Benutzerunterstützung für Wissensentdeckung in Datenbanken.*

Robert Engels[1], Guido Lindner[2] und Rudi Studer[1]
[1] Universität Karlsruhe, Institut AIFB, D-76128, Karlsruhe,
Email: {engels,studer}@aifb.uni-karlsruhe.de
[2] Daimler Benz AG, Forschung und Technik, F3S/E
p.o. Mercedes Benz AG, T-402 D-70322 Stuttgart
Email: lindner@str.daimler-benz.com

Zusammenfassung Der Bereich der Wissensentdeckung aus großen Datenbeständen ('Knowledge Discovery in Databases', KDD) spielt eine immer größer werdende Rolle in der Wirtschaft. Ein wichtiges Akzeptanzkriterium bei der Einführung von Wissensentdeckungssverfahren in der Industrie bildet die Benutzerunterstützung. Insbesondere für ungeschulte Anwender, die mit einer Vielzahl von möglicherweise überaus komplexen Problemen konfrontiert sind, die mit Hilfe von Wissensentdeckungsverfahren gelöst werden könnten, stellt sich die Frage, wie diese Anwender unterstützt werden können. Frühere Erfahrungen aus den Bereichen Wissensakquisition und Knowledge Engineering weisen darauf hin, daß die Bereitstellung einer methodischen Vorgehensweise und das Wiederverwenden von bereits erfolgreich angewendeten (Teil-) Lösungen eine gute Basis bilden, um eine Benutzerunterstützung zu entwickeln. In unserem Ansatz spielt das Zerlegen von Aufgaben und die Anwendung von Problemlösungsmethoden ('Problem Solving Methods', PSMs) zusammen mit einer halbautomatischen Konfigurierung einer Anfangslösung für ein KDD-Problem eine zentrale Rolle. Diese Anfangslösung kann dem Anwender vorgeschlagen werden, der sie als Basis für weitere Tests und Verfeinerungen benutzen kann.

Dieses Kapitel behandelt den Aspekt der Benutzerunterstützung bei der Lösung von KDD-Problemen basierend auf methodischen Arbeiten in den Bereichen KDD und Knowledge Engineering.

1 Einführung

Generell gesehen ist die Idee der Benutzerunterstützung nicht neu. In verschiedenen Forschungsgebieten finden Fragen der Benutzerunterstützung großes Interesse: Das Thema taucht u.a. als strategische Ebene im Modell der Expertise im Knowledge Engineering auf (Schreiber et al., 1993), aber auch in der Statistik, wo die Auswahl anwendbarer Verfahren für ein Problem eine wichtige Rolle spielt (Hand, 1994b), (Hand, 1994a). Dieselbe Anforderung gewinnt zunehmend

* Publiziert in: Ch. Nakhaeizadeh. Data Mining: Theoretische Aspekte und Anwendungen. Springer Verlag, 1998.

an Bedeutung in der Wissensentdeckung in Datenbanken (KDD), da immer mehr Algorithmen für die verschiedenen Teilaufgaben im KDD–Prozeß ((Fayyad et al., 1996), (Reinartz and Wirth, 1995), (Adriaans and Zantinge, 1996)) zur Verfügung stehen. Die Notwendigkeit einer Benutzerunterstützung im KDD-Prozeß zeigt die wachsende Weiterentwicklung des Forschungsgebiets KDD, da bei der Spezifikation einer Benutzerunterstützung von den technischen Details der verschiedenen eingesetzten Algorithmen abstrahiert werden muß und es notwendig ist, sich Gedanken über die Eigenschaften der Aufgabenstellung und die Einsetzbarkeit der Algorithmen in den verschiedenen Kontexten zu machen. Das Benutzerunterstützungsmodul ('User Guidance Module', UGM) (Engels, 1996)) soll den Anwender bei der Problemspezifikation und der Auswahl der geeigneten Algorithmen unterstützen.

Methodisch betrachtet ist eine solche Komponente relativ unabhängig vom zugrundeliegenden Software-Tool, aber im Rahmen des CITRUS-Projektes basiert die Implementierung auf dem CLEMENTINE System[2] (Wirth et al., 1997).

Das UGM bietet dem Anwender, der kein Experte im Entwickeln und Ausführen von KDD-Prozessen ist, Unterstützung auf verschiedenen Ebenen:

- **Unterstützung** bei der Beschreibung des betrachteten KDD-Problems.
- **Abbildung** auf Aufgaben (Teilaufgaben), die eine Lösung des KDD-Problems ermöglichen.
- **Vorschlagen** von ersten Lösungen, die der Anwender direkt ausführen oder als Basis für Weiterentwicklungen nutzen kann.
- **Beschreibung** von Lösungen und Teillösungen, um sie verständlich und damit wiederverwendbar zu machen.
- **Verwaltung** von Lösungskomponenten zur Wiederverwendung.

In diesem Kapitel wollen wir diese fünf Punkte einzeln behandeln, obwohl oft sehr starke Abhängigkeiten zwischen den Ebenen existieren.

