für das TFBS, da bei zweiterem das Hauptaugenmerk auf die Repräsentation der Informationen in einem Index gelegt wird. Trotzdem spielen die horizontalen Dimensionen ebenfalls eine Rolle, da einige davon bei der Erschließung einer Domäne und Entwicklung einer Anwendung beachtet werden müssen.

4.2 Domänenmodell im FBS

Bevor eine CBR-Anwendung erstellt wird, muss die Domäne erschlossen werden, um das Wissen und die Erfahrung von Experten erfassen und in einem Modell repräsentieren zu können. Dabei ist es wichtig, dass der richtige Abstraktionsgrad des Modells für die Anwendung gefunden wird. In [Berghofer2003] wird hervorgehoben, dass ein Domänenmodell zum einen so speziell sein muss, dass alle Aspekte der Erfahrung abbildbar sein müssen und es zum anderen aber nicht zu speziell sein darf, damit es seine Gültigkeit behält. Ein zu granulares Modell führt zudem dazu, dass zu viele nicht benötigte Details aufgenommen werden, die bei der Wissens-

akquisition wie auch bei der Wartung der Fallbasis einen hohen Aufwand verursachen [Berghofer2003].
Der Aufbau des Domänenmodells hängt zudem von der Art des FBS ab, für welches das Modell erstellt wird. Soll beispielsweise ein strukturelles FBS durchgeführt werden, eignet sich ein objekt-orientierter Ansatz, obwohl der Aufwand das Domänenmodell zu erstellen ungleich höher ist als bei einer Attribut-Wert-Repräsentation.
Der Modellierer muss das Einsatzgebiet des Systems kennen und wissen, bei welchen Aufgaben der Anwender unterstützt werden soll. Für Anwendungen in denen eine begrenzte Anzahl von Attribut-Werte-Paaren zur Repräsentation ausreicht, bedarf es keinem objekt-orientierten Modell, wogegen sie sich bei komplexeren Modellen mit einer großen Menge von Attribut-Werte-Paaren anbieten. Die objekt-orientierte Modellierung ist notwendig, wenn das Domänenmodell die Grundlage für die Ähnlichkeitsberechnung bildet und Abhängigkeiten zwischen den modellierten Objekten besteht. Weiterhin kann es genutzt werden, um den Lösungsraum durch die Anfrage einzuschränken.

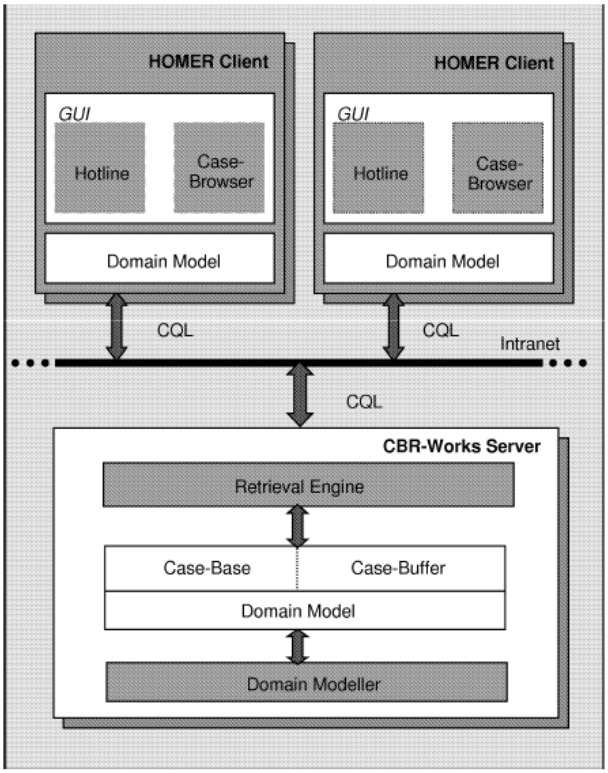


Abbildung 12: System-Architektur vom HOMER [vgl. GökerEtAl1998].

Bei einer Help-Desk-Anwendung wie in [GökerEtAl1998] beschrieben, kann somit vermieden werden, dass mehrere hundert teils redundante Fragen gestellt werden, um zu einer Lösung zu kommen, obwohl einige Antworten auf vorangegangne Fragen den Lösungsbereich soweit eingeschränkt hatten, dass andere Fragen nicht mehr notwendig wären. Um dies zu verhindern muss eine Art Entscheidungsbaum mit den Fragen und deren möglichen Antworten aufgebaut werden. Ein objekt-orientiertes Domänenmodell hat ebenfalls ein komplexeres Fallformat zur Folge und

damit ist die initiale Erstellung der Fallbasis deutlich aufwendiger, als bei Attribut-Wert-Repräsentationen [GökerEtAl1998].
Im Weiteren hat das aus einem Domänenmodell resultierende Fallformat auch Einflüsse auf die Implementierung eines CBR-Systems, da das Fallformat die Datenstruktur vorgibt, die verarbeitet werden muss (vgl. Abbildung 12). Denn die Anfrage muss der vom Fallformat vorgegebenen Struktur entsprechen, denn nur so kann das Retrieval in einer Fallbasis beginnen. So wird aus Abbildung 12 ersichtlich, dass das Domänenmodell für die Retrieval Engine auf dem Server zur Bereitstellung der Fälle wie auch am Client, der die Anfrage zusammensetzt, benötigt wird.

4.3 Einordnung in die Problemstellung dieser Arbeit

Nachdem die vorangegangenen Abschnitte die Bedeutung der Domäne in verwandten Gebieten innerhalb der Informatik sowie im FBS behandelten, soll in diesem Abschnitt Bezug auf die vorliegende Arbeit genommen werden.
Zuerst sollen die beiden grundlegenden Begriffe Domäne und Domänenmodell im Sinne dieser Anwendung definiert werden:

Definition 20: Domäne nach [PietroDíaz1990]

> Eine Domäne ist im allgemeinen Zusammenhang eine „Abgrenzung einer Aktivität oder Betrachtung: ein Bereich“ [Webster] Im Software Engineering wird es meist als Anwendungsgebiet, als Bereich für den eine Software entwickelt wurde, verstanden. Beispiele dafür sind Systeme zur Flugbuchung, Zahlungssysteme, Kommunikations- und Kontrollsysteme, Tabellenkalkulationen oder numerische Steuerungen (bspw. CNC-Systeme). Domänen können sehr weit gefasst werden wie das Bankwesen oder begrenzt als arithmetische Funktion. [vgl. PietroDíaz1990, S.50]

Demnach ergibt sich folgende Definition für ein Domänenmodell:

Definition 21: Domänenmodell nach [Arango1994]

> Ein Domänenmodell ist die Beschreibung eines Anwendungsbereiches in Form von Konzepten und Taxonomien.

Somit soll ein Domänenmodell die Realität so detailgetreu wie möglich abbilden, ohne, dass es zu spezifisch ist und nicht jede Möglichkeit abbilden kann. Bei der Entwicklung eines Domänenmodells muss in der Regel ein Experte des Anwendungsbereiches dem Modellierer die Anwendung näher bringen. Dieser Schritt ist äußerst wichtig für das richtige Arbeiten eines Systems und dessen Akzeptanz bei seinen Benutzern.

Die hier vorliegende Arbeit befasst sich mit der Erschließung neuer Domänen für das Textuelle Fallbasierte Schließen. Dabei wird davon ausgegangen, dass die Fälle im Freitextformat in einer Datenbank vorliegen. Die Ablage der Daten zum Aufbau einer Fallbasis in Datenbanken dient dazu, dass die Vorteile von Datenbankmanagementsystemen ausgenutzt werden können. Es ist einfacher ein Backup

einer Datenbank zu erstellen und die Fallbasis auf einem anderen Server als die Datenbank vorzuhalten, um die Performanz zu steigern. Außerdem können mehrere Nutzer zeitgleich auf die Datenbank zugreifen und die Extraktion der Daten für das Retrieval gestaltet sich einfacher [vgl. Pfuhl2003].
Aus diesem Grund wird in den nachfolgenden Ausführungen auch von der Extraktion aus einer Datenbank gesprochen. Somit passen Definition 20 und Definition 21 in das verwendete Konzept, denn eine Domäne ist ein abgeschlossener Bereich und das Domänenmodell enthält die Konzepte des Bereiches. Die Konzepte wurden dahingehend erstellt, dass anhand des Domänenmodells ein Fallformat modelliert werden kann (vgl. Abschnitt 5.1) und einzelne Fälle in einer Fallbasis repräsentiert werden können (vgl. Abschnitt 5.4).
Neben der Modellierung ist es bei der Erschließung von Texten wichtig, die verwandten Worte zu kennen und repräsentieren zu können. Für die Repräsentation werden beim TFBS durch CRNs die IEs genutzt und für die Worterkennung benötigt man ein weit reichendes Vokabular. Der Aufbau und die Erweiterung des Vokabulars stellt das Kernthema dieser Arbeit dar und wird in Kapitel 5 detailliert erläutert.

5 Domänenmodellierung zum Aufbau eines CRN

5.1 CRN als Informationssystem

Die Grundlage eines jeden Case Retrieval Netzes (CRN) ist die Wissensbasis, aus der Informationen extrahiert werden. Damit die Extraktion der Informationen vorgenommen werden kann, ist die Analyse der Domäne, in der das Informationssystem eingesetzt werden soll, notwendig. Dabei müssen grundlegende Aspekte betrachtet werden, die zu Beginn dieses Kapitels vorgestellt werden. Der zweite Abschnitt zeigt, wie Wertebereiche erfasst und Fälle in die Fallbasis aufgenommen werden. Danach werden Verfahren zur Erweiterung eines bestehenden Vokabulars gezeigt und bewertet. Abschließend wird an einem konkreten Beispiel die Indexierung mit den zuvor erarbeiteten Wissensquellen vorgestellt und es wird gezeigt, wie unbekannte Wort ermittelt und modelliert werden können.

5.1.1 Aspekte der Domänenmodellierung im CRN

Die grundlegende Frage der Domänenmodellierung ist die Zielsetzung der Anwendung und welche Daten zum Aufbau des CRN zur Verfügung stehen. Im Weiteren müssen Methoden gewählt werden, welche ausreichend viele Informationen aus den Daten extrahieren und in IEs umwandeln.
Im Mittelpunkt der Domänenmodellierung steht demnach die Domänenanalyse, die aufzeigen soll, in welchem Umfeld die Anwendung bereitgestellt wird, welche Informationen bereitgestellt werden müssen und welche Prozesse betroffen sind. Ebenso sollte dabei betrachtet werden welche Aufwendungen notwendig sind, um die Nutzung weiter voran zu treiben. Dafür ist es notwendig zu betrachten, welche Nutzer es für diese Anwendung geben wird.

5.1.1.1 Bestandteile einer Domäne

Eine Domäne wird meist als Anwendungsdomäne (engl. application domain) aufgefasst und beschreibt ein abgrenzbares Problemfeld des täglichen Lebens oder eine bestimmten Einsatzbereich für ein Computersystem. Anwendungsdomänen stellen typischerweise sehr spezielle Anforderungen an ein technisches System, welches zur Bewältigung der domänenspezifischen Aufgaben und Probleme eingesetzt werden soll [Krabbel2000].
Für die Beschreibung einer Domäne müssen alle Gruppen der Stakeholder[7] betrachtet werden, damit sichergestellt wird, dass die Domäne adäquat im System abgebildet wird. Des Weiteren müssen die folgenden Aspekte betrachtet werden, um eine Domäne zu beschreiben: Domänenobjekte, Domänenmodell sowie das Geschäftsmodell [Lenz1999, S.94].

[7] deutsch: Interessenvertreter

Domänenobjekte

Domänenobjekte beschreiben die Daten, mit denen eine Anwendung arbeiten soll. Diese Daten liegen im Rohzustand vor und werden im CRN durch die Fälle repräsentiert. Dafür bedarf es einer geeigneten Art der Wissensrepräsentation. Weiterhin muss festgelegt werden, welche Teile der Domänenobjekte die Fälle in der Fallbasis repräsentieren, denn nur diese Teile werden in die Fallrepräsentation aufgenommen und damit während des Retrievals betrachtet [Lenz1999, S.95].
Je nach Vorlage der Daten muss im Fallformat definiert werden, wie auf die Domänenobjekte zugegriffen wird, so dass beispielsweise für sich regelmäßig ändernde Daten eine automatisierte Konvertierungsmöglichkeiten für die Extraktion der Fälle bereitgestellt wird. Auch kann verlangt werden, dass bei der Extraktion der Fälle nicht auf Operativsysteme zugegriffen wird und daher die Falldaten einer Datenbank über *views* extrahiert werden. Daher müssen Informationen über die Domäne, ihre Objekte und wie sich die Änderungen auf die Domänenobjekte auswirken vorliegen [Lenz1999, S.95f].

Domänenmodell

Das Domänenmodell eines CRNs sollte alle möglichen Attribute enthalten, die auch in den Domänenobjekten vorhanden sind, denn es dient dazu, die Ähnlichkeiten zwischen Attributen zu beschreiben.
Zum Aufbau des Domänenmodells können folgende Quellen Betracht in gezogen werden:

1. Datenbanken, die Informationen beinhalten und von einem System verarbeitet werden können,
2. Glossare, mit deren Hilfe Informationen abgebildet werden können
3. Thesauri,
4. Ontologien zur Beschreibung und Erfassung der Domäne,
5. im Internet bereitgestellte Quellen zur Anreicherung des Vokabulars (zum Beispiel Webservices) und
6. Domänenexperten, die in letzter Instanz die Beschreibung der Domäne vervollständigen.

Im Idealfall beschreiben die Quellen 1. bis 5. bereits Beziehungen zwischen den einzelnen Objekten, die der Informationsanreicherung dienen. Domänenexperten sollten erst dann dazu gezogen werden, wenn alle anderen Quellen erschöpft sind und Beziehungen oder Merkmale offen bleiben [Lenz1999, S.95, S. 124f].

Geschäftsmodell

Das Geschäftsmodell gibt vor, wie die Anwendung genutzt werden soll. Dafür müssen alle Benutzergruppen bekannt sein und Prozesse definiert werden wie die Stakeholder auf die Anwendung zugreifen. Dabei soll eine Anwendung einen möglichst geringen Mehraufwand für die Benutzer mit sich bringen und in vorhandene Prozesse integriert werden.

5.2 Definition des Fallformats

Das Fallformat gibt an, wie die Fälle einer Fallbasis aufgebaut sind und worauf das Retrieval ausgeführt werden kann. Im Rahmen der Domänenmodellierung muss ein Fallformat gefunden werden, welches alle Fälle, die in der Fallbasis erfasst werden sollen, abbilden kann. Somit legt es fest, aus wie vielen und welchen Arten von IEs ein Fall besteht.
Bei der Modellierung muss darauf geachtet werden, dass die Inhalte auch korrekt repräsentiert werden. So sind Freitexte über Texte über Text-IEs, Wertausprägungen jedoch als AVPs zu erfassen.
Wird ein Attribut als AVP modelliert, so muss sichergestellt werden, dass die möglichen Ausprägungen bekannt sind. Wenn beispielsweise ein Attribut den Preis eines Produktes darstellt, muss der Modellierer das Format und den Datentyp der beachteten Werte angeben. Sind die Werte der Attribute dagegen aus einer Aufzählung zu entnehmen, so müssen alle möglichen Ausprägungen vorab angegeben werden. Die Beschreibung des Aufbaus eines Fallformates kann Abbildung 13 entnommen werden. Die dort dargestellte DTD (engl. Document Type Definition) zeigt den Schematischen Aufbau eines Fallformates, welches durch die Modellierung instanziiert wird.

```
1.  <!ELEMENT case      (section)+>
2.  <!ELEMENT section   (name,datatype,isRetrieval,kindOfRetrieval)>

3.  <!ELEMENT name              (#PCDATA)>
4.  <!ELEMENT datatype          (#PCDATA)>
5.  <!ELEMENT isRetrieval       (#PCDATA)>
6.  <!ELEMENT kindOfRetrieval   (#PCDATA)>
```

Abbildung 13: DTD des Fallformates

Jeder Fall besteht aus einer oder mehreren *sections* (Zeile 2), die durch ihren Namen *name*, den Datentyp *datatype* und die Retrievaleigenschaften *isRetrieval* und *kindOfRetrieval* näher beschrieben sind. *isRetrieval* gibt dabei an, ob die section indexiert wird. Handelt es sich um ein Retrievalattribut, so muss zudem die Art des Retrievals (*AVP* oder *Text*) festgelegt werden. In Anhang 9.2 befindet sich ein instanziiertes Beispiel eines Fallformates entsprechend der in Abbildung 13 angegebenen DTD.