2 Beschreibung des aktuellen KDD-Problems

Das Beschreiben eines KDD-Problems ist sicherlich keine triviale Aufgabe. Obwohl ein Anwender vermeintlich denkt, daß er ein ziemlich klar definiertes Problem hat, ist es oft der Fall, daß die endgültige Lösung eine andere Frage beantwortet als die anfangs gestellte. Verschiedene Ursachen können für diesen Wechsel der Zielsetzung angeführt werden, aber die wichtigste ist wahrscheinlich der kreative Prozeß, der durch die Analyse der Daten und das Vertrautwerden mit den Details des Problems ausgelöst wird. Im Hinblick auf diese Kreativität ist es die erste Aufgabe des UGM, dem Benutzer einige seiner Problembesonderheiten bewußt zu machen. Diese Vorgehensweise veranlaßt den Anwender, über

[2] Trademark by INTEGRAL SOLUTIONS LTD. (ISL) (Shearer, 1996).

sein Problem nachzudenken und zu versuchen, einige *harte* Bedingungen zu definieren anstelle von vagen Charakterisierungen. Sofern widersprüchliche Ziele von Seiten des Anwenders aufgestellt werden, zeigt das UGM diese auf, so daß der Anwender die betreffenden Kriterien ändern kann. Es ist dann Aufgabe des UGM, die Problembeschreibung auf entsprechende Aufgaben abzubilden. Die im folgenden benutzte Terminologie ist aus (Engels, 1996) entnommen:

Problem: Ein Problem ist im allgemeinen definiert als ein Artefakt, das drei Elemente enthält: einen Ausgangszustand, einen Zielzustand und eine Diskrepanz zwischen diesen Zuständen.

Bekanntermaßen kann zwischen schlecht definierten und wohldefinierten Problemen unterschieden werden. Ein Problem ist schlecht definiert[3], wenn der Zielzustand nicht klar beschrieben ist, der Ausgangszustand nicht eindeutig ist (einige relevante Charakteristika des Ausgangszustands sind möglicherweise unbekannt) oder benötigte Informationen zur Lösung des Problems nicht verfügbar sind, d.h. ein Problem ist auch schlecht definiert, wenn die Charakteristik des Ausgangszustands in keinem Zusammenhang mit der Charakteristik des Zielzustands steht, da sie dann nicht vergleichbar sind. Der Ausgangszustand ist beschrieben durch die Art des Anwendungsgebietes, das Format der Daten (relationale Datenbank, einfache Datei,etc.) sowie die Datencharakteristika (Anzahl der Fälle, Verteilungen, Standardabweichung, etc.). An einem kleinen Beispiel sollen die Begriffe, die in diesem Kapitel verwendet werden, verdeutlicht werden.

Beispiel: *Der Manager eines neuen Motorradrennteams $\mathcal{WEIRD}$ sucht nach neuen jungen Talenten. Normalerweise ist hierfür eine Menge Erfahrung notwendig, der Manager aber hat noch ein anderes As auf der Hand. Einer seiner Partner ist sehr an neuen Techniken interessiert und weiß, daß einige amerikanische Basketballteams Techniken des maschinellen Lernens zur Einschätzung der gegnerischen Stärken einsetzen. Da er Informationen über frühere Spitzenfahrer und über den Beginn ihrer Karrerien besitzt, entscheidet er sich, diese neuen Verfahren für die Lösung seines Problemes einzusetzen. Die zur Verfügung stehenden Informationen beinhalten die Kubikklasse, mit der die Spitzenfahrer ihre Karriere begonnen haben, die unterschiedlichen Typen Motorräder, die sie gefahren haben, mit welchem Alter sie ihren ersten großen Erfolg hatten, die Trainingsmethode, Gewicht und Körpergröße.*

[3] Siehe auch (Simon, 1985)

Name	Alter Erster Erfolg	erste MC	MC Typen	Trainingsmethode	Gewicht	Größe
Loe S.	*24*	*125*	*all*	*Winter*	*64*	*173*
Lars K.	*30*	*50*	*250*	*ganzjährig*	*78*	*168*
Gerhard D.	*19*	*125*	*all*	*Herbst*	*83*	*175*
Dwane R.	*23*	*125*	*all*	*Winter*	*70*	*170*
Bob E.	*22*	*50*	*250*	*Herbst*	*97*	*187*
......	..	...	...	....	..	...
......	..	...	...	....	..	...

Tabelle 1: *Auschnitt aus den zur Verfügung stehenden Daten.*

Ein wohldefiniertes Problem ist ein Problem, bei dem ein eindeutiger Anfangszustand und ein Zielzustand gegeben sind sowie eine Diskrepanz zwischen diesen beiden Zuständen, für die keine Lösung bekannt ist. Ein Problem kann in natürlicher Sprache formuliert werden. Diese Formulierung bildet die Basis, auf der eine Problembeschreibung definiert werden kann.

Problembeschreibung: Eine Problembeschreibung besteht aus einer Beschreibung des derzeitigen Ausgangszustandes des Systems sowie des zu erreichenden Zielzustandes.

Fortführung des Beispiels: *Das Finden eines neuen Talentes bildet das eigentliche Problem. Deshalb existiert eine Diskrepanz zwischen dem Anfangszustand, in dem eine Menge von Beschreibungsmerkmalen von Spitzenfahrern in deren früherer Karriere zur Verfügung steht, und dem Endzustand, in dem nach explizitem Wissen gesucht wird, das Gewinner beschreibt.*

Das UGM hilft dem Anwender, in einem natürlichsprachlicher Dialog sein Problem zu beschreiben. Hierzu wird die zuerst nur als schlecht definiertes Problem vorliegende Problemstellung in einem Frage-Antwort-Dialog in eine eindeutige Problembeschreibung transformiert, die konsistent und präzise ist.

Fortführung des Beispiels: *Wenn unser $\mathcal{WEIRD}$-Manager zum Beispiel sein Problem mit der Anforderung* `interpretierbar = Ja` *beschreibt, dann müssen keine Fragen gestellt werden, die für Aufgaben relevant sind, für die die Anforderung* `interpretierbar = Nein` *gilt.* `interpretierbar = Nein` *könnte der Fall sein, wenn Entscheidungen in Echtzeit benötigt werden (z.B. wenn man zu entscheiden hat, ob ein Fahrer einen Boxen-Stop einlegen soll, wenn er gerade in Führung liegt). Ein interpretierbares Modell enthält in diesem Fall logische Regeln, die möglicherweise weniger wichtig für den Manager sind als ein Klassifikator, der schnell entscheidet, aber kein interpretierbares Ergebnis liefert.*

Während des Frage-Antwort-Dialogs werden die charakteristischen Eigenschaften des Problems soweit definiert wie es notwendig ist, um eine Vorgehensweise zur Lösung des Problems zu finden. Im Dialog sind dabei gegebenenfalls frühere Entscheidungen zurückzunehmen, um Inkonsistenzen in der Problembeschreibung zu beseitigen. Dies führt dazu, daß der Anwender neue Fragen in Bezug auf die dadurch geänderten Anforderungen beantworten muß. Durch diesen explorativen Dialog erhält der Anwender schrittweise ein klares Verständnis über seine ihm vorliegende Problemstellung.

3 Aufgabenidentifikation und -zerlegung

Sobald das Problem so detailliert definiert ist, daß eine passende Aufgabe ausgewählt werden kann, wird dies getan. Diese Aufgabe bildet die Basis für Verfeinerungen der Vorgehensweise. Hierzu wird die Aufgabe in Teilaufgaben zerlegt, gegebenenfalls über mehrere Verfeinerungsebenen hinweg. Ab einer gewissen Verfeinerungsstufe, die von Anwendung zu Anwendung variieren kann, erhält man sogenannte primitive Aufgaben, die nicht weiter zerlegt werden. Im nächsten Schritt müssen dann die Techniken bestimmt werden, die die primitiven Aufgaben ausführen. Dieser Schritt wird in Kapitel 4 beschrieben. Im folgenden werden wir die Terminologie für die Aufgabenzerlegung einführen, die auf den in (Chandrasekaran et al., 1992), (Schreiber et al., 1993) und (Breuker and van de Velde, 1994) beschriebenen Arbeiten aufbaut.

Aufgabe: Eine Aufgabe beschreibt die Transformation eines Anfangszustandes in einen Zielzustand. Dabei sind die beiden Zustände wohldefiniert, die Aktion, mit der die Transformation durchgeführt werden kann, jedoch noch nicht. Eine Aufgabe hat eine Menge von Vor- und Nachbedingungen.

Die Vorbedingungen legen die möglichen Anwendungsbereiche einer Aufgabe fest, und die Nachbedingungen beschreiben das Ergebnis nach der Ausführung der Aufgabe (task effects (Decker et al., 1996)). Diese Vor- und Nachbedingungen werden benutzt, um verschiedene (Teil-)Aufgaben miteinander zu verbinden. Hierzu paßt die Metapher eines JIGSAW-Puzzles sehr gut. Man stelle sich die Aufgabe als ein Puzzleteil vor, wobei die Ausstanzung der einen Seite die Vorbedingungen darstellen und die entgegengesetzte Seite des Puzzleteils die Nachbedingungen darstellen.

Aufgabenzerlegung: Eine Aufgabenzerlegung ist die Verfeinerung einer Aufgabe in eine Aufgaben- Teilaufgabenhierarchie.

Die Aufgabe auf oberster Ebene ist identisch mit der Aufgabe, die zu lösen ist, und die Blätter des Baumes bilden eine Menge von Teilaufgaben, die gemeinsam die Ausgangsaufgabe lösen. Die Zerlegung ermöglicht verschiedene Stufen der Abstraktion. Eine Teilaufgabe, die nicht in weitere zerlegt werden kann, nennt man primitive Aufgabe.

Die Aufgabenzerlegung ist eine konjunktive Verfeinerung, da die Funktionalität der Ausgangsaufgabe durch eine partiell geordnete Ausführung aller Teilaufgaben erreicht wird. Disjunktive Verfeinerungen sind in unserem Ansatz nicht möglich, da immer die am besten passende ausgewählt wird, sofern für das Überbrücken der Diskrepanz zwischen den Vor- und Nachbedingungen einer Aufgabe mehrere alternative Verfeinerungen existieren. So wird in unserem Ansatz letztendlich ein Verfeinerungsbaum generiert, der nur konjunktive Knoten enthält. Dabei wird die Funktionalität einer bestimmten Aufgabe durch ihre Vor- und Nachbedingungen definiert. Im allgemeinen kann die Aufgabenzerlegung zu einer Spezialisierung der Funktionalität der Ausgangsaufgabe führen, da eine Teilaufgabe zusätzliche und/oder genauere Vor- und Nachbedingungen einführen kann. Die Zerlegung einer Aufgabe in Teilaufgaben führt zu eine Folge von Teilaufgaben, bei denen die Nachbedingungen der Teilaufgabe i hinreichende Vorbedingungen für die Ausführung der Teilaufgabe $(i+1)$ spezifizieren.

Fortführung des Beispiels: *Das Ziel der Anwendung ist die Konstruktion einer Wissensbasis, die junge Talente in Durchschnittliche und Vielversprechende einteilen kann, basierend auf den Beschreibungsmerkmalen früherer Spitzenfahrer. Die Zerlegung des KDD-Prozesses in Teilaufgaben führt zur Identifikation folgender Teilaufgaben: Beschaffung der Daten, Vorverarbeitung (Ausfiltern der Attribute Gewicht und Größe, Hinzufügen des Ratioattributes Gewicht/Größe), Klassenbildung und Aufbau eines Klassifikators. Wir werden uns im folgenden auf die letzten drei Teilaufgaben beschränken.*

Problemlösungsmethode: Eine Problemlösungsmethode[4] (PSM) beschreibt, wie eine Aufgabe durch Zerlegung in Teilaufgaben sowie die Ausführung der Teilaufgaben in der durch die Kontrollstruktur festgelegten Reihenfolge gelöst werden kann.