5.3 Bestimmung der Wertebereiche

Die Wertebereiche geben im Domänenmodell an, welche Werte für die Attribute erwartet werden können. Es ist wichtig, dass der Wertebereich eines Attributes bekannt ist, damit fehlerfrei auf die Daten zugegriffen werden kann. Gegebenenfalls muss der Aufnahme der Daten eine geeignete Konvertierung vorangehen. Festgelegt wird der Datentyp im Fallformat, wie es in Abbildung 13, Zeile 2, durch die DTD vorgeschrieben wird.

Darüber hinaus weisen die Wertebereiche darauf hin, was für eine Art von Retrieval für den betrachteten Bereich in Frage kommt. Falls der Abschnitt aus Freitext besteht, bietet sich ein Retrieval über den Text (mit Repräsentation durch IEs) an, wogegen sich bei Aufzählungen oder Zahlen Attribut-Werte-Paaren eher eignen (mit Repräsentation durch AVPs).
Für die Werte der AVPs müssen mögliche Ausprägungen angegeben und Ähnlichkeiten zwischen ihnen modelliert werden. Dies ist vor allem notwendig, wenn es sich nicht um reine Zahlenwerte, sondern um Aufzählungen handelt. Die Mengen von Attributausprägungen können in Nominal-, Ordnial, Intervall- oder Verhältnisskalen eingeordnet werden. Weiterhin können Begriffe in Taxonomien (is-a, part-of) modelliert werden.

5.4 Datenmodell der Fallbasis

Die Fallbasis als zentrale Komponente im TFBS mit CRNs hält die Fälle für das Retrieval vor.
Nach dem Indexieren wird das CRN durch die erkannten IEs aufgebaut. Dafür müssen die IEs zur Repräsentation der Fälle bereitgestellt werden. Das in Abbildung 14 dargestellte Datenmodell wurde in Anlehnung an [Hilbig2003] zur Erfassung der Fallbasis in einer relationalen Datenbank entwickelt. Das Modell verzichtet auf die Abbildung der Ähnlichkeits- und Relevanzkanten, da das CRN nicht in der Datenbank sondern im Arbeitsspeicher aufgebaut werden soll.

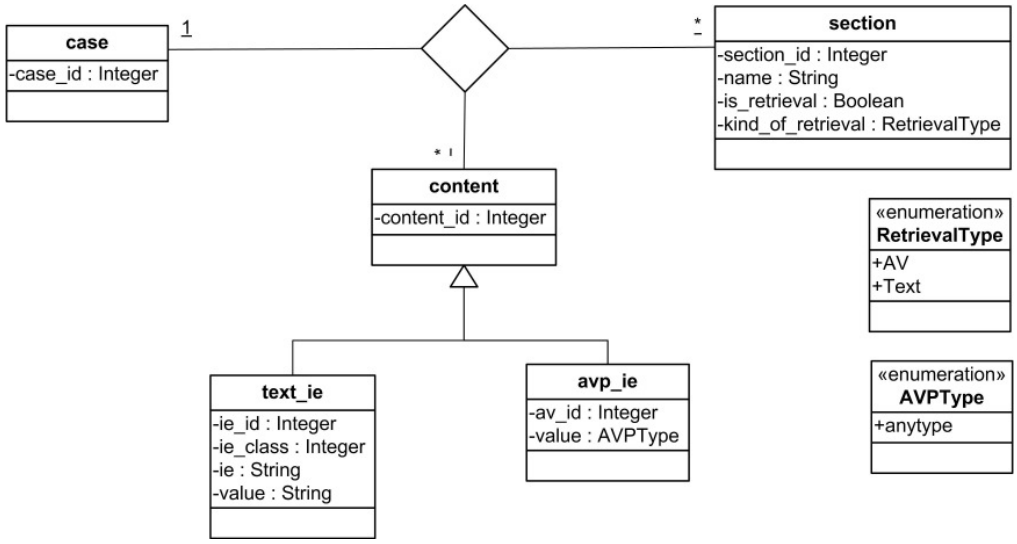


Abbildung 14: Datenmodell der Fallbasis

Das Datenmodell der Fallbasis enthält nicht die Originalfälle, sondern nur die für das Retrieval notwendigen Informationen in Form von IEs. Jeder Fall ist durch die *case_id* eindeutig identifiziert und besteht aus einer oder mehreren *sections*, die wiederum durch einen Namen *name* und seine Retrievaleigenschaften beschrieben sind. Dabei wird unterschieden, ob eine *section* für das Retrieval bereit steht oder nicht. Für retrievalrelevante Abschnitte muss angegeben werden, welcher Retrieval-

typ angewandt wird. Beim Aufbau des CRNs (vgl. Abschnitt 3.2) wurden AVPs und Text-IEs vorgestellt, die sich im Attribut *RetrievalType* widerspiegeln.
Für jede *section*, die für das Retrieval genutzt wird, werden die darin vorhandenen IEs in der generalisierten Klasse *content* erfasst. Die Spezialisierungen sind *text_ie* und *avp_ie*. In *text_ie* werden IEs gespeichert, wenn der *RetrievalType* „Text" vorliegt. In einer *section* können beliebig viele *text_ie* und/oder *avp_ie* Objekte vorhanden sein, die über die *content_id* eindeutig identifizierbar sind. Zu jeder *text_ie* wird neben der eigentlichen IE (*value*) erfasst, durch welche *ie_id* sie im CRN repräsentiert wird. Ist die aktuell betrachtete *text_ie* eine Ausprägung einer IE, so wird zudem die Ähnlichkeit des Wertes zur IE aufgenommen.
Die Daten der Klasse *text_ie* stammen aus dem Indexlexikon (vgl. Abschnitt 3.1.3). Für jedes Objekt von *avp_ie* werden die *av_id* und der Wert (*value*) in die Fallbasis aufgenommen. Dabei können nur modellierte AVPs betrachtet werden. Die Modellierungen können beliebig gestaltet werden.
In der Fallbasis wird darauf verzichtet, den gesamten Inhalt der Fälle vorzuhalten, da über die *case_id* auf die Rohdaten referenziert werden kann. Dies hat den Vorteil, dass nur solche Daten in der Fallbasis vorhanden sind, über die auch das Retrieval erfolgt.
Das Datenmodell befindet sich nicht in der dritten Normalform, denn *ie* und *value* sind redundant. Dadurch kann die Weiterverarbeitung aber effizienter gestaltet werden. Für ein Datenmodell in dritter Normalform reicht es aus, wenn die Klasse *text_ie* die *ie_id* enthält, da alle weiteren Informationen über Verbunde (engl. joins) mit den Klassen des Datenmodells vom Indexlexikon hergeleitet werden können. Bei großen Fallbasen und umfangreichen Indexlexika ist dieser Verbund sehr aufwendig, so dass die redundante Datenhaltung einen schneller Zugriff ermöglichen kann.

5.5 Anreicherung des Indexvokabulars

Als Ausgangspunkt für das Indexvokabular wurden die potentiellen Text-IEs aus dem ExperienceBook II[8] wurden entnommen, welches an der Humboldt-Universität zu Berlin entwickelt wurde. Das Assistenzsystem dient dem Austausch von Erfahrungswissen, unterstützt bei der Systemadministration und wird für die administrativen und technischen Belange der täglichen Arbeit jenseits der Forschungsarbeit am dortigen Lehrstuhl eingesetzt [vgl. Hanft2004, S. 9].
Die IEs liegen strukturiert vor und sind folgendermaßen klassifiziert:

- allgemeinsprachliche Begriffe
- Fachbegriffe (z.B. aus der IT-Welt)
- Attributdefinitionen
- Attribut-Werte-Paare
- spezifische Begriffe

Bei der Anwendung des vorhandenen Vokabulars auf eine neue Domäne werden weniger Worte als IEs erkannt. Der Grund dafür ist, dass das Vokabular des

[8] https://roy.informatik.hu-berlin.de/ExpBookII/

ExperienceBook II speziell für die Beschreibung von Problemen in der Systemadministration erstellt wurde.
Daher bedarf es Erweiterungen des Vokabulars, die in den nachfolgenden Abschnitten vorgestellt werden. Das Ziel der Anreicherung des Vokabulars aus externen Wissensquellen besteht darin, die unbekannte Domäne so weit wie möglich zu erfassen.

5.5.1 Anreicherung durch GermaNet[9]

GermaNet ist ein lexikalisch-semantisches Netz, welches an der Universität Tübingen als Thesaurus – ähnlich dem Vorbild WordNet der Princton University[10] – entwickelt wurde und die häufigsten Konzepte des deutschen Grundwortschatzes abbildet.
Die Wörter in GermaNet sind in XML-Dateien abgespeichert, abhängig von ihrer Wortart und einer inhaltlichen Klassifikation. Sie sind in so genannten *Synsets* (set of synonyms) angeordnet, so dass die Synonyme leicht mit einem XPath-Ausdruck aufgefunden werden können.
GermaNet umfasst 54 XML-Dateien, die die Adjektive, Nomen und Verben in ihrer jeweiligen Grundform enthalten. Jede Datei besteht aus *Synsets* und dazugehörigen *lexUnits*:
Ein Synset beschreibt ein Konzept zu einem Wort (Nomen, Adjektiv, Verb) und hat in GermaNet folgende Attribute:

id	zur eindeutigen Identifikation des Synsets
lexGroup	gibt die Klassifizierung, die so genannten Semantic Fields (tops) an
wordClass	gibt die Klasse an, zu dem das Wort gehört

Eine *lexUnit* beschreibt eine lexikalische Einheit und hat folgende Attribute

id	zur eindeutigen Identifikation der LexGroup
stilMarkierung	gibt an, ob das Wort umgangssprachlich ist oder nicht
sense	Beschreibt je nach Klasse, wie das Wort zu deuten ist (Bsp Adjektiv: hell vs. dunkel)
orthVar	gibt an, ob es andere Schreibweisen für das Wort gibt („Yoga“ und „Joga“)
artificial	gibt an, ob der Begriff künstlich erzeugt ist (damit Taxonomien aufgebaut werden können oder Konzepte vollständig dargestellt werden können)
Eigenname	gibt an, ob der Term ein Eigenname ist

Die XML-Struktur von GermaNet-Dateien sieht folgendermaßen aus:

[9] http://www.sfs.uni-tuebingen.de/lsd/
[10] http://wordnet.princeton.edu/

```
<synsets>
      <synset id="nMensch.1" lexGroup="Mensch" wordClass="nomen">
            <lexUnit Eigenname="nein" artificial="nein"
            id="nMensch.1.Mensch" orthVar="nein" sense="1"
            stilMarkierung="nein">
                  <orthForm>Mensch</orthForm>
             </lexUnit>
            <lexUnit Eigenname="nein" artificial="nein"
            id="nMensch.1.Person" orthVar="nein" sense="1"
            stilMarkierung="nein">
                  <orthForm>Person</orthForm>
            </lexUnit>
            <lexUnit Eigenname="nein" artificial="nein"
            id="nMensch.1.Persönlichkeit" orthVar="nein" sense="1"
            stilMarkierung="nein">
                  <orthForm>Persönlichkeit</orthForm>
            </lexUnit>
            <lexUnit Eigenname="nein" artificial="nein"
            id="nMensch.1.Individuum" orthVar="nein" sense="1"
            stilMarkierung="nein">
                  <orthForm>Individuum</orthForm>
            </lexUnit>
      </synset>
<synsets>
```

Abbildung 15: XML-Beispiel aus GermaNet

In Abbildung 15 wird das Synset *Mensch* beschrieben. Das Attribut lexGroup gibt an, dass das Wort *Mensch* zur lexikalischen Gruppe Mensch gehört und ein Nomen (*wordClass*) ist. In GermaNet sind für *Mensch* wie in dem XML-Beispiel gezeigt drei Synonyme gelistet, die jeweils durch eine *lexUnit* repräsentiert werden.

5.5.1.1 Erweiterung der IEs durch GermaNet

In der hier beschriebenen Anwendung dient GermaNet als Erweiterung des Indexvokabulars des ExperienceBook II, um neben der speziellen Domäne (Systemadministration) auch unbekannte Domänen erschließen zu können.
Dafür wurden zur Bestimmung der IEs alle in GermaNet befindlichen Nomen extrahiert und mit ihrer Klassenbezeichnung in die Struktur der vorhandenen IE-Tabelle der Datenbank eingefügt (vgl. Abbildung 14). Die darüber hinaus vorhandenen Adjektive und Verben wurden genutzt, um für das System unbekannte Worte zu ermitteln (vgl. Abschnitt 5.9).

Die potentiellen IEs konnten durch das Einbinden von GermaNet um die folgenden Klassen erweitert werden:

- Artefakt
- Attribut
- Besitz
- Form
- Gefühl
- Geschehen
- Gruppe
- Körper
- Kognition
- Kommunikation
- Menge
- Mensch
- Motiv
- Nahrung
- natGegenstand
- natPhaenomen
- Ort
- Pflanze
- Relation
- Substanz
- Tier
- Tops
- Zeit

Die komplette Liste der Klassifizierung der aus GermaNet stammenden IEs sind im Anhang, Abschnitt 9.2 aufgelistet. Bei der Klassifizierung wurden die Quellen aus denen die Indexterme stammen erfasst, damit im Nachhinein nachvollziehbar ist,

wie sich das Indexvokabular zusammensetzt. Die IE-Klasse unterscheidet die verschiedenen Kategorien in denen sich Indexterme befinden. Die Klassifizierung wurde aus dem ExperienceBook II und GermaNet übernommen.

5.5.1.2 Grenzen von GermaNet

Die Grenzen von GermaNet liegen in der Auflösung von Polysemie, da es vorkommen kann, dass bei der Suche nach Synonymen ein und dasselbe Wort in mehreren Dateien auftaucht, wie es bei *Haus* der Fall sein kann. So werden bei der Suche nach *Haus* folgende Synonyme geliefert: *Hausbesitz*, *Hauseigentum*, *Geschlecht*, *Sternzeichen*, *Sternbild*, *Tierkreiszeichen*. Dies erklärt sich aus der im Anhang (siehe Abschnitt 9.2) auszugsweise dargestellten Struktur der GermaNet-XML-Dateien nomen.Artefakt.xml, nomen.Besitz.xml, nomen.Gruppe.xml und nomen.Kommunikation.xml.
Im oben genannten Beispiel wird verdeutlicht, wie die verschiedenen Bedeutungen des Begriffs *Haus* durch Einträge in verschiedenen Synonym-Konzept-Knoten dargestellt sind. Diese können sich wiederum in verschiedenen Konzept-Klassen (und somit verschiedenen GermaNet-XML-Dateien) befinden.

5.5.1.3 Ergebnis der Anreicherung

Nachdem die Einträge aus GermaNet in das Indexvokabular aufgenommen wurden, hat die Wissensquelle GermaNet auch Auswirkungen auf das Ähnlichkeitsvokabular. Denn jeder Eintrag aus GermaNet, der in seinem *Synset* ein oder mehrere Synonym(e) enthält, wird auch in das Ähnlichkeitsvokabular aufgenommen. Bei einer Anfrage werden im CRN dadurch nicht nur die in der Anfrage enthaltenen IEs, sondern auch zu ihnen ähnliche Indexterme aktiviert. Nach der Erweiterung des Indexvokabulars durch GermaNet können vor allem allgemeinsprachliche Begriffe erkannt werden. Im Hinblick auf die Domänenmodellierung für unbekannte Domänen reduzieren sich somit die nicht erkannten Worte auf potentiell domänenspezifisches Vokabular und Rechtschreibfehler (vgl. Abschnitt 5.9).
Die Datenstruktur des Indexlexikons (Abschnitt 3.1.3) zeigte, dass zu jeder Ausprägung entsprechend der Ähnlichkeitsklasse (sim_class) ein Ähnlichkeitswert zugeordnet werden kann. Wie in [Minor2006] beschrieben, wird von den absoluten Zahlenwerten der Ähnlichkeitsfunktion auf Ähnlichkeitstypen abstrahiert, um die Modellierung zu vereinfachen. Ein Ähnlichkeitstyp ist wie folgt definiert:

Definition 22: Ähnlichkeitstyp

> Ein Ähnlichkeitstyp S ist eine binäre Relation zwischen IEs, es gilt also $S \subseteq E \times E$. Eine Menge S von Ähnlichkeitstypen bezeichnet das Ähnlichkeitslexikon.

Ähnlichkeitstypen haben entweder einen quantitativen oder qualitativen Bezeichner. Quantitative Bezeichner sind z. B. „WEAK_SIM“ für „leichte Ähnlichkeit“ oder „STRONG_SIM“ für „starke Ähnlichkeit“. Für die Beziehung zwischen IEs, die aus einem Synset stammen, wird der Ähnlichkeitstyp „STRONG_SIM“ angenommen, da es sich um Synonyme handelt. Quantitative Bezeichner werden genutzt, um beispielsweise „HAS_PART“-Verbindungen zu modellieren [vgl. Minor2006, S. 46].

5.5.2 Anreicherung durch das *Projekt Deutscher Wortschatz*[11]

Die vorhandenen IEs aus GermaNet lieferten unvollständige oder keine grammatikalischen Formen der IEs und ihrer Ausprägung. Aus diesem Grund werden die Daten des Projekts Deutscher Wortschatz genutzt, um die grammatikalischen Ausprägungen hinzuzufügen. Denn gerade bei Beschreibungen von Sachverhalten im Freitext werden viel häufiger gebeugte Wortformen als Grundformen genutzt. Die Flexionen der Worte erweitern das Indexvokabular und lassen das System mehr potentielle IEs erkennen, die zuvor als unbekannt identifiziert wurden [vgl. Minor2006].
Das Projekt Deutscher Wortschatz baut zu einzelnen Sprachen umfangreiche Textkorpora auf, wertet diese statistisch aus und stellt die Ergebnisse als Webservice zur Verfügung. Für das Deutsche sind zusätzlich syntaktische und semantische Angaben wie die Grundform zu Vollformen, Grammatikangaben zur Grundformen, Sachgebiete und Synonyme erschlossen [vgl. Quasthoff2006].
Für die Anreicherung wurde ein Webservice genutzt, der auf der Webseite vom Projekt Deutscher Wortschatz (http://wortschatz.uni-leipzig.de/) angeboten wird und einen direkten Zugriff auf die Daten erlaubt. Die Schnittstelle ist standardisiert (SOAP) und offen gelegt. Das Projekt bietet verschiedene Stufen des Zugriffs an, die von einfachen Datenabfragen (Stufe 1), über komplexere Abfragen mit entsprechenden Rechenzeitaufwand (Stufe 2) bis hin zu experimentellen Abfragen oder Massendatenabfragen (Stufe 3). Die Abfrage des GermaNet-Korpus' bezüglich der Flexionen der einzelnen Worte wurde mit einem Zugang der Stufe 3 realisiert. Die Anbindung wird in Abschnitt 6.1 erläutert.
Als Ergebnis der Anreicherung stehen neben den Begriffen aus GermaNet weiterhin ihre Ausprägungen zur Verfügung, damit mehr potentielle IEs in den zu analysierenden Texten gefunden werden kann. Denn durch die Erweiterung können Flexionen auf ihre Grundform zurückgeführt und einem bereits vorhandenem IE zugeordnet werden.

5.5.3 eCl@ss zur Erkennung von Fachbegriffen

eCl@ss ist ein standardisiertes Klassifikationssystem für Warengruppen und Warenmerkmale mit dem Ziel, Produkte des elektronischen Handels zu klassifizieren. Der eCl@ss-Ansatzes stellt ein Informationsmodell bereit, was den Informationsaustausch über Unternehmensgrenzen hinaus ermöglicht. Neben eCl@ss gibt es noch weitere Klassifikationsmodelle für verschiedene Branchen (z.B. ETIM, proficl@ss, UNSPSC, ECCMA, …) [vgl. Heeg2003]. Mit der Version 5.1.3 sind die wichtigsten Industriestandards zur Klassifikation von ETIM, proficl@ss und bau:class integriert [vgl. Heeg2003].

[11] http://wortschatz.uni-leipzig.de/

Das Datenmodell besteht aus folgenden Entitäten:

- Klassen in vier Klassifikationsebenen
- Schlagworte und Synonyme der klassifizierten Produkte
- Standardmerkmalleisten
- Merkmale
- Merkmalsausprägungen

Abbildung 16 zeigt das komplette Datenmodell von eCl@ss, wie es dem Nutzer nach dem Download der Daten von der Website[12] steht.

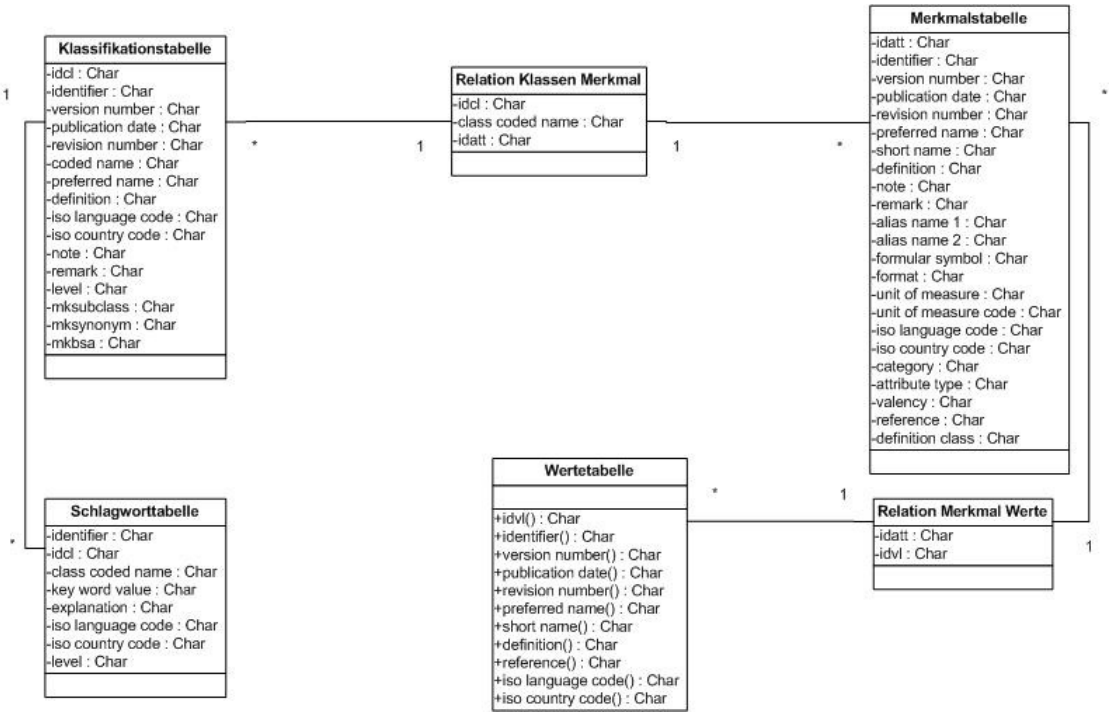


Abbildung 16: Datenmodell von eCl@ss

Die in eCl@ss verwendete Klassifikationshierarchie umfasst vier Ebenen, deren Spezialisierung mit der Hierarchietiefe zunimmt (vgl. Abbildung 17):

1. Ebene: Sachgebiet
2. Ebene: Hauptgruppe
3. Ebene: Gruppe
4. Ebene: Untergruppe

Die Beschreibung der Klassen auf den Ebenen 1 bis 4 erfolgt durch ihre Bezeichnungen und der bijektiven Zuordnung zusätzlicher Schlagworte.

Die Untergruppen werden zusätzlich durch die Zuordnung einer Standard-Merkmalleiste beschrieben. Die Standard-Merkmalleisten dienen der Strukturierung der klassenspezifischen Gruppierung der Merkmale. Bei der Klassifizierung wird zwischen klassifizierenden und quantitativen Merkmalen unterschieden. Die Ausprägungen der klassifizierenden Merkmale sind numerisch oder alphanumerisch und haben keine Einheit, wogegen die quantitativen Merkmale ausschließlich numerische Werte (int, float, double…) annehmen können und eine Einheit besitzen. Die Merkmale werden durch eine Reihe von Attributen genauer beschrieben [vgl. Heeg2003]:

[12] http://www.eclass.de

- *Identifizierende Attribute* zur eindeutigen Identifikation der Objekte
- *Semantische Attribute* dienen der Definition der einzelnen Objekte
- *Attribute der Datenwerte* dienen der Festlegung von Einheiten und so genannten Werteformaten
- *Relationale Attribute* werden für die Verknüpfungen der Objekte genutzt

Abbildung 17 zeigt exemplarisch die Klassifikation eines *Produkts* (Funktelefons) in eCl@ss inklusive der dazugehörigen Schlagworte (roter Kasten) und Merkmalleiste (blauer Kasten).

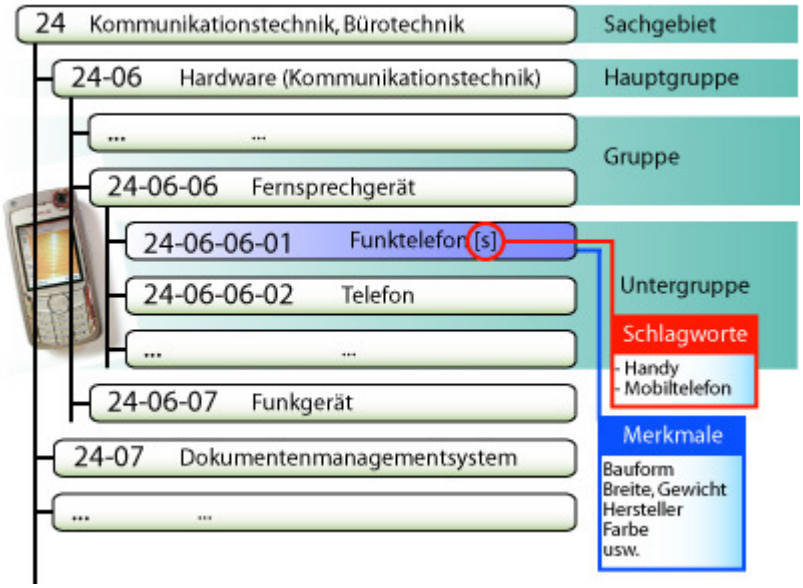


Abbildung 17: Beispiel eines Produktes in eCl@ss

Aus der Abbildung 17 wird weiterhin ersichtlich, dass eCl@ss keine Produktnamen, sondern die Produktklassifikation darstellt.
Die Klassifikation der Produkte bietet sich an, um Fachbegriffe in unbekannten Texten zu erkennen und Ähnlichkeiten zu modellieren. Denn alle zu einer Untergruppe gehörenden Produkte wie auch die Schlagworte können das Indexvokabular anreichern. Dafür müssen die Daten aus der eCl@ss Datenbank extrahiert und in die in Abschnitt 3.1.3 vorgestellte Datenstruktur des Indexlexikons transformiert werden.

5.5.4 Behandlung von Rechtschreibfehlern

Je nach Anwendung können sich Tippfehler in den Quelldokumenten wiederholen, so dass es wichtig ist zu definieren, wie sie verarbeitet werden sollen: Entweder die Tippfehler werden als Ausprägungen der IEs in den Korpus aufgenommen und als Indexterme behandelt oder sie werden korrigiert und das falsch geschriebene Wort wird explizit als Tippfehler in das Indexvokabular aufgenommen. In dem hier vorliegenden Ansatz wurde die zweite Möglichkeit gewählt, da die Datensätze durch die Korrekturen von den Rechtschreibfehlern befreit werden, jedoch die Syntax möglicher Schreibfehler bekannt ist. Dem Modellierer kann bei wiederholtem Auftreten vermeintlich falsch geschriebener Worte die bekannte Korrektur vorgeschlagen werden. Handelt es sich dagegen um ein korrekt geschriebenes Wort, kann es aber

auch im Indexvokabular als IE oder Ausprägung eines IEs erfasst werden. Dadurch werden Tippfehler bei der Analyse des Korpus identifiziert und als solche klassifiziert, aber trotzdem die Verbindung zu einem IE hergestellt. Die Klassifizierung wird in der Klasse *quality* (vgl. Abbildung 9: Datenmodell des Indexlexikons) vorgenommen, indem ein Unterschied zwischen initial vorhandenen Worten, manuell erfassten Worten und Rechtschreibfehlern gemacht wird.

5.5.5 Ausblick: Weitere Möglichkeiten der Vokabularanreicherung

5.5.5.1 eCl@ss zur Identifikation von Produkten

In Abschnitt 5.5.3 wurde eCl@ss als Klassifikationsmodell vorgestellt. Neben der allgemeinen Beschreibung der Produkte kann eCl@ss ebenfalls genutzt werden, um Produktnamen zu identifizieren. Allerdings ist dafür die Produktdatenbank der Unternehmen notwendig, um über die Klassifikationsnummer auf die Artikelnummer oder den EAN-Code zu schließen. Die Integration dieser Daten kann im Weiteren auch dabei helfen, ähnliche Produkte von verschiedenen Anbietern zu erkennen.
Stehen die Produktkataloge zur Verfügung, so können die einzelnen Produktbezeichnungen in das Vokabular aufgenommen werden. Damit eine Beziehung zwischen der Produktbezeichnung und den Begriffen, die bereits im Vokabular aufgenommen wurden, hergestellt werden kann, muss die Produktbezeichnung als IE-Ausprägung aufgenommen werden.
Für Korpora, die viele Produkteigennamen enthalten, kann diese Anbindung sehr nützlich sein (vgl. Abschnitt 7.1.2f).

5.5.5.2 Named Entity Recognition

Eine weitere Möglichkeit, Eigennamen zu erkennen, ist die Named Entity Recognition (NER). Die NER entstammt dem Bereich der Information Extraction des Information Retrieval und ermöglicht das Erkennen, Klassifizieren und Markieren von Named Enitities wie beispielsweise Personen(-namen), Organisationen, Orten, Datums-, Zeit- oder Prozentangaben sowie Geldbeträgen.
Das NER basiert entweder auf *listenbasierten Systemen*, *regelbasierten Systemen*, Systemen des *maschinellen Lernens* oder *hybriden Systemen*. Listenbasierte Systeme erkennen Eigennamen lediglich aufgrund vorher bekannter Worte, die in Listen erfasst sind. Die Leistungsfähigkeit hängt vom Umfang der Liste ab – Variationen sind schwer zu erfassen und Ambiguitäten (Mehrdeutigkeiten) nicht aufzulösen, so dass auf die anderen beiden Ansätze zurückgegriffen werden sollte. Basiert die NER auf maschinellem Lernen, so muss ein Trainingskorpus vorbereitet und das System damit trainiert werden. Anhand der darin enthaltenen statistischen Vorkommnisse von Named Entities können später auch Eigennamen mit hoher Zuverlässigkeit in unbekannten Texten erkannt werden. Regelbasierte Systeme arbeiten mit vorab erstellten Regeln und können somit nicht ad-hoc eingesetzt werden [MaynardEtAl2001].
Sind in einer Domäne sehr viele verschiedene Eigennamen vorhanden, eignet sich das NER nicht, da ein repräsentativer Korpus gefunden oder erstellt werden muss. Ist das Vokabular nicht allzu weitläufig kann man NER in die Überlegung der Wort-

erkennung einschließen. Allerdings muss dann abgewogen werden, welchen Aufwand es kostet, das Vokabular von Hand anzureichern oder im Gegenzug einen NER-Tagger zu trainieren.