Wir benutzen das Konzept der PSM, um vordefinierte, wiederverwendbare Teilprozesse in einer Bibliothek ablegen zu können. Durch die Wiederverwendung einer PSM kann die Zerlegung einer bestimmten Aufgabe in die durch die PSM festgelegten Teilaufgaben erreicht werden. Damit kann einem ungeschulten Anwender eine allgemeine Lösungsstruktur für sein Anwendungsproblem zur Verfügung gestellt werden.

Wenn eine komplette Aufgabenzerlegung gefunden ist, ist der nächste Schritt, geeignete Techniken den einzelnen Teilaufgaben zuzuordnen. In manchen Fällen ergibt sich auch die Möglichkeit, PSMs auf noch nicht komplett zerlegte Teilaufgaben abzubilden. Die Auswahl solcher PSMs wird in unserem Ansatz unterstützt, aber in unserem Beispiel nicht weiter erwähnt.

[4] Ein nähere Beschreibung des Konzeptes 'Problemlösungsmethode' ist in (Schreiber et al., 1993), (Breuker and van de Velde, 1994) zu finden, eine Analyse von Problemlösungsmethoden ist in (Benjamins et al., 1996) beschrieben.

4 Abbildung von Aufgaben auf Techniken und Verfahren: Ein zweistufiger Prozeß

Die verschiedenen Teilaufgaben aus der Aufgabenzerlegung müssen nun den verschiedenen zur Verfügung stehenden Techniken zugeordnet werden. Unserer Ansicht nach bedingt dies ein zweistufiges Vorgehen. Erstens das Zuordnen von Technikklassen zu den einzelnen Teilaufgaben und zweitens die Auswahl einer speziellen Technik, die letztlich dem Anwender vorgeschlagen wird.

Zunächst wollen wir aber die Terminologie für diesen Abschnitt einführen.

Techniken: Techniken sind Algorithmen, die die definierte Funktionalität einer Aufgabe erfüllen können. In unserem Zusammenhang sind solche Algorithmen nicht notwendigerweise Data Mining Algorithmen. Es können genauso andere Algorithmen, z.B. zur Fokussierung oder zur Visualisierung, sein oder Schritte, die manuell ausgeführt werden müssen.

In unserer Beispielanwendung *sind an Techniken verfügbar C4.5, ein Regellerner, ein Clusteringalgorithmus und lineare Regression. Für jede Technik ist eine Menge von Parametern definiert, mit denen das Verhalten des Techniken beeinflußt werden kann. So beeinflußt das Abbruchkriterium oder Akzeptanzkriterium die Qualität des Lernergebnisses oder das Pruningkriterium die Komplexität der gelernten Modelle.*

Abbildung: Der Abbildungsprozeß bezieht sich auf die Abbildung der Aufgabenzerlegung zusammen mit der dafür definierten Kontrollstruktur auf eine geordnete Sequenz von Techniken, die zusammen aus dem Anfangszustand den Zielzustand der Problembeschreibung erreichen.

Dieser Schritt verbindet die Aufgabenzerlegung mit den Techniken. Der Prozeß der Zerlegung und Abbildung ist ein iterativer Prozeß, der auch die Initialisierung der Parameter für die verschiedenen Techniken unterstützt. Die Kontrollstruktur ist momentan einfach definiert. Die Blätter eines Aufgabenzerlegungsbaums sind so angeordnet, daß die Ausführung von links nach rechts durchgeführt werden muß. Iterationen sind zur Zeit nicht zugelassen. Dies ist jedoch keine prinzipielle Einschränkung des Ansatzes. Der Anwender kann solch eine erste (einfache) Prozeßstruktur als Ausgangspunkt für eine weitere Verfeinerung nehmen. Auf Grund der Evaluierung erster Ergebnisse, die mit der Ausführung der vorliegenden Prozeßstruktur erzielt werden, wird i.a. eine Verfeinerung bzw. Modifikation der Prozeßstruktur notwendig. Dadurch wird der gesammte KDD-Prozeß iterativ. Die Unterstützung der Evaluierung von Ergebnissen sowie der dadurch ausgelösten Iterationen ist Gegenstand zukünftiger Forschung.

Lösung: Eine Lösung ist das Ergebnis eines iterativen Planungs-prozesses. Während dieses Planungsprozesses transformiert der Anwendungsexperte, eventuell in Zusammenarbeit mit einem Knowledge Engineer, ein schlecht definiertes Problem in ein wohl definiertes Problem. Diese wohl definierte Problembeschreibung (zusammen mit eventuell zusätzlich bekannten Anwendungsmerkmalen) bildet die Ausgangsbasis für den Planungsprozeß in einem stark strukturierten, aber nicht abgeschlossenen Suchraum. Das Ergebnis dieses Planungs- und Abbildungsprozesses besteht aus einer Zerlegung der KDD-Aufgabe in Teilaufgaben und einer Abbildung der Teilaufgaben auf eine Menge von Techniken, beides zusammen bildet die Lösung. Teil dieser Lösung ist eine Kontrollstruktur, die die Ausführungsreihenfolge definiert.

Der Suchraum ist prinzipiell abgeschlossen, da man annehmen kann, eine Lösung zu finden, wenn man eine wohl-definierte Problembeschreibung hat, die zu der Aufgabe auf oberster Ebene paßt. Trotzdem kann das Finden einer Lösung nicht in jedem Fall garantiert werden, weil das Verfeinern von Aufgaben dazu führen kann, daß zusätzliche Vor- und Nachbedingungen eingeführt werden, die eventuell nicht erfüllt werden können. Ist dies der Fall, so wird versucht, für eine abgeschwächte Aufgabenstellung eine Lösung aus der Bibliothek anzubieten.

Beispielanwendung: *Die endgültig gefundene Lösung besteht aus einen zweistufigen Prozeß. Im ersten Schritt werden die Fahrer durch einen multidimensionalen Raum beschrieben, in dem sie in Gruppen geclustert werden. Der zweite Schritt liefert dann eine Beschreibung der gebildeten Klassen. Auf diese Weise erhält der Manager eine Beschreibung der verschiedenen Gruppen von jungen, potentiell erfolgreichen Fahrern (z.B. eine Gruppe von jungen Fahrern, die von Anfang an große Maschinen fährt, im Gegensatz zu einer Gruppe von Fahrern, die zu Beginn ihrer Karriere nicht vielversprechend aussieht und erst im Laufe ihrer Karriere zu erfolgreichen Fahrern wird).*

Eingabedaten: Eingabedaten sind die Daten, die analysiert werden sollen. Diese Daten bilden die Ausgangsbasis für den KDD–Prozeß.

Eingabedaten können als intensionale und extensionale Beschreibung von anwendungsspezifischem Wissen vorliegen und sind der Gegenstand der Analyse. Meist liegen die Daten in extensionaler Form als Menge von Fakten vor, die ein Verhalten oder eine Beziehung aus der Anwendung beschreiben. In unserer Beispielanwendung ist eine Tabelle mit Beispielen als Eingabedaten gegeben.

Entdecktes Wissen: Entdecktes Wissen ist das Ergebnis der Ausführung der erstellten Lösung für das KDD-Problem, das in der Problembeschreibung definiert ist.

Wir unterscheiden explizit zwischen Lösung und endecktem Wissen. Eine Lösung ist das Ergebnis des Planungsprozesses, den das UGM unterstützt. Entdecktes Wissen dagegen ist das Ergebnis der Ausführung einer Lösung.

Beispielanwendung: *Entdecktes Wissen in unserer Beispielanwendung ist im wesentlichen eine Menge von Begriffsbeschreibungen. Ein Beispiel, wie ein solches Ergebnis aussehen könnte, ist in der folgenden Tabelle dargestellt* [5]*:*

Regeln	
Regel 1:	*wenn* `MC Typen(alle)` $\wedge$ `trainingsmethode(ganzjährig)` $\rightarrow$ `Potentieller Spitzenfahrer`
Regel 2:	*wenn* `Alter Erster Erfolg(21-24)` $\wedge$ `erste MC(125)` $\rightarrow$ `Potentieller Spitzenfahrer`
......:	..

Der Anwender, der das Problem gestellt hat, wird hauptsächlich an diesem 'entdeckten Wissen' interessiert sein. Unser Ansatz zielt jedoch primär darauf ab, den Prozeß der Lösungsfindung zu unterstützen.

4.1 Zuordnung von Techniken zu Aufgabenstellungen

Beim Zerlegen einer Anfangsaufgabe unter Verwendung von PSMs wird eine Baumstruktur hergeleitet, die die Aufgabe in mehreren Abstraktionsstufen beschreibt. Wie im vorangegangenen Kapitel aufgezeigt wurde, müssen den auf der untersten Verfeinerungsebene identifizierten primitiven Teilaufgaben geeignete Techniken zu ihrer Lösung zugeordnet werden.

Es ist eher unwahrscheinlich, sofort eine eindeutige Zuordnung zwischen einer primitiven Aufgabe aus der Aufgabenzerlegung und einer Technik zu finden. Viel wahrscheinlicher ist es, daß eine Teilmenge aller möglichen Techniken zur Lösung der betrachteten Aufgabe einsetzbar ist. In solchen Fällen wird dann die Klasse von Techniken ausgewählt, die am besten zu der Aufgabe paßt. Abb. 1 verdeutlicht die Beziehung zwischen der Problembeschreibung, einer Aufgabenzerlegung, den Datencharakteristika und der Auswahl von Technik-Klassen.

Beispielanwendung: *Da wir unsere Beispielanwendung letztendlich auf zwei primäre Teilaufgaben reduziert haben, Klassenbildung und Aufbau eines Klassifikators, wird unsere Suche auf eine Klasse von Techniken eingegrenzt, die Cluster bilden und interpretierbare Klassifikatoren generieren können, wie es in der Problembeschreibung definiert ist.*

Eine solche Klasse von Techniken kann später weiter aufgegliedert werden, solange bis nur noch eine einzelne Technik angeboten werden kann. Hierbei spielen Datencharakteristika eine zentrale Rolle.

[5] Es sei noch einmal daraufhingewiesen, daß das Beispiel nur zur Veranschaulichung gedacht ist. Wir versuchen nicht einen bestimmten Algorithmus zu erklären. Beispielhaft besagt die erste Regel, daß wenn die Person alle Typen von Motorrädern gefahren hat und das ganze Jahr trainiert, die Person ein Kandidat für einen Spitzenfahrer ist.