5.5.5.3 Kataloge aus Webportalen

Als weitere Quelle für strukturierte Begriffshierarchien können Kataloge wie beispielsweise yahoo.de[13], web.de[14] oder wissen.de[15] dienen. In ihnen sind Begriffe bereits hierarchisch angeordnet, so dass etwa ein Benutzer durch die Themen navigieren kann, um sein Suchgebiet einzugrenzen. Innerhalb einer jeden Kategorie findet er Verweise zu Dokumenten, die der aktuell betrachteten Kategorie angehören oder mögliche Unterkategorien anbieten, um das Suchgebiet zu konkretisieren [vgl. Glintschert1998 und Bell2004].
Diese vorgefertigten Strukturen aus den Webportalen können ebenfalls als Grundlage für den Aufbau von Taxonomien verwendet werden, denn sie bieten den Vorteil einer bereits vorhandenen groben Struktur mit Fachbegriffen.

5.6 Analyse der zugrunde liegenden Dokumente

Die Domänenanalyse beginnt wie bereits in Abschnitt 5.1 beschrieben mit der Betrachtung der zugrunde liegenden Dokumente. Dabei wird unterschieden, ob die Dokumente voll-, semi- oder nicht strukturiert sind. Vollstrukturierte Dokumente sind beispielsweise Datenblätter zu technischen Geräten, semistrukturierte Dokumente bezeichnen unter Anderem FAQ-Listen und nicht strukturierte Dokumente sind Freitextdokumente.
Ebenso wird unterschieden, ob die Dokumente in ein und derselben Sprache vorliegen. Zudem sollte vorab angegeben werden, in welcher Domäne das FBS-System genutzt werden soll, damit dieses Wissen zum Auflösen von Ambiguitäten genutzt werden kann. Denn falls Entscheidungen getroffen werden müssen, welche IE gewählt wird, kann dies entscheidend für das Ergebnis des Retrievals sein. Dennoch kann es bei der Auflösung von Polysemie zu Problemen kommen, wenn ein Wort eine doppelte Bedeutung hat (Ambiguitäts-Problem) oder ein Sachverhalt mit unterschiedlichen Worten ausgedrückt werden kann (Paraphrase-Problem) [vgl. Adriani2000]. Ein Beispiel für das Ambiguitäts-Problem ist das Wort *Bank*, was zum einen das Geldinstitut und zum anderen die Sitzgelegenheit beschreiben kann. Das Paraphrase-Problem tritt beispielsweise auf, wenn in der Versicherungsdomäne einmal von *Versicherungsnehmern* und in einem anderen Zusammenhang von *Kunden* gesprochen wird, obwohl der *Kunde* einer Versicherung der *Versicherungsnehmer* ist.
Damit so viele Textbereiche wie möglich in Betracht gezogen werden, ist es wichtig, die Struktur der Dokumente so homogen wie möglich zu halten. Falls dies nicht möglich ist, schlägt [Minor2006, S.60] vor, pro Dokumentenart eine Fallbasis mit

[13] http://de.yahoo.com/
[14] http://web.de/
[15] http://wissen.de/

einer eigenen Fallstruktur anzulegen. Damit Anfragen trotzdem über alle Wissensquellen laufen, müssen die Fallstrukturen konsolidiert werden.
Weiterhin ist zu beachten, dass das erste Beispiel in einer Falldatenbank immer als Referenzfall für einen Sachverhalt gilt und somit als Vergleichsdokument für die nachfolgenden dient.
Neben denen im Dokument verwendeten Fachwörtern können weitere Informationen enthalten sein, die jedoch nicht direkt in textueller Form vorliegen, wie z. B. zusätzliche Strukturinformationen, Diagramme oder implizite Informationen.

5.7 Erfassen der einzelnen Fälle

In diesem Abschnitt wird dargestellt, wie ein konkreter Fall in die Fallbasis aufgenommen wird.
Die Extraktion der IEs unterscheidet sich darin, ob nach Text-IEs oder AVPs gesucht wird. Auf der Suche nach Text-IEs wird ein Abschnitt geparst und alle darin erkannten IEs im Datenmodell gespeichert. Soll ein Textabschnitt dagegen durch ein AVP repräsentiert werden, muss nach den im Fallformat vorgegebenen Einträgen gesucht werden. Dabei sind nur jene Werte zulässig, die auch vorab modelliert wurden.
Das Ergebnis der IE-Extraktion ist eine Menge von IEs, die den jeweiligen Textabschnitt repräsentieren.

Das Beispiel, an dem die IE-Extraktion gezeigt werden soll, stammt aus der Versicherungsdomäne. In dieser Domäne werden Schadensfälle der Versicherungsnehmer aufgenommen und deren Prüfung veranlasst. Neben der Aufnahme der personenbezogenen Daten der Versicherungsnehmer erfassen die Sachbearbeiter ebenfalls den Schadensfall. Diese Angaben werden in der Anwendung genutzt, um ähnliche Vorfälle zu finden.
In Abbildung 18 ist ein konkreter Schaden und dessen Weiterverarbeitung erfasst. Die Rohdaten (grün hinterlegt) stammen aus einem Operativsystem und zeigen, dass ein vollständiger Fall aus zwölf Attributen besteht, deren Ausprägung entweder Freitext, Zahlenwerte oder Zahlenwerte mit Einheiten sind. Die *Id* ist ein künstlich erzeugtes Attribut zur eindeutigen Identifizierung eines Falles. Die Attribute *Ursache*, *Bemerkung*, *Kurzbeschreibung*, *Anschaffungswert*, *Zeitwert*, *Objekt*, *Zustand*, *Gerätealter* und *Schäden* werden bei einer Neuerfassung einer Schadensmeldung im System erfasst und sollen damit als Retrievalattribute gelten. Die Attribute Prüfergebnis und Reparaturkosten enthalten Informationen, welche von Gutachtern hinzugefügt wurden, nachdem sie die Schadensmeldung begutachtet haben. Diese beiden Attribute werden in das Retrieval nicht mit einbezogen, dienen jedoch als Informationsattribute und bilden somit einen Teil der Lösung auf eine Anfrage.
Im Rahmen der Modellierung des Fallformates wurde festgelegt, wie die einzelnen Attribute weiterverarbeitet werden sollen (gelbe Spalte). Die Spalte *is_retrieval* zeigt auf, ob das Attribut für das Retrieval verwendet werden soll. Falls es sich bei einem Attribut um ein Retrievalattribut handelt, gibt die Spalte *kind_of_retrieval* an, ob es sich um ein AVP (AV) oder ein Text-IE (Text) handelt.

		is_retrieval	kind_of_retrieval	IE
Id	1612	FALSCH		
Ursache	Verdacht auf Flüssigkeitsschaden.VU bittet um Info über Schadensursache und vielleicht über Reparaturmöglichkeiten	WAHR	Text	Verdacht, Info, Schadensursache, Reperaturmöglichkeiten
Bemerkung	Gerät kann beim Anspruchsteller besichtigt werden.	WAHR	Text	Gerät
Kurzbeschreibung	HIFI-Stereoanlage	WAHR	Text	Stereoanlage
Anschaffungswert	200 €	WAHR	AV	200
Zeitwert	50 €	WAHR	AV	50
Objekt	HIFI- Verstärker Pioneer, Model A-204R	WAHR	Text	Verstärker, Model
Zustand	gebraucht, leichte Kratzer am Verstärkergehäuse. An der Front fehlen vier Einstell-Drehknöpfe.	WAHR	Text	gebraucht, Kratzer, Front, fehlen
Geräteälter	9	WAHR	AV	9
Schäden	Der Verstärker weist keine spannungsspezifische Funktion auf. Auf der Haupt- und Endstufenplatine sind Flüssigkeitsspuren zu finden. In diesem Bereich sind die Leiterbahnen und Bauteile wie z.B. Widerstände, Kondensatoren und IC- Kontaktbeine korrodiert.	WAHR	Text	Verstärker, keine, Funktion, Haupt, finden, Bereich, Bauteile, Widerstände
Prüfergebnis	Ein Feuchtikeits- bzw. Flüssigkeitsschaden ist plausibel, weil die typischen Spuren (getrocknette Flüssigkeitsflecken, korrodierte Bauteile und Leiterbahnen) auf den Haupt- und Endstufenplatinen zu finden sind.	FALSCH		
Reperaturkosten	Eine Reparatur des HIFI- Verstärkers würde den Neupreis eines Neuersatzgerätes weit übersteigen.	FALSCH		

Abbildung 18: Beispielfall zur Erfassung in der Fallbasis

Die dritte (blau hinterlegte) Spalte zeigt für die Retrievalattribute, welche Werte aus den Abschnitten extrahiert werden können, um sie in der Fallbasis zu erfassen. Abschnitte aus Text-IEs werden durch die sich im Freitext befindenden Indexterme repräsentiert, wogegen AVPs durch das Attribut und die konkreten Werte beschrieben sind. AVPs und Indexterme sind die IEs, die in einem CRN einen Fall repräsentieren.

5.8 Import der Fälle in die Fallbasis

Die Werte der Attribute werden wie im Datenmodell beschrieben in einer Datenbank abgelegt und stehen danach dem Retrieval zur Verfügung (vgl. Abbildung 19).
Die Tabelle *case* enthält die Id aller indizierten Fälle und in *section* sind alle Attribute zu diesem Fall aufgeführt, die im Fallformat bestimmt wurden. Falls es sich um ein Retrievalattribut handelt, nimmt das Attribut *is_retrieval* den Wert *TRUE* an und die Art des IEs ist angegeben. Wie im Datenmodell beschrieben, kann dies entweder *Text* oder *AV* sein.
Soll eine Section durch ein AVP (section.kind_of_retrieval=’AV’) repräsentiert werden, so wird nach einem Wert entsprechend des im Fallformat angegebenen Wertebereiches gesucht (referenziert durch av_id) und zugeordnet (*value*).

Wenn eine *section* durch eine Menge von Text-IEs repräsentiert werden soll, so wird der Abschnitt nach bekannten IEs durchsucht. In der Tabelle *text_ie* wird jede gefundene IE durch ein Tupel bestehend aus der *case_id*, *section.name* (stammen aus den Quelldaten), *ie_id*, *ie_class*, *ie* (aus dem Indexvokabular) und dem gefundenen Wort (*value*) dargestellt.

case	case_id
	1612

section	name	is_retrieval	kind_of_retrieval
	Ursache	WAHR	Text
	Bemerkung	WAHR	Text
	Kurzbeschreibung	WAHR	Text
	Anschaffungswert	WAHR	AV
	Zeitwert	WAHR	AV
	Objekt	WAHR	Text
	Zustand	WAHR	Text
	Gerätealter	WAHR	AV
	Schäden	WAHR	Text
	Prüfergebnis	FALSCH	
	Reperaturkosten	FALSCH	

avp_ie	case.case_id	section.name	av_id	value
	1612	Anschaffungswert	1	200
	1612	Zeitwert	1	50
	1612	Gerätealter	3	9

text_ie	case.case_id	section.name	ie_id	ie_class	ie	value
	1612	Ursache	32840	60	Verdacht	Verdacht
	1612	Ursache	32042	60	Information	Info
	1612	Ursache	32043	60	Info	Info
	1612	Ursache	98569	56	Schaden	Schadensursache
	1612	Ursache	41203	56	Reparatur	Reparaturmöglichkeiten
	1612	Bemerkung	14083	51	Gerät	Gerät
	1612	Kurzbeschreibung	15541	51	Stereoanlage	Stereoanlage
	1612	Objekt	18183	51	Verstärker	Verstärker
	1612	Objekt	38803	62	Model	Model
	1612	Zustand	7976	2	brauchen	gebraucht
	1612	Zustand	25342	56	Kratzer	Kratzer
	1612	Zustand	30107	58	Kratzer	Kratzer
	1612	Zustand	26872	57	Front	Front
	1612	Zustand	45638	67	Front	Front
	1612	Zustand	45637	67	Vorderseite	Front
	1612	Zustand	8890	2	fehlen	fehlen
	1612	Zustand	36001	61	vier	vier
	1612	Schäden	18183	51	Verstärker	Verstärker
	1612	Schäden	9746	2	kein	keine
	1612	Schäden	18936	51	Funktion	Funktion
	1612	Schäden	50154	69	Funktion	Funktion
	1612	Schäden	29293	58	Haupt	Haupt
	1612	Schäden	42730	62	Haupt	Haupt
	1612	Schäden	8941	2	finden	finden
	1612	Schäden	45487	67	Bereich	Bereich
	1612	Schäden	14084	51	Bauteil	Bauteile
	1612	Schäden	16061	69	Widerstand	Widerstände
	1612	Schäden	50488	69	Widerstand	Widerstände

Abbildung 19: Beispielfall zur Erfassung in der Fallbasis

5.9 Ermitteln der für das System unbekannten Worte

In den vorangegangenen Abschnitten wurde die Erkennung von Text-IEs beschrieben. Dafür ist es essentiell, ein möglichst umfangreiches Vokabular zur Verfügung zu haben. Dazu gehören allgemeinsprachliche wie auch domänenspezifische Indexterme. Das allgemeinsprachliche Vokabular ist durch das Einbinden des GermaNet-Wortschatzes sehr gut abgedeckt (vgl. Abschnitt 7.2 und Abschnitt 5.5.1), so dass im Weiteren spezielle, in der Domäne genutzte Begriffe, erkannt und aufgenommen werden müssen. Ein Ansatz dafür ist die Integration durch eCl@ss (vgl. Abschnitt 5.5.3). Im Rahmen dieser Arbeit wurde ein Verfahren entwickelt, dass den Modellierer bei der Erweiterung der Domäne unterstützt.
Auf der Suche nach Worten, die im Indexvokabular nicht vorhanden sind, bietet die Anwendung die Möglichkeit, die Quelldaten auf *unbekannten Worte* zu analysieren. *Unbekannte Worte* sind hierbei Worte, die weder zu den Stoppwörtern, den Indextermen, noch zu deren Ausprägungen gehören.
Nach der Extraktion der bekannten Worte sollen die nicht erkannten nachmodelliert werden. Dafür werden alle bekannten Wörter (Stoppworte, Indexterme) entfernt und die übrigen Worte separat in einer Tabelle in der Datenbank abgelegt. Zur Erschließung der unbekannten Worte wird diese Wortliste zum einen nach der Häufigkeit und zum anderen nach dem Alphabet geordnet ausgegeben.
Es wird davon ausgegangen, dass häufig verwendete und unbekannte Worte für die Domäne wichtig sind und ihre Modellierung einen hohen Informationsgewinn zur Folge hat. Die Ordnung nach dem Alphabet soll den Modellierer dabei unterstützen syntaktisch ähnliche Worte zum häufigsten Wort zu modellieren, um eine neue IE mit ihren Ausprägungen so vollständig wie möglich zu erfassen. Weiterhin wird dem Modellierer der Webservice zum Projekt Deutscher Wortschatz zur Verfügung gestellt, um die Flexionen zu einer Grundform hinzuzufügen.
Als weiteres unterstützendes Element werden die Textabschnitte, in denen ein unbekanntes Wort vorkommt, angezeigt, so dass Disambiguitäten durch den Kontext aufgelöst werden können.
Wie bereits in Abschnitt 5.5.4 beschrieben, können auch Rechtschreibfehler korrigiert werden. Dabei kann der Nutzer entscheiden, ob das unbekannte Wort einen Rechtschreibfehler enthält, wie dieser erfasst werden soll (vgl. Abschnitt 3.2.4 und Abschnitt 6.4)
Falls das unbekannte Wort zu einem bereits vorhandenem IE zugeordnet werden kann, muss das entsprechende IE ausgewählt werden, um die Beziehung zwischen den Worten zu modellieren.
Weiterhin ist es möglich, dass eine komplett neue IE mit ihren Ausprägungen modelliert werden muss. Dafür wird das IE entweder einer bekannten IE-Klasse zugeordnet oder es kann eine neue IE-Klasse erstellt werden.
Neben dem eigentlichen IE können die Flexionen aus dem Projekt Deutscher Wortschatz hinzugefügt und über die alphabetisch geordnete Liste syntaktisch ähnlicher Worte gefunden und assoziiert werden. Dabei muss darauf geachtet werden, dass der Indexterm, der das IE repräsentieren soll in seiner Grundform erfasst wird und seine Flexionen und Synonyme als Ausprägungen modelliert werden.