4.2 Auswahl von Techniken mit Hilfe von Datencharakteristika

Ein wichtiger, letzter Schritt bei der Entwicklung eines KDD-Prozesses ist die Auswahl der Data-Mining-Technik. Die Auswahl der hier einsetzbaren Technik hängt von zwei Dingen ab.

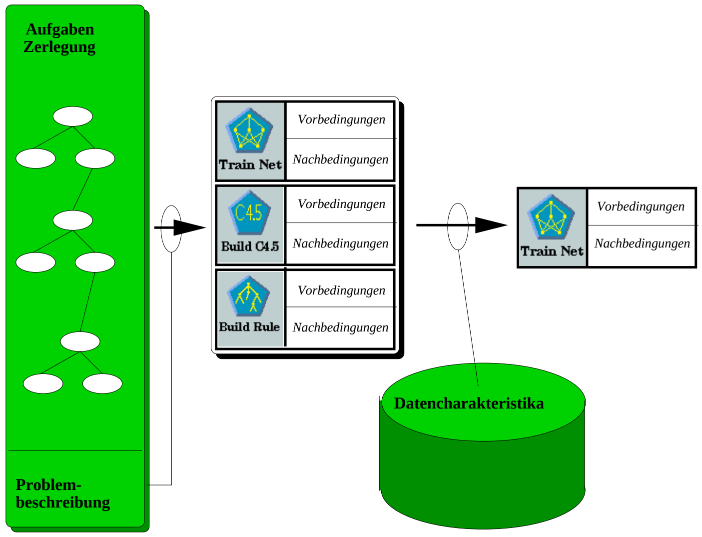

Abbildung1. Das Prozeß der Selektion einer Technik für eine Aufgabe.

Zum einen beinhalten die Daten selbst Informationen, die bei der Wahl der Data-Mining-Technik berücksichtigt werden sollten. Diese Anforderung ist mehr oder weniger pragmatisch, da verschiedene Techniken in ihrer Einsetzbarkeit für Anwendungsbereiche mit unterschiedlichen Datencharakteristika eingeschränkt sind. Da die meisten (großen) Datenbanken ein Data-Dictionary besitzen, ist es naheliegend, dies zu benutzen, um charakteristische Informationen über die Daten zu erhalten. Außerdem können einfache statistische Maße, wie beispielsweise Standardabweichung, Mittelwerte und Wahrscheinlichkeiten, die die Daten beschreiben und charakterisieren, berechnet werden. Im Statlog-Projekt (Michie et al., 1994) werden solche Maße zum Test der Anwendbarkeit von statistischen Verfahren und Data-Mining-Algorithmen eingesetzt. Das UGM wird ähnliche Maße zur Auswahl von Techniken aus der vorher bestimmten Teilklasse verwenden. Hierbei nutzen wir Informationen aus dem Data-Dictionary, wie beispielsweise fehlende Werte, und statistische Maße wie die Schiefe der Verteilung der Attributwerte.

Auf der einen Seite werden diese Informationen für die endgültige Auswahl einer Technik aus einer Menge von Techniken benutzt. Auf der anderen Seite werden die Datencharakteristika zusammen mit den Informationen, die aus der Problembeschreibung und der Aufgabenzerlegung kommen, verwendet, um Parameter für die ausgewählte Technik zu belegen. Diese Information wird in sogenannten Initialisierungsboxen verwendet. Abb. 2 zeigt ein Beispiel einer solchen Box.

	Daten			
	Balanciert		Unbalanciert	
	kleine Menge	große Menge	kleine Menge	große Menge
Windowing	Nein	Ja	???	Ja
Kreuz-validierung	Ja	Nein	Ja	Nein

Edit-parameter Initialisierungsregeln

```
if number(data_tuples) > 10.000
then
   windowing(sample_size(10%), number_of_trials(5), sample_increment(5%));
endif;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if (number(data_tuples) < 10.000 ^  balanced(data_tuples) == true)
then
   cross_validation == true;
endif;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
if (number(data_tuples) < 10.000 ^  balanced(data_tuples) == false)
then
```

Abbildung2. Eine Parameterinitialisierungsbox für Clementine's C4.5 Knoten basierend auf Datencharakteristika (teilweise nach (Quinlan, 1993).

In der Abbildung sind Datencharakteristika, wie beispielsweise die Größe der Datenbank oder ob die Datenmenge balanciert ist, zu finden. Balanzierte Daten weisen für jede Klasse im Durchschnitt gleiche Mengen an Beispielsdaten auf. In Abhängigkeit davon wird angegeben, ob bestimmte Heuristiken (wie beispielsweise Windowing und Kreuzvalidierung) angewendet werden können. Die Parameterinitialisierungsregeln bieten also Unterstützung, Parameter kontextabhängig mit Werten zu belegen. Kontextbezogene Parameterinitialisierung kann den Entwicklungsaufwand einer KDD-Anwendung merklich reduzieren.

5 Die Architektur des UGM, wiederverwendbare Prozesskomponenten und die Benutzung von Annotationen

Die Architektur des UGM ist in Abbildung 3 dargestellt. Die Architektur zeigt die Abhängigkeiten zwischen der Benutzungsschnittstelle, der Planungskomponente, der Suchkomponente und verschiedenen Datenquellen, die die einzelnen

Komponenten mit ausreichendem Wissen versorgen.

Die *Problemanalysekomponente* hilft dem Anwender, durch das Stellen von relevanten Fragen, sein Problem zu beschreiben. Die *Komponente zur Fragengenerierung* ist eingebettet in die Problemanalysekomponente. Sie ist intelligent in dem Sinne, daß sie dem Anwender keine irrelevanten Fragen stellt. Weiterhin werden die Fragen über eine natürlichsprachliche Schnittstelle gestellt.

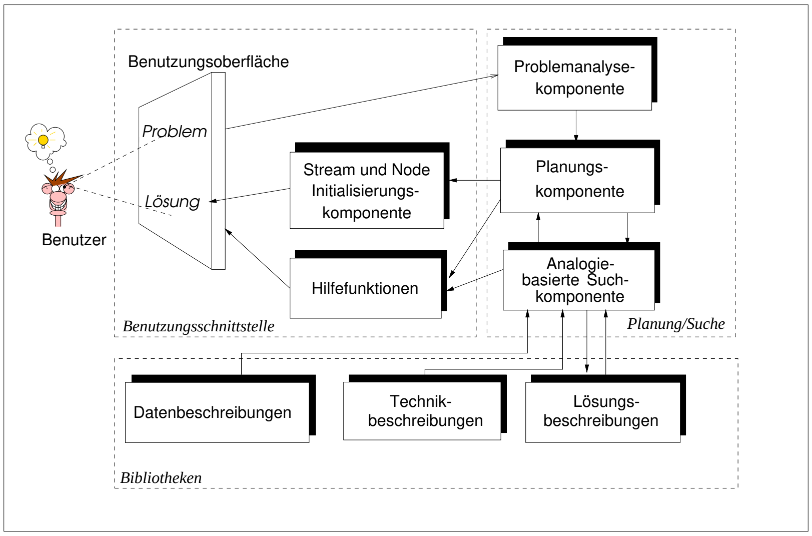

Abbildung3. Architektur des UGM

Die Problemanalysekomponente erstellt die Problembeschreibung durch Interaktion mit dem Anwender. Hierbei wird die zuerst nur als schlecht definiertes Problem vorliegende Problemstellung interaktiv in einem Frage- Antwort- Dialog in eine eindeutige Problembeschreibung transformiert, die konsistent und präzise ist.

Ein weiterer wichtiger Punkt bei der Benutzerunterstützung ist das Annotieren von wiederverwendbaren Prozeßkomponenten. Von einer Lösung können entweder der gesamte Ablauf oder Teile des Ablaufs für neue KDD-Lösungen wiederverwendet werden. Deswegen werden Lösungen oder Teile davon als Komponenten für eine spätere Wiederverwendung gespeichert. Diese Komponenten stammen entweder aus vorangegangenen, erfolgreich angewendeten KDD-Lösungen oder aus einer Komponenten-Bibliothek, die allgemeine Lösungen für typische Teilaufgaben von KDD-Prozessen zur Verfügung stellt. Diese Komponenten sind von speziellen Anwendungen unabhängig.

Um die Wiederverwendung der in der Bibliothek abgelegten Komponenten zu erleichtern, müssen die Komponenten mit Annotationen verknüpft werden, die ihre charakteristischen Eigenschaften beschreiben. Zu diesen charakteristischen Eigenschaften gehören auch Angaben darüber, bei welchen Lösungen diese Komponenten in anderen Kontexten eingesetzt worden sind. Einerseits werden diese Annotationen dem Anwender in natürlichsprachlicher Form beim Durchsuchen der Bibliothek angeboten. Andererseits wird eine (semi-)formale interne Darstellung der Annotationen in der UGM-Planungskomponente verwendet, um geeignete Komponenten für die Konstruktion einer Lösung für ein neues Anwendungsproblem auszuwählen. Damit kann das UGM einem unerfahrenen Anwender einen ersten Vorschlag für die Lösung seines KDD-Problems machen. Annotationen werden halbautomatisch erzeugt, d.h. es gibt Teile, die automatisch generiert werden, und es gibt Teile, für die die Informationen durch den Anwender geliefert werden müssen.

Annotationen müssen damit zwei Anforderungen genügen:

- Zum einen sollte ein Anwender die Annotationen, die ihm das UGM zur Verfügung stellt, verstehen können, wenn er oder sie mehr über die Funktionalität der in der neuen Lösung wiederverwendeten Komponenten wissen möchte. Deshalb sind allgemeinverständliche Annotationen notwendig.
- Zum anderen sollten die gleichen Annotationen in formalerer Weise beschrieben werden, damit das UGM semi-automatisch planen und dem Anwender Lösungen vorschlagen kann.

Diese beiden Anforderungen sind teilweise widersprüchlich. Aus diesem Grund benutzen wir Vor- oder Nachbedingungen für die formale Definition der Komponenten, was uns die Möglichkeit bietet, Lösungen für KDD-Prozesse zu planen und vorzuschlagen. Auf der anderen Seite ist jede Komponente mit einer Funktionalitätsbeschreibung ausgestattet, die dem Anwender eine verständliche Prozessbeschreibung liefert. Funktionalitätsbeschreibungen werden also als Schnittstelle zum Anwender verwendet.