6 Implementierung

Dieses Kapitel stellt Ausschnitte der Implementierung vor, deren Konzepte in den vorangegangenen Abschnitten erläutert wurden. Das entwickelte Werkzeug richtet sich an erfahrene Modellierer, welche die Daten einer Datenbank für das TFBS aufbereiten wollen.
Zu Beginn wird die Vokabularanreicherung kurz erläutert und im zweiten Teil wird die grafische Benutzeroberfläche vorgestellt.

Die Anwendung wurde in Java 5[16] implementiert und genutzte Datenbank ist PostgreSQL[17] in der Version 8.2

6.1 Erfassung des GermaNet-Korpus

Für den schnellen und effizienten Zugriff auf den GermaNet-Korpus wurde dieser in das Datenmodell des Indexlexikons (Abschnitt 3.1.3) integriert. Wie bereits in Abschnitt 5.5.1 beschrieben liegt GermaNet in XML-Dateien vor. Mit einem SAXBuilder wird jedes Dokument in den Speicher geladen und die synsets werden folgendermaßen ausgelesen:

```
private void createStatements(org.jdom.Element rootNode) {
        // first create a list with all synsets
        List synsets = rootNode.getChildren();
        // In each synset the children <lexUnit> are nested and a child of
        // lexUnit is orthForm and orthForm contains the term searched for
        for (Iterator iter = synsets.iterator(); iter.hasNext();) {
                org.jdom.Element synset = (org.jdom.Element) iter.next();
                String klasse = synset.getAttributeValue("lexGroup");
                String ie = synset.getChild("lexUnit").getChild("orthForm").getText();

                List lexUnits = synset.getChildren("lexUnit");

                for (Iterator iterator = lexUnits.iterator(); iterator.hasNext();) {
                        org.jdom.Element lexUnit = (org.jdom.Element) iterator.next();
                        String auspraegung = lexUnit.getChild("orthForm").getText();
                        writeStatement(klasse, ie, auspraegung);
                }
        }
}
```

Für jedes Synset werden so das Wort an sich (orthForm in lexUnit) und falls vorhanden seine Synonyme (for-Schleife) ausgelesen und in ein SQL-Statement geschrieben, welches die Daten in die Datenbank einfügt.

6.2 Webservice-Anbindung zum Projekt Deutscher Wortschatz

Als weiteres Beispiel der Anreicherung des Vokabulars soll die Anbindung an das Projekt Deutscher Wortschatz beschrieben werden. Der Webseite des Projektes

[16] http://java.sun.com/
[17] http://www.postgresql.org/

Deutscher Wortschatz stellt Java-Clients zur Verfügung, über die auf den Webservice zugegriffen werden kann.
Der Client wird folgendermaßen instanziiert:

```
WordformsClient client = new WordformsClient();
        client.setUsername("anonymous");
        client.setPassword("anonymous");
        client.setCorpus("de");

        client.addParameter("Word", suchwort);
        client.addParameter("Limit",limit);
        client.execute();
```

Neben Username und Kennwort muss der angesprochene Korpus (in diesem Fall *de* für deutsch) übergeben werden. Jede Grundform aus GermaNet wird als *suchwort* an den Webservice übergeben. Das Limit gibt die maximale Anzahl der zurück gelieferten Worte an. Wird ein Limit über den vorhandenen Wortformen angegeben, so werden alle gefundenen Flexionen in einem Stringarray zurückgeliefert. Das Ergebnis wird wiederum in SQL-Statements geschrieben und in der Datenbank gespeichert.

6.3 Fallformat erstellen

Im Abschnitt 5.1 wurde die Bedeutung des Fallformates hervorgehoben, so dass hier die dazu passende Oberfläche präsentiert wird.

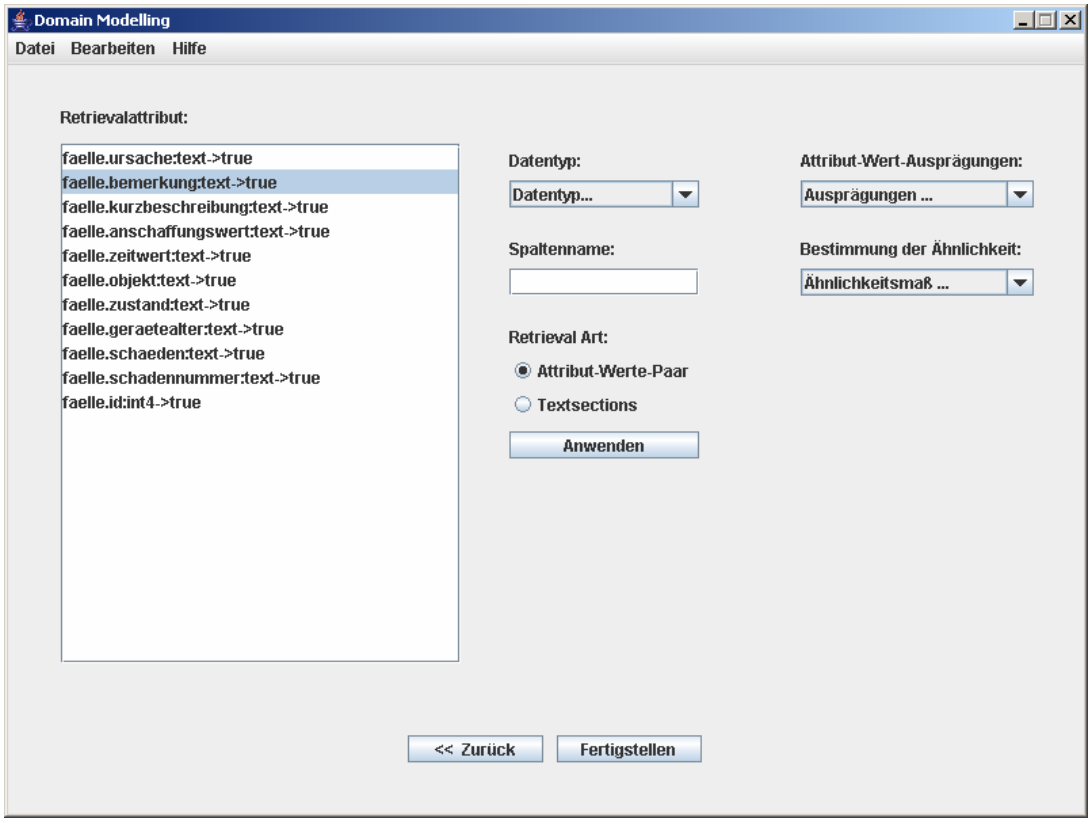


Abbildung 20: Oberfläche zur Beschreibung eines neuen Fallformates

Abbildung 20 zeigt den letzten von drei Schritten zur Beschreibung eines neuen Fallformates. Im ersten Schritt wir die Verbindung zu einer Datenbank hergestellt und die Metadaten zu den enthaltenen Tabellen extrahiert. Nachdem der Benutzer die zu betrachtenden Tabellen ausgewählt hat, bestimmt er die Retrieval- und Informationsattribute um im letzten Schritt (vgl. Abbildung 20) für die Retrievalattribute anzugeben, wie das Retrieval durchgeführt werden soll.

6.4 Dialog zur Erfassung neuer IEs

Als weiterer Dialog der Benutzeroberfläche wird in Abbildung 21 die Behandlung unbekannter IEs im System gezeigt. Der Dialog wird von oben nach unten chronologisch abgearbeitet. Im oberen Drittel wird das Fallformat eingelesen und die unbekannten Worte können ermittelt werden.
Im mittleren Drittel werden die für das System unbekannten Worte gezeigt und für die Erfassung relevanten Informationen dargestellt.
Das untere Drittel gibt die Möglichkeiten an, wie unbekannte Worte in das Vokabular aufgenommen werden können.

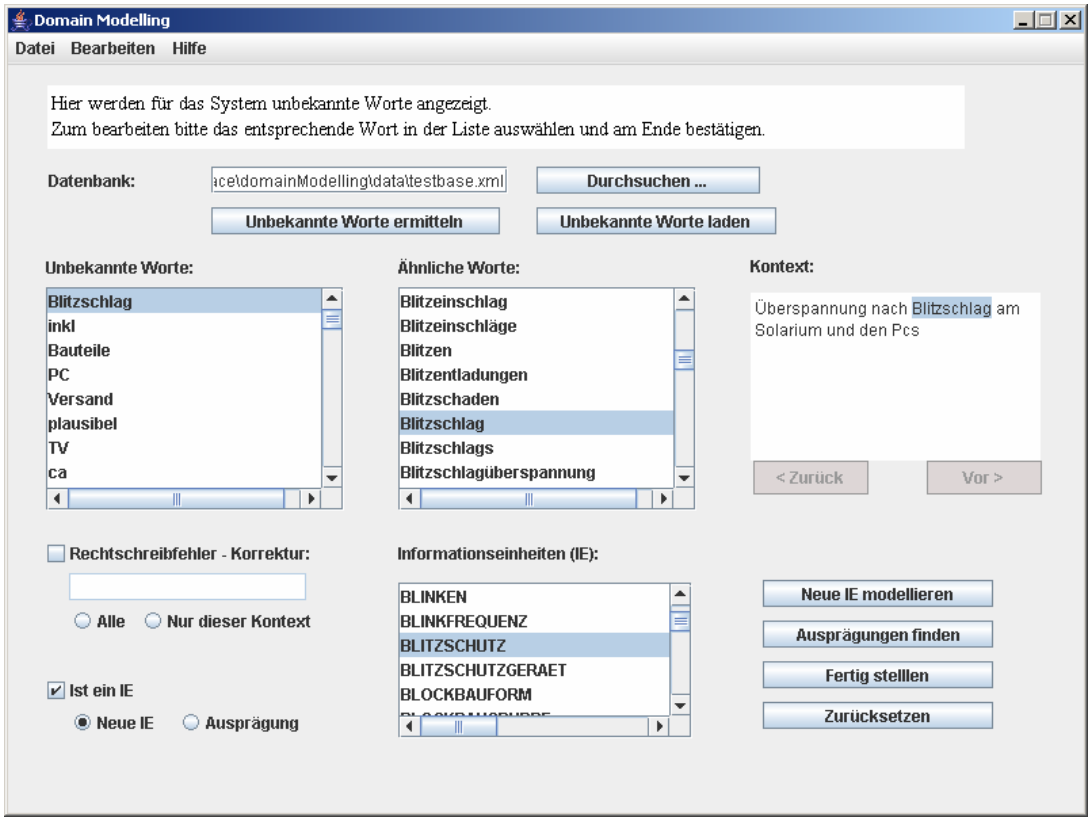


Abbildung 21: Oberfläche zur Erfassung neuer IEs

Als Datenquelle für die zu analysierende Datenbank erwartet das System eine XML-Datei, die dem in Abschnitt 5.2 definierten Fallformat entspricht. Eine gültige Datenquelle ist im Anhang 9.2 aufgeführt.
Falls für das angegebene Fallformat bereits eine Analyse vorgenommen wurde, kann der Nutzer entscheiden, ob er die Analyse erneut durchführt oder die bereits

vorhandenen unbekannten Worte weiter bearbeiten möchte. Wird ein Fallformat zum ersten Mal genutzt, muss Analyse des Korpus durchgeführt werden, um weiterarbeiten zu können.
Der mittlere Bereich enthält zwei Listboxen und ein Textfeld. In der linken Listbox werden die unbekannten Worte nach ihrer Häufigkeit angeordnet angezeigt. Bei der Auswahl eines Elementes aus dieser Liste wird der gewählte Eintrag in der mittleren Listbox (*Ähnliche Worte*) markiert. Dadurch werden dem Modellierer syntaktisch ähnliche Worte gezeigt. Falls sich darunter Flexionen oder Synonyme für das ausgewählte Wort befinden, können diese ebenfalls modelliert und im gleichen Arbeitsschritt zu einer IE zugeordnet werden. Das Textfeld auf der rechten Seite (Kontext) zeigt an, in welchem Zusammenhang das Wort in der Datenbank steht. Sind mehrere Einträge mit dem unbekannten Wort vorhanden, kann durch diese navigiert werden.
Das untere Drittel behandelt die Erfassung der unbekannten Worte als IE. Auf der linken Seite wird unterschieden, ob es sich um Rechtschreibfehler handelt oder nicht. Rechtschreibfehler können dann zum einen korrigiert und zum anderen zu einer IE zugewiesen werden (vgl. Abschnitt 5.5.4).
Soll das unbekannte Wort als Indexterm aufgenommen werden, kann in der Listbox (*Informationseinheiten*) auf alle bereits vorhandenen IEs zur Orientierung, welche IEs bereits vorhanden sind, zugegriffen werden. Auf der rechten Seite können zwei weitere Dialoge aufgerufen werden: Ein Dialog zur Erfassung einer neuen IE (*Neue IE modellieren*) anhand der bis dahin vorhandenen Informationen und ein zweiter Dialog zum Abrufen aller Flexionen der neu modellierten Ausprägungen (*Ausprägungen finden*), wie in Abschnitt 5.5.2. beschrieben. Um die Modellierung abzuschließen wird durch den Button „*Fertig stellen*“ die Modellierung des unbekannten Wortes abgeschlossen und der Modellierer kann mit dem nächsten Wort fortfahren. Der darunter liegende Button „*Zurücksetzen*“ löscht alle Eingaben zum aktuell bearbeiteten Wort.

7 Evaluierung

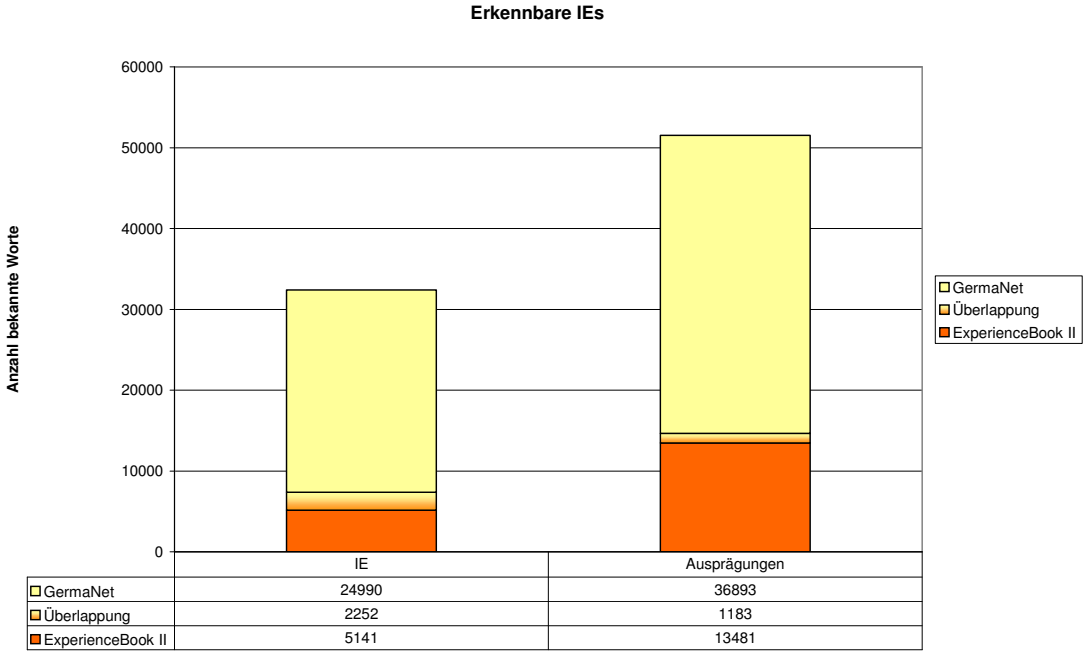
Zur Evaluierung des in der Arbeit vorgestellten Konzeptes wurden Datensätze aus der Versicherungsdomäne herangezogen. Die Datenbank besteht aus 9640 Datensätzen á 12 Attributen (jeder Datensatz ist wie in Abbildung 18 gezeigt aufgebaut).

7.1 Erweiterung des Indexvokabulars

Die nachfolgenden Auswertungen veranschaulichen die Entwicklung des Indexvokabulars durch die Hinzunahme der in Abschnitt 5.5 vorgestellten Datenquellen und zeigen dadurch den Mehrwert, welcher durch die Anreicherung des Indexvokabulars entsteht.

7.1.1 Überlappungen von IEs zwischen ExperienceBook II und GermaNet

Als potentielle IEs dienen zum einen die aus dem ExperienceBook II stammenden Indexterme als Referenzwert. Hinzugefügt wurden Begriffe (Substantive) aus GermaNet.



	IE	Ausprägungen
GermaNet	24990	36893
Überlappung	2252	1183
ExperienceBook II	5141	13481

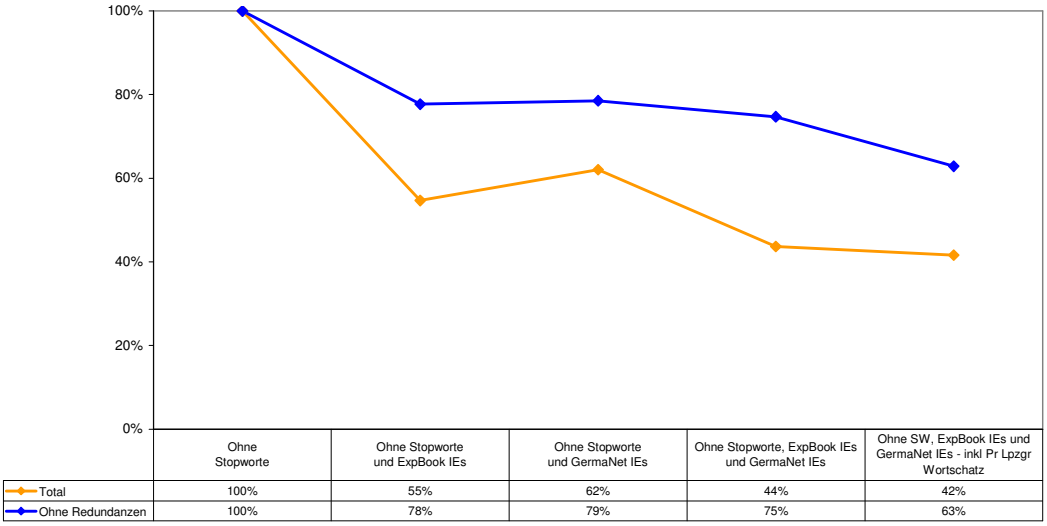
Abbildung 22: Überlappungen der IEs und Ausprägungen

Das ExperienceBook II wurde für eine technische Domäne (Systemadministration) konzipiert, wogegen GermaNet allgemeinsprachliche Worte enthält, so dass die Überschneidungen der Wortschätze gering sind. Somit kann davon ausgegangen werden, dass mehr Worte erkannt und als IE identifiziert werden können. Allerdings zeigt dies auch, dass in einer speziellen Domäne Begriffe genutzt werden, die aus

GermaNet nicht erschlossen werden können. Aus diesem Grund ist die unterstützte Nachmodellierung von IEs wie in Abschnitt 5.9 vorgestellt, notwendig.
Die linke Seite der Abbildung zeigt die Indexterme, die als IE im Text erkannt werden können. Wie zuvor erwähnt, sind die IEs nur in ihren Grundformen erfasst. Flexionen und Synonyme finden sich in der rechten Säule (Ausprägungen) wieder. Der Bereich zwischen GermaNet und ExperienceBook II zeigt die anteilige Überlappung zwischen den beiden Vokabularen auf.

7.1.2 Abdeckung des vorliegenden Korpus

Nach der Anreicherung des Indexvokabulars durch die IEs des Experience Book II, GermaNet und dem Projekt Deutscher Wortschatz zeigt Abbildung 23 die prozentuale Abdeckung der Daten durch das Vokabular. Als Referenzwert für die anteilige Berechnung wurde der Korpus abzüglich der Stoppworte gewählt.



	Ohne Stopworte	Ohne Stopworte und ExpBook IEs	Ohne Stopworte und GermaNet IEs	Ohne Stopworte, ExpBook IEs und GermaNet IEs	Ohne SW, ExpBook IEs und GermaNet IEs - inkl Pr Lpzgr Wortschatz
Total	100%	55%	62%	44%	42%
Ohne Redundanzen	100%	78%	79%	75%	63%

Abbildung 23: Anteil der unbekannten Worte nach den Erweiterungen

Die blaue Linie zeigt die prozentuale Verringerung der unbekannten IEs ohne Berücksichtigung der Redundanzen. Die orangefarbene Linie zeigt wie die Gesamtanzahl der unbekannten IEs durch die Vobabularanreicherung abnimmt.

Abbildung 24 und Abbildung 25 zeigen die Verteilung der unbekannten Worte über die einzelnen Attribute. Abbildung 24 betrachtet die Gesamtanzahl der unbekannten Worte inklusive der Redundanzen, wogegen Abbildung 25 die absoluten unbekannten Worte vergleicht. Um die Größenverhältnisse (Anzahl der unbekannten Worte) zu verdeutlichen, wurde in Abbildung 25 die Ordinate auf 100.000 Worte skaliert.

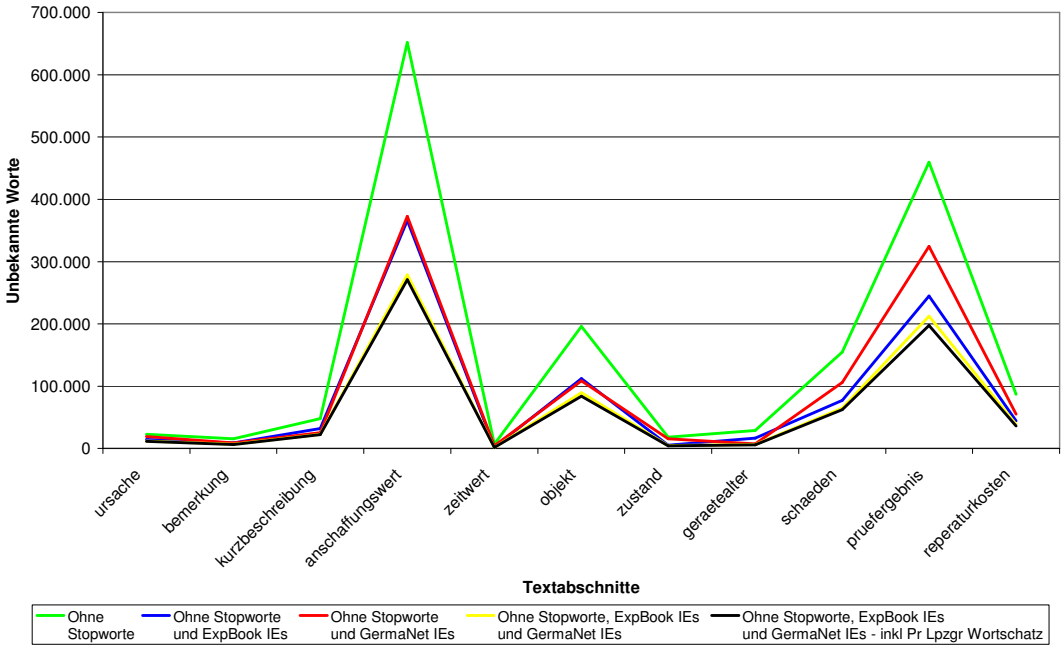


Abbildung 24: Verteilung der unbekannten IEs über den getesteten Korpus (absolut)

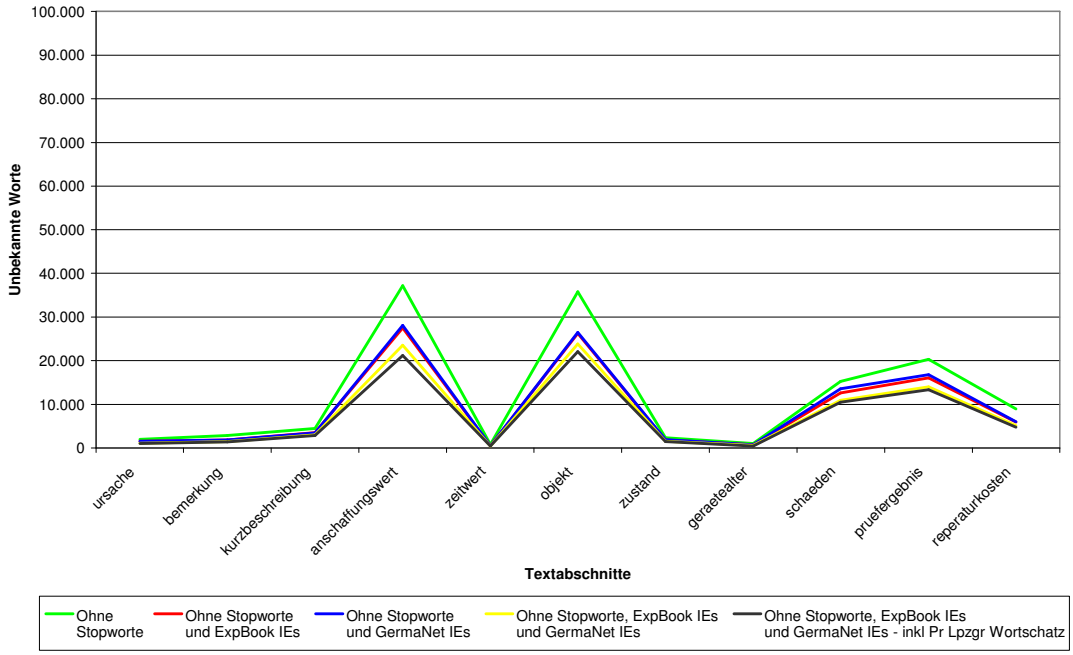


Abbildung 25: Verteilung der unbekannten IEs über den getesteten Korpus (Ohne Redundanzen)

Aus Abbildung 24 wird ersichtlich, dass der Anschaffungswert die meisten unbekannten Worte beinhaltet. Bei genauerer Betrachtung zeigt sich, dass es sich dabei um Angaben wie *inkl* und *MwSt* handelt. Trotz der Entfernung der Redundanzen bleiben weiterhin viele Worte in dieser Passage unerkannt. Dabei handelt es sich vor allem um Produktnamen, die zusammen mit dem Anschaffungswert in einer Textpassage erfasst sind und vom Vokabular nicht abgedeckt werden.

Die allgemeinen textuellen Beschreibungen in *Bemerkung*, *Kurzbeschreibung* und *Schaeden* sind vom Vokabular sehr gut abgebildet. Die auffällig vielen *unbekannten Worte* in *Objekte* erklären sich dadurch, dass die einzelnen Produkte, die mit ihren Produktnamen erfasst wurden darin beschrieben wurden.
Diese Ergebnisse zeigen, dass für Anwendungen die Produktbeschreibungen enthalten auch die Produktdatenbanken zur Verfügung stehen sollten, um Texte besser erfassen zu können.
Abbildung 25 zeigt wie viele syntaktisch unbekannte Worte es in den einzelnen Textabschnitten gibt. Beachtlich ist, dass auch nach der Entfernung redundanter Wort die Attribute *anschaffungswert* und *zeitwert* deutlich mehr unbekannte Worte beinhalten als alle anderen. Bei genauerer Betrachtung des analysierten Korpus zeigt sich, dass in beiden Attributpassagen hauptsächlich Produktnamen unbekannt sind.

7.2 Verteilung der unbekannten Worte

In diesem Abschnitt soll die Verteilung der unbekannten Worte untersucht werden. Wie Abbildung 26 zeigt gibt es verhältnismäßig wenige unbekannte Worte, die sehr häufig auftauchen. Somit hat deren Modellierung einen großen Einfluss auf die Nachmodellierung. Durch die große Anzahl von unbekannten Produktnamen kommen viele Worte in dieser Übersicht nur einmal vor.

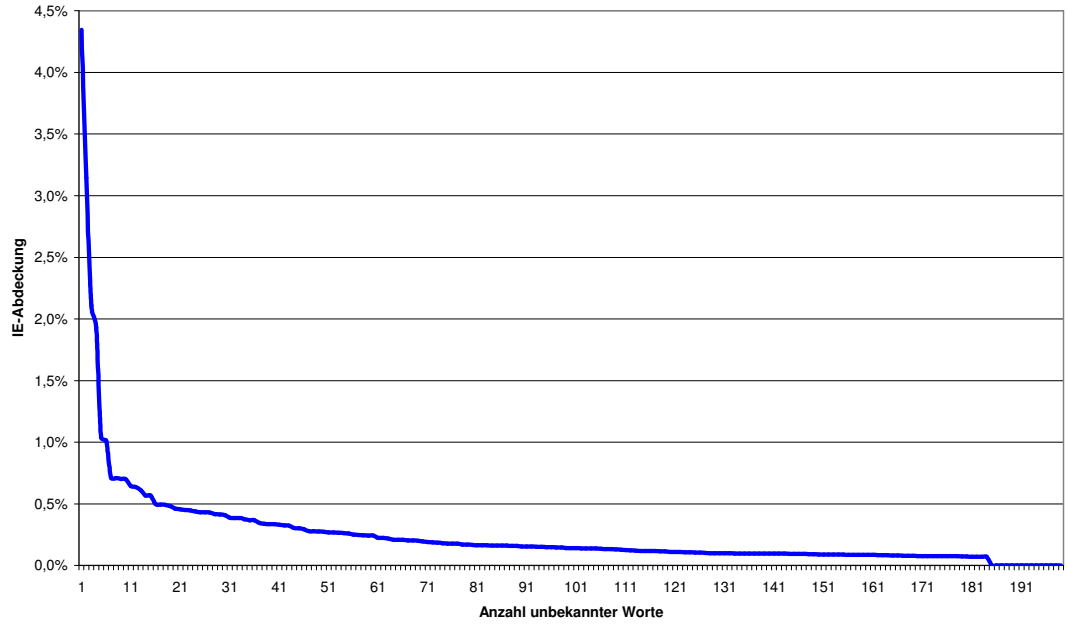


Abbildung 26: Verteilung der unbekannten Worte

Aufgrund der in Abbildung 26 auftretenden Verteilung zeigt Abbildung 27 welchen Effekt die Nachmodellierung der unbekannten Worte hat. Allein durch die zehn häufigsten als unbekannt gekennzeichneten Worte können 16% (113.417 Worte) der *unbekannten Worte* erkannt werden. Um die *unbekannten Worte* im System zu

halbieren müssen 200 Worte nachmodelliert werden. Damit werden auch die meisten domänenspezifischen Worte abgedeckt.
Wie bereits in Abschnitt 7.1.2 beschrieben, sind ein Großteil der unbekannten Worte Eigennamen von Produkten. Diese können durch die Einbindung von Produktkatalogen (vgl. 5.5.5.1) oder Anwendung der Named Entity Recognition erfassbar gemacht werden.

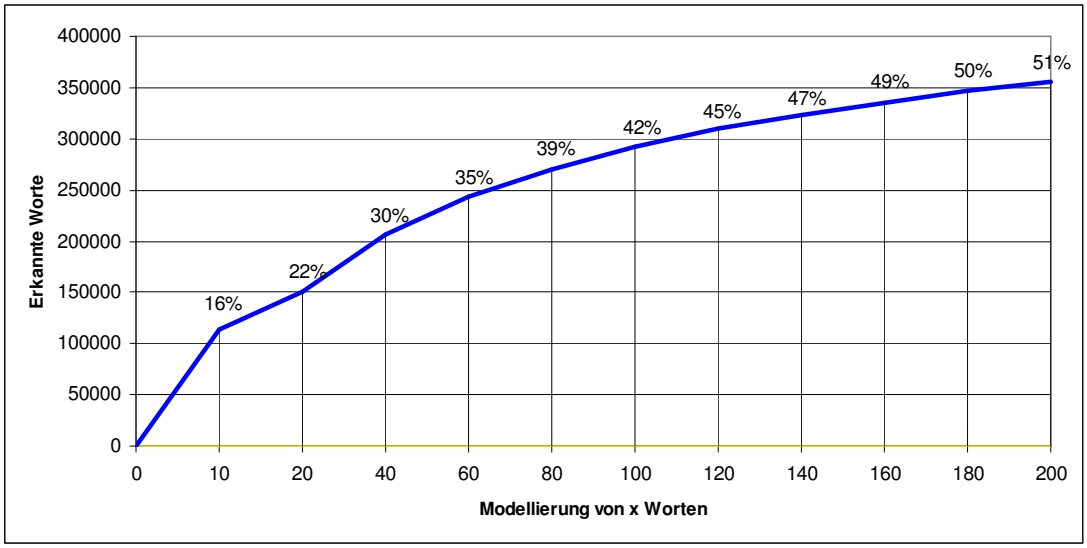


Abbildung 27: Verbesserung der IE-Erkennung

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Die vorliegende Arbeit zeigt einen Ansatz zur Domänenmodellierung für das Textuelle Fallbasierte Schließen. Es wurde die Bedeutung des Domänenmodells für die Erfassung einer Domäne gezeigt. Im Domänenmodell für ein TFBS-System ist zum einen die Struktur der zugrunde liegenden Dokumente und zum anderen das Vokabular zur Erkennung der darin verwendeten Worte enthalten. Die große Herausforderung bei der Erschließung von deutschen Freitextdokumenten besteht darin, die Flexionen und Kompositionen zu erkennen. Die Flexionen der bekannten Worte werden vollständig durch die Anbindung an den Webservice des Projektes deutscher Wortschatz abgedeckt und Kompositionen werden im Rahmen des Grundwortschatzes erkannt. Die Auflösung von Ambiguitäten stellt allerdings weiterhin ein Problem dar, so lange keine kontrollierte Sprache in den Texten verwendet wurde.
Neben der Abbildung der Fallstruktur wurde im Rahmen dieser Arbeit vorgestellt, wie eine unbekannte Domäne erschlossen werden kann. Es wurden heterogene Informationsquellen integriert, so dass ein Modellierer lediglich domänenspezifisches Wissen von Hand erfassen muss und dabei unterstützt wird. Die neu erfassten Worte werden in einer Datenbank zusammen mit dem allgemeinsprachlichen Vokabular abgelegt. Dazu sind die domänenspezifischen Worte durch eine eigene Klasse gekennzeichnet.

Zur Unterstützung der Arbeit eines Modellierers wird eine Oberfläche mit Anbindungen zum Vokabular wie auch zum Webservice des Projektes Deutscher Wortschatz bereitgestellt. Zudem wird der Nutzer dabei unterstützt, die Verbindung zu seiner Datenbank aufzubauen und ein Fallformat entsprechend der in der Datenbank vorhandenen Daten zu definieren. Das Fallformat wird in XML erfasst und kann auch beim TFBS mit CRNs weiterverwendet werden. Als Ergebnis der Modellierung stehen für einen unbekannten Anwendungsbereich die Daten zum Aufbau eines CRNs bereit.

Das Konzept wurde anhand einer Datenbank mit Versicherungsschadensfällen angewandt und evaluiert. Dabei zeigte sich, dass zum einen die Worte des deutschen Grundwortschatzes sehr gut abgedeckt sind, jedoch Produktnamen und spezielle Abkürzungen nachmodelliert werden müssen. Dafür wurden Ansätze wie beispielsweise eCl@ss gezeigt, dessen Einbindung vor allem für Produkt- und Dienstleistungsanwendungen nützlich ist.

8.2 Verwandte Arbeiten

8.2.1 Fallbasiertes Schließen

Die in [Lenz1999] vorgestellten Anwendungen nutzten jeweils ein für die Anwendung speziell erstelltes Domänenmodell. Dieses wurde im Weiteren auch für die Erstellung eines Ähnlichkeitsmodells genutzt. Je nach Domäne wurde das Model entweder aus den bekannten, vorliegenden Fällen von Modellierern von Hand beschrieben oder für komplexere Zusammenhänge durch das Wissen von Experten ergänzt.

In [Pfuhl2003] wurde ein Ansatz vorgestellt, in dem eine „strukturierte Suche“ – ein Retrieval in einem Strukturierten FBS-System – in unstrukturierten Quellen durchgeführt werden sollte. Dafür wurden Wirtschaftsnachrichten vorverarbeitet (mit Natural Language Processing (NLP) und/oder Tagging), indiziert und in relationalen Datenbanken abgelegt. Die Wirtschaftnachrichten sind einheitlich aufgebaut (Datum, Uhrzeit, Kategorie, Wertpapiernummer, Überschrift und Nachrichtentext) und die Passagen in der Überschrift und im Nachrichtentext enthalten Freitext.
Die indexierten Dokumente werden als Attribut-Werte-Paare mit dynamischer Länge verwaltet und die Metadaten werden genutzt, um die Informationen den entsprechenden Attributen im Wissensmodell abzulegen. Die für das Retrieval notwendigen Daten wurden in einer relationalen Datenbank abgelegt (ähnliche wie auch in [Hilbig2003] beschrieben) [vgl. Pfuhl2003].
Jene Arbeit befasst sich mit einer konkreten Domäne und geht auch davon aus, dass die zu indexierenden Dokumente den gleichen Aufbau haben, daher lag das Augenmerk nicht in der Erfassung einer beliebigen Domäne. Außerdem wurde das Vokabular von Hand aufgebaut, indem Worte, die mehr als zwanzig Mal in den 7582 Dokumenten vorkamen, einzeln modelliert wurden. Zur Erkennung von Flexionen wurde die Flexionsanalyse nach Lezius [vgl. Lezius1994] genutzt, die ähnlich einem Stemming-Algorithmus auf der Stammformreduktion basiert und im Deutschen weniger gute Resultate liefert wie im Englischen. Zusammenfassend stellt der Autor heraus, dass die Suche in den Wirtschaftsnachrichten gut funktioniert und eine Anwendung der Vorgehensweise auf beschränkte Anwendungsdomänen möglich ist. Ähnlich wie das Ergebnis dieser Arbeit wird zudem festgestellt, dass ohne die Verwendung eines festgeschriebenen Vokabulars die semantische Erfassung nahezu unmöglich ist.

8.2.2 Worterkennung

Auf dem Bereich der Worterkennung kann neben der Möglichkeit ein sehr umfangreiches Vokabular aufzubauen auch das Stemming genutzt werden, um Worte zu erkennen. Stemming erfasst lediglich die syntaktische Ähnlichkeit von Worten. Einer der bekanntesten und auch auf das Deutsche anwendbaren Algorithmen ist der Stemming-Algorithmus nach Porter [Porter1980]. Da durch die Veränderung des

Wortstammes bei Flexionen im Deutschen das Stemming oftmals nicht erfolgreich ist, werden zusätzlich statistische Verfahren zur Worterkennung angewandt.
Soll den erkannten Worten eine Bedeutung zugewiesen werden, kann dies durch Tagger oder NER erfolgen. Beide Verfahren benötigen meist einen Trainingskorpus und müssen von Hand annotiert werden. Für allgemeinsprachliche Domänen werden meist so genannte *language models* mitgeliefert, wogegen für fachspezifische Domänen das *language model* teilweise oder vollständig von Hand erstellt werden muss.

Eine bereits im TFBS angewandte Methode zur Flexionserkennung ist der so genannte TreeTagger, der den Wörtern eines Textes grammatikalische Wortklassen sowie ihren Kasus, Numerus und Genus zuweist (als Tag). Der TreeTagger basiert auf einem binären Entscheidungsbaum, der den Text mit Hilfe eines modifizierten ID3-Algorithmus in Trigramme[18] aufteilt. Beispielsweise würde aus dem Satz „*Gerät kann beim Anspruchsteller besichtigt werden*“:

> (Ger, erä, rät, bei, eim, Ans, nsp, spr, pru, ruc, uch, chs, hst, ste, tel, ell, lle, ler, bes, esi, sic, ich, cht, hti, tig, igt, wer, erd, rde, den)

Durch diese Aufteilung haben Endungen und Umlauten nicht mehr einen so hohen Einfluss auf die syntaktische Ähnlichkeit zwischen Worten und durch ein Lexikon mit a-priori Wahrscheinlichkeiten können Worte in verschiedenen Sprachen zuverlässig erkannt werden. [vgl. LenzEtAl1998]. Allerdings werden dadurch nur syntaktisch ähnliche Worte, aber keine Synonyme erkannt oder Mehrdeutigkeiten aufgelöst.

8.3 Ausblick

Abschließend soll ein Ausblick gegeben werden, der die Domänenmodellierung besonders im TFBS in Zukunft beeinflussen kann.
Im Moment werden keine Produktnamen vom Vokabular erfasst, allerdings ist die Einbindung einer Produktdatenbank durch die Erfassung des eCl@ss-Datenbestandes bereits vorbereitet, so dass in einem nächsten Schritt für eine konkrete Anwendung darauf zurückgegriffen werden kann.
Weiterhin können in einer erweiterten Version von GermaNet die Relationen zwischen den Begriffen abgeleitet werden. Dieses Wissen kann für die Bestimmung von Ähnlichkeiten genutzt werden und das Retrieval verbessern. Ähnliche Informationen zu Worten stellt aber auch das Projekt Deutscher Wortschatz zur Verfügung, so dass beispielsweise bei der Anfrage die gefundenen IEs mit Synonymen angereichert werden können, um beim Retrieval im CRN mehrere IEs zu markieren und somit mehrere Fälle während der Propagierung zu betrachten. Die erweiterte Version von GermaNet bietet zudem die Möglichkeit Meronyme, Hyperonyme und Hyponyme zu erkennen, so dass im Weiteren „is-part-of“ oder „has-a“ Beziehungen modelliert werden können.

[18] Trigramm = Sequenz aus drei Buchstaben, in die ein Text aufgespaltet wird. Beispiel:

Ebenfalls kann der Aufbau des Vokabulars im Deutschen analog für das Englische vorgenommen werden, da WordNet, das semantische Netz der englischen Sprache, frei verfügbar ist und auch der Leipziger Wortschatz seine Analysen für das Englische[19] bereitstellt. Damit könnte zum einen ein multilinguales TFBS-System erstellt werden, wie auch ein monolinguales System welches englische Dokumente verarbeitet.

[19] http://corpora.informatik.uni-leipzig.de/?dict=en

9 Anhang

9.1 IE - Klassen

Die nachfolgende Tabelle zeigt die im System abgebildeten IE-Klassen auf.

classname	classnumber	classtype	classweight
Stopwords	0	SW	0,00
Begriffe aus der IT-Welt	1	TE	1,00
allgemeinsprachliche Begriffe	2	TE	1,00
Attributdefinitionen	3	AD	0,00
Attribut-Werte-Paare	4	AV	1,00
GASPL2-spezifische Begriffe	5	TE	1,00
Fachbegriffe der Gasdomäne	6	TE	1,00
SIMATIC	14	TE	1,00
TOPIC	15	AV	1,00
MERLIN	17	TE	1,00
MODULE	18	AV	1,00
RELEASE	19	AV	1,00
PREFIX	20	AV	1,00
REF_DOC	21	AV	0,70
REF_PREFIX	22	AV	0,70
Artefakt	51	GN	0,75
Attribut	52	GN	0,75
Besitz	53	GN	0,75
Form	54	GN	0,75
Gefuehl	55	GN	0,75
Geschehen	56	GN	0,75
Gruppe	57	GN	0,75
Koerper	58	GN	0,75
Kognition	59	GN	0,75
Kommunikation	61	GN	0,75
Menge	62	GN	0,75
Mensch	63	GN	0,75
Motiv	64	GN	0,75
natGegenstand	65	GN	0,75
natPhaenomen	66	GN	0,75
Ort	67	GN	0,75
Pflanze	68	GN	0,75
Relation	69	GN	0,75
Substanz	70	GN	0,75
Tier	71	GN	0,75
Tops	72	GN	0,75
Zeit	73	GN	0,75

Abbildung 28: IE-Klassen der Anwendung

Die Spalte *classname* ist der textuelle Name der Klasse, wogegen *classnumber* die eindeutige Identifizierung der Klasse ist und zur Referenzierung wird im System genutzt wird.

Die Spalte *classtype* bezeichnet die Art der Klasse. Dabei wird folgende Unterscheidung getroffen:

SW für die Stoppworte

TE für Textuelle IEs

AD für Attributdefinitionen

AV für Attribut-Wert-Paare

GN für Textuelle IEs, die aus dem GermaNet-Korpus stammen

Die vierte Spalte (*classweight*) gibt die Gewichtung an, die genutzt wird, um die lokalen Ähnlichkeiten zwischen IEs zu bestimmen.

9.2 Beispiel eines Fallformates

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE database SYSTEM "dbStructure.dtd">
<database>
	<connectiondata>
		...
	</connectiondata>
	<tables>
		<table>
			<tblname>faelle</tblname>
				<column>
					<clmnname>ursache</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>bemerkung</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>kurzbeschreibung</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>anschaffungswert</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>AV</kindOfRetrieval>
				</column>
				<column>
					<clmnname>zeitwert</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>AV</kindOfRetrieval>
				</column>


				<column>
					<clmnname>objekt</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>zustand</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>geraetealter</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>AV</kindOfRetrieval>
				</column>
				<column>
					<clmnname>schaeden</clmnname>
					<datatype>text</datatype>
					<isRetrieval>true</isRetrieval>
					<kindOfRetrieval>Text</kindOfRetrieval>
				</column>
				<column>
					<clmnname>pruefergebnis</clmnname>
					<datatype>text</datatype>
					<isRetrieval>false</isRetrieval>
					<kindOfRetrieval></kindOfRetrieval>
				</column>
				<column>
					<clmnname>reparaturkosten</clmnname>
					<datatype>text</datatype>
					<isRetrieval>false</isRetrieval>
					<kindOfRetrieval></kindOfRetrieval>
				</column>
		</table>
	</tables>
</database>
```

Abbildung 29: Instanz eines Fallformats

9.3 GermaNet

```
<synset id="nArtefakt.2011" lexGroup="Artefakt" wordClass="nomen">
      <lexUnit Eigenname="nein" artificial="nein"
      id="nArtefakt.2011.Haus" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Haus</orthForm>
      </lexUnit>
</synset>
<synset id="nBesitz.259" lexGroup="" wordClass="">
      <lexUnit Eigenname="nein" artificial="nein"
      id="nBesitz.259.Haus" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Haus</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nBesitz.259.Hausbesitz" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Hausbesitz</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nBesitz.259.Hauseigentum" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Hauseigentum</orthForm>
      </lexUnit>
</synset>
<synset id="nGruppe.840" lexGroup="" wordClass="">
      <lexUnit Eigenname="nein" artificial="nein"
      id="nGruppe.840.Haus" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Haus</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nGruppe.840.Geschlecht" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Geschlecht</orthForm>
    </lexUnit>
</synset>
<synset id="nKommunikation.177" lexGroup="Kommunikation"
wordClass="nomen">
      <lexUnit Eigenname="nein" artificial="nein"
      id="nKommunikation.177.Sternzeichen" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Sternzeichen</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nKommunikation.177.Sternbild" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Sternbild</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nKommunikation.177.Tierkreiszeichen" orthVar="nein"
      sense="1" stilMarkierung="nein">
            <orthForm>Tierkreiszeichen</orthForm>
      </lexUnit>
      <lexUnit Eigenname="nein" artificial="nein"
      id="nKommunikation.177.Haus" orthVar="nein" sense="1"
      stilMarkierung="nein">
            <orthForm>Haus</orthForm>
      </lexUnit>
</synset>
```

Abbildung 30: Grenzen von GermaNet

Abbildung 30 zeigt auszugsweise die Struktur der XML-Dateien aus GermaNet, in denen das Wort Haus enthalten ist. Die verschiedenen Bedeutungen finden sich in

- nomen.Artefakt.xml,
- nomen.Besitz.xml,
- nomen.Gruppe.xml und
- nomen.Kommunikation.xml

wieder. Damit wird deutlich, dass es bei der Nutzung von GermaNet dazu kommen kann, dass ein falsches Synonym zurückgeliefert werden kann. Um dies zu verhindern, kann die Klasse der umliegenden Worte eventuell zur Auflösung der Ambiguität führen.

9.4 Abbildungen

9.5 Definitionen

Literaturverzeichnis

[Aamodt1994] Aamodt A., Plaza E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches, S. 39-59 In AICom - Artificial Intelligence Communications, Vol. 7: 1, IOS Press, 1994.

[Adriani2000] Adriani, M.: Ambiguity Problem in Multilingual Information Retrieval. In: Cross-Language Information Retrieval and Evaluation: Workshop of Cross-Language Evaluation Forum, CLEF 2000, Lisbon, Portugal, 2000.

[AhaBreslow1997] Aha, D., Breslow, L. Refining conversational case libaries. In: Leake, B., Plaza, E. (Hrsg.), Case-Based Reasoning Research and Development (ICCBR-97), Lecture Notes in Artificial Intelligence. Springer, Berlin, Heidelberg, 1997.

[AhaEtAl2001] Aha, D., Weber, R. (Hrsg.). Proceedings of the Workshop on Intelligent Lessons Learned Systems at 17th National Conference on AI (AAAI-00). American Association for Artificial Intelligence, 2000.

[Althoff2001] Althoff, K.-D.: Case-Based Reasoning. S. 549-588 In: S. K. Chang (Hrsg.), Handbook on Software Engineering and Knowledge Engineering. Vol. 1 "Fundamentals", World Scientific, 2001.

[Althoff2005] Althoff, K.-D.: Vorlesung: Grundlagen des Wissensmanagements mit Schwerpunkt fallbasiertem Schließen, Vorlesungsfolien, Universität Hildesheim, 2005: http://www.iis.uni-hildesheim.de/Lehre/Sommersemester2005/VorlesungFBS/ Verifizierungsdatum: 26. März 2007

[AlthoffEtAl2006] Althoff, K.-D., Decker, B., Hanft, A., Mänz, J., Newo, R., Nick, M., Rech, J., Schaaf, M.: Intelligent Information Systems for Knowledge Work(ers). Industrial Conference on Data Mining 2006: S. 539-547, 2006.

[Althoff2006] Althoff, K.-D.: Vorlesung: Verteilte Lernende Systeme, Vorlesungsfolien, Foliensatz 04: Introduction to Multiagent Systems, Universität Hildesheim, 2005: http://www.iis.uni-hildesheim.de/Lehre/Sommersemester2006/VorlesungVLS/Resourcen/VLS-04-6.pdf, Verifizierungsdatum: 29. März 2007.

[Arango1994] Arango, G. “Domain Analysis Methods,” In: Schäfer, W., Pierto-Díaz, R., Matsumoto, M. (Hrsg.), Software Reusability, S.17-49, Ellis Horwood Ltd., New York, USA, 1994.

[Bell2004] Bell, C.: SIMLEX Experience Management für die Simulationsliga des RoboCups, Diplomarbeit, Humboldt-Universität zu Berlin, 2004.

[BergmannEtAl2003] Bergmann, R., Althoff, K.-D., Breen, S., Göker, M., Manago, M., Traphöner, R., Wess, S.. Developing Industrial Case-Based Reasoning Applications. LNAI 1612, Springer Verlag, 2003.

[BergmannSchumacher2000] Bergmann, R., Schumacher, J.: An Efficient Approach to Similarity-Based Retrieval on Top of Relational Databases. EWCBR 2000, S. 273-284, 2000.

[Bergmann2002] Bergmann, R. Experience Management: Foundations, Development Methology, and Internet Based Applications. LNAI 2432, Springer Verlag, 2002.

[Berghofer2003] Roth-Berghofer Thomas R.: Knowledge Maintenance of Case-Based Reasoning Systems – The SIAM Methodology, Dissertation an der Universität Kaiserslautern, verfügbar als DISKI 262, Akademische Verlagsgesellschaft Aka GmbH, Berlin 2003.

[Berard1993] Berard, E.V.: Object-Oriented Domain Analysis. In: Edward V. Berard: Essays on Object-Oriented Software Engineering, Volume 1, Prentice Hall, Englewood Cliffs et al., S. 182-195, 1993.

[BiundoStephan2006] Biundo-Stephan, S.: Vorlesung: Einführung in die Künstliche Intelligenz, Foliensatz 9: Wissensmodellierung, 2006: http://www.informatik.uni-ulm.de/ki/Edu/Vorlesungen/GdKI/WS0506/skript/09-Wissensmodellierung.pdf, Verifizierungsdatum: 29. März 2007.

[CoadYourdon1994] Coad, P., Yourdon, E.: Objektorientierte Analyse, München 1994 (Prentice Hall Verlag).

[EiseneckerCzarnecki2000] Eisenecker, U., Czarnecki, K.: Generative Programming: Methods, Tools and Applications. Addison-Wesley, Boston, 2000.

[FayyadEtAl1996] Fayyad, U. M., Piatetski-Shapiro, G., Smyth, P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data, Comm. of the ACM, 39(11), 1996, S. 27 - 34.

[FriedmanEtAl1977] Friedman, J.H., Bently, J.L., Finkel, R.A:: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Software 3, S. 209-226, 1977.

[FZI2001] FZI, IESE: Domänenmodellierung für Webanwendungen. Technischer Bericht, Projekt APPLICATION2WEB, 2001.

[Glintschert1998] Glintschert, A.: ThemeSearch, Aufbau einer intelligenten, themenspezischen Suchmaschine im WWW. Diplomarbeit, Humboldt-Universität zu Berlin, 1998.

[GökerEtAl1998] Göker, M., Roth-Berghofer, T., Bergmann, R., Pantleon, T., Traphöner, R., Wess, S., Wilke, W.: The development of HOMER: A case-based CAD/CAM help-desk support tool. In Proceedings of the European Workshop on Case-Based Reasoning, EWCBR'98, S. 346 - 357, Springer 1998.

[Hanft2004] Hanft, A.: Collaborative Maintenance in einem FBS System, Diplomarbeit, Humboldt-Universität zu Berlin, 2004.

[Heeg2003] Heeg M., Epple U.: Vergleich und Erweiterung bestehender Klassifikationssysteme und deren Merkmal-Modelle. In: Schnieder, E. (Hrsg.): Entwurf komplexer Automatisierungssysteme (EKA 2003), Braunschweig, 2003.

[Hilbig2003] Hilbig, C.: Retrievalmechanismen für BCRNs in Datenbanken, Diplomarbeit, Humboldt-Universität zu Berlin, 2003.

[Kang1990] Kang, K., Cohen,S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Nr. CMU/SEI-90-TR-21, 1990.

[KitanoShimazu1996] Kitano, H., Schimazu, H.: The Experience-Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control In: Leake, D.B.: Case-Based Reasoning: Experiences, Lessons and Future Directions, Kapitel 13, S. 420ff, Menlo Park, CA, USA: AAAI Press, 1996.

[Krabbel2000] Krabbel, A.: Entwurf, Auswahl und Anpassung aufgabenbezogener Domänensoftware, Dissertation Universität Hamburg, 2000.

[Kruchten1995] Kruchten, P.: The 4+1 View Model of Architecture. IEEE Software, Vol. 12 (No.6), S. 42–50, 1995.

[Kunze1998] Kunze, M.: Das Experience Book – Dokumentation eines fallbasierten Systems zur Unterstützung der Systemadministration, Diplomarbeit, Humboldt-Universität zu Berlin, 1998.

[Kolodner1993] Kolodner, J. L.: Case-Based Reasoning. Morgan Kaufmann Publishers, San Mateo 1993.

[Leake1996] Leake, D. B.: CBR in Context: The Present and Future. In: Leake, D.B.(Hrsg.): Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press/MIT Press, Menlo Park, USA 1996.

[LenzBurkhard1996] Lenz, M., Burkhard, H.D. Case Retrieval Nets: Basic Ideas and Extensions, In: Görz, Hölldobler (Hrsg.), Proceedings of the KI'96, LNAI 1137, S. 227 - 239, Springer, 1996.

[LenzEtAl1998] Lenz, M., Hübner, A., Kunze, M.. Textual CBR, In: Lenz, Bartsch-Spörl, Burkhard, Wess (eds.), Case-Based Reasoning Technology – From Foundations to Applications, LNAI 1400, Springer Verlag, 1998.

[Lenz1999] Lenz, M. Case Retrieval Nets as a Model for Building Flexible Information Systems, Dissertation Humboldt Universität zu Berlin, Berlin 1999.

[Lezius1994] Lezius, W. : Aufbau und Funktionsweise von Marphy, Technocal Report, Fachbereich Psychologie der Universität Paderborn. 2004

[ManagoEtAl1994] Manago, M., Bergmann, R., Wess, S., Traphöner, R.: CASUEL: A Common Case Representation Language – Version 2.0. ESPRIT-Projekt INRECA, Deliverable D1, 1994.

[MaynardEtAl2001] Maynard, D., Tablan, V., Ursu, C., Cunningham, H., Wilks, Y.: Named Entity Recognition from Diverse Text Types, In: Proceedings of the Recent Advances in Natural Language Processing 2001 Conference, S. 257-274, 2001. http://gate.ac.uk/sale/ranlp2001/maynard-etal.pdf, Verifizierungsdatum: 16. April 2007.

[Minor2005] Minor, M., Biermann, B. Learning and Linking Textual Cases. In Brüninghaus, Steffi (Hrsg.): ICCBR 2005 Workshop Proceedings. Chicago, Illinois, August 2005, DePaul University, Chicago, USA, S. 128-137.

[Minor2006] Minor, M.: Erfahrungsmanagement mit fallbasierten Assistenzsystemen, Dissertation Humboldt Universität zu Berlin, Berlin 2006.

[Mierswa2006] Mierswa, I., Wurst, M. ,Klinkenberg, R., Scholz, M., Euler, T. YALE: Rapid prototyping for complex data mining tasks. IN: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), 2006.

[Oellien2002] Oellien, F.: Algorithmen und Applikationen zur interaktiven Visualisierung und Analyse chemiespezifischer Datensätze, Dissertation Friedrich-Alexander-Universität Erlangen-Nürnberg, Nürnberg 2002.

[PietroDíaz1990] Prieto-Díaz, R.: Domain Analysis: An Introduction. In: Engineering Notes, Vol. 15, No. 2, S. 47-54, 1990.

[Pfuhl2003] Pfuhl, M.: Case-Based-Reasoning auf der Grundlage Relationaler Datenbanken – Eine Anwendung zur strukturierten Suche in Wirtschaftsnachrichten, Dissertation Universität Marburg, Deutscher Universitäts-Verlag, Wiesbaden 2003

[Porter1980] Porter, M.F.: An algorithm for suffix stripping. In: Program, 14(3), S. 130-137, Juli 1980

[Quasthoff2006] Quasthoff, U., Richter, M.: Projekt deutscher Wortschatz, http://www.babylonia-ti.ch/BABY30/uariit.htm Verifizierungsdatum: 20 März 2007.

.[Richter1995] Richter, M. M.: The Knowledge Contained In Similarity Measures. Invited talk, First International Conference on Case-Based Reasoning (ICCBR'95), Sesimbra, Portugal 1995.

[Richter2003] Richter, M. M.: Fallbasiertes Schließen. In: Görz, Günther; Rollinger, Claus-Rainer; Schneeberger, Josef (Hrsg.): Handbuch der Künstlichen Intelligenz. 4. Auflage, München/Wien 2003, S. 407-430.

[Rijsbergen1979] Rijsbergen, C.J. van: Information Retrieval. Butterworths, London, 1979.

[RiesbeckSchank1989] Riesbeck C. K., Schank R. C.: Inside Case-Based Reasoning, Lawrence Erlbaum Associates 1989.

[Schank1982] Schank, R.: Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press, New York, 1982.

[Schaaf1998] Schaaf, J.: Über die Suche nach situationsgerechten Fällen im fallbasierten Schließen. Dissertation an der Universität Kaiserslautern, verfügbar als DISKI 179, infix Verlag, Berlin 1998.

[Schiemann2007] Schiemann,,B., Reiß,P.: Web (Site) Engineering (WebSE), Vorlesung 14: KI, Agenten und NLP im Internet, Universität Erlangen, 2007: http://www8.informatik.uni-erlangen.de/IMMD8/Lectures/WEB/vorlesung/WS2006-2007/index.html, Verifizierungsdatum: 29. März 2007.

[Steinmüller2005] Steinmüller, J.: Expertensysteme, Vorlesungsskript der TU Chemnitz Sommersemester 2005 www-user.tu-chemnitz.de/~stj/lehre/xps.pdf, Verifizierungsdatum: 13.April 2007.

[StolpmannWess1998] Stolpmann, M., Wess, S.: Optimierung der Kundenbeziehung mit CBR-Systemen, Addison-Wesley, 1998

[WilsonBradshaw1999] Wilson, D., Bradshaw, S.: CBR Textuality. In Proceedings of the Fourth UK Case-Based Reasoning Workshop, 1999.

[Wikipedia2007] Internet-Enzyklopädie Wikipedia. http://wikipedia.org, 2007. – März 2007

[Wess1994] Wess, S., Althoff, K.-D., Derwand, G. (1994). Using K-D Trees to Improve the Retrieval Step in Case-Based Reasoning. In: S. Wess, K.-D. Althoff & M. M. Richter (eds.), Topics in Case-Based Reasoning, Springer Verlag, S. 167-181

[Wess1995] Wess, S.: Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik, Dissertation an der Universität Kaiserslautern, verfügbar als DISKI 126, infix Verlag, Sankt Augustin, 1995

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die Masterarbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht. Ferner erkläre ich, dass die Arbeit noch nicht in einem anderen Studiengang als Prüfungsleistung verwendet wurde.

Kerstin Bach
Hildesheim, den 17. April 2007

Einverständniserklärung

Ich erkläre hiermit mein Einverständnis, dass die von mir erstellte Masterarbeit in der Bibliothek der Universität Hildesheim ausgestellt werden darf.

Kerstin Bach
Hildesheim, den 17. April 2007