Unser Beispiel *endet hier. Die Anwendung der generierten Regeln halfen unserem Manager, Rainy D. aus der Menge der interessanten Bewerber auszuwählen. Der Manager entschied sich für erfahrenere Fahrer anstelle von 'Youngstern', um den Vorteil ihrer überlegenen Erfahrung während der Rennen auszunutzen. Die Entscheidungen basierten auf Clusterbeschreibungen, die aus den Originaldaten abgeleitet werden konnten.*

6 Abschließende Betrachtungen

Unsere Erfahrung zeigt, daß Anwenderunterstützung bisher nicht auf weitverbreitetes Interesse stößt. Bei der Analyse, wie Anwendungsexperten Wissensentdeckungsverfahren und -werkzeuge einsetzen, haben wir jedoch herausgefunden, daß häufig, wegen der mangelnden Unterstützung durch das KDD-Werkzeug, nach dem Trial-and-Error Prinzip gearbeitet wird. Nur wirkliche KDD-Experten, die eine Menge über die eingesetzten Techniken wissen, können passende Techniken zur Problemlösung finden und kombinieren. Aber selbst sie sind nicht immer auf dem richtigen Weg. Unser System zielt darauf ab, bereits erfolgreich angewandte Lösungsteile unerfahrenen Anwendern zugänglich zu machen und zusätzlich Annotationen anzubieten, die das Verständnis für die Komponenten und die dadurch beschriebenen Abläufe verbessern.

Wie wir bereits in der Einleitung erwähnt haben, ist das Interesse für Benutzerunterstützung in anderen Fachgebieten größer als auf dem KDD-Gebiet. Im Bereich der Statistik existieren ähnliche Probleme, die von (Hand, 1994a) analysiert werden. Obwohl die Probleme der Zuordnung von Anwenderproblemen zu passenden Lösungstechniken aufgezeigt werden, wird dort keine wirkliche Lösung angeboten.

Zum zweiten gibt es Ansätze im Forschungsgebiet des Maschinellen Lernens (z.B. (Piatetsky-Shapiro et al., 1996), (Fayyad et al., 1996), (Craw et al., 1992)). Hierfür ist das MLT-Projekt (Consortium, 1993) ein Beispiel, in dem das sogenannte CONSULTANT- Werkzeug erstellt worden ist (Craw et al., 1992). CONSULTANT verfolgt den Ansatz, mit Hilfe einer Wissensbasis einen für ein bestimmtes Problem passenden Algorithmus aus einer Menge von 10 Algorithmen auszuwählen. Diese Wissensbasis wird hauptsächlich dazu verwendet, zwischen den Algorithmen zu differenzieren. Wir beschreiben jedoch die Funktionalität der einzelnen Techniken, um dadurch die Integration neuer Techniken zu erleichtern. Außerdem kann CONSULTANT nicht die Situation verarbeiten, daß verschiedene Techniken kombiniert werden müssen, um ein Problem zu lösen. Es existiert auch keine Möglichkeit, frühere (Teil-) Lösungen wiederzuverwenden. In unserem Ansatz unterstützen wir jedoch einen komplexeren Prozeß, in den mehrere Techniken sowie frühere (Teil-)Lösungen eingebunden werden können. Generell wird für die Entwicklung von KDD-Anwendungen eine Methodik zur Konstruktion von KDD-Prozessen benötigt. Erste Ansätze hierfür kann man in (Brodley and Smyth, 1995) und (Reinartz and Wirth, 1995) finden.

Die Wiederverwendung von Komponenten sowie Methoden zur Zerlegung von Aufgaben in Teilaufgaben sind aus dem Bereich des Wissensakquisition bekannt und werden z.B. in den Ansätzen KADS (Wielinga et al., 1992), Task Structures (Chandrasekaran et al., 1992) und MIKE (Angele et al., 1996) verwendet. Dort beschriebene Methoden lassen sich teilweise für unsere Zielsetzung verwenden, frühere Lösungen von KDD-Problemen wiederzuverwenden und sie hierfür in Bibliotheken zu speichern. Die Tatsache, daß auf der einen Seite die meisten KDD-Probleme anfänglich schlecht definiert sind und daß auf der anderen Seite eine Vielzahl von Techniken existieren, die für die Lösung von KDD-Teilaufgaben eingesetzt werden können, macht es notwendig, den Anwender sowohl bei der Definition einer exakten Problembeschreibung als auch bei der Auswahl geeigneter Techniken zu unterstützen. Auf die Lösung dieser Fragestellungen zielt die Entwicklung des UGM primär ab.

Vorteile eines solchen Ansatzes sind:

- Die unklare Definition eines Problems wird präzisiert. Ein Anwender ist gezwungen, sein Problem sorgfältiger zu beschreiben, dadurch wird er vertrauter mit allen Aspekten des Problems.
- Probleme werden von den Aufgaben getrennt, die ausgeführt werden können, um diese Probleme zu lösen. Dadurch erhalten wir gleichzeitig eine Problembeschreibung und einen möglichen Weg, diese zu lösen. In dem Fall, daß schon mehrere Lösungen erfolgreich eingesetzt wurden und entsprechend dokumentiert sind, kann bei einem ähnlichen Problem sofort auf die vorhandenen Lösungen zurückgegriffen werden.
- Aufgabenstellungen werden in Teilaufgaben unterteilt, woraus eine verfeinerte Aufgabenstellung und zusätzliche Annahmen über die Eigenschaften der Teilaufgaben resultieren.
- Die Lösung eines KDD-Problems wird als eine Aufgabenzerlegung zusammen mit einer Menge von Abbildungen auf geeignete Techniken beschrieben. Komponenten in einer solchen Hierarchie können bei der Lösung neuer Probleme wiederverwendet werden.

Das weitere Vorgehen beinhaltet die Weiterentwicklung der in diesem Kapitel beschriebenen Ansätze. Für das UGM wird gerade eine Wissensbasis definiert, die dem Anwender hilft, sein Problem näher zu spezifizieren und einen geeigneten KDD-Prozeß zu definieren. Im Hinblick auf die Konstruktion einer Lösung für ein KDD-Problem konzentrieren wir uns auf das Auffinden geeigneter früherer Teillösungen aus der UGM-Bibliothek.

Literaturhinweise

Adriaans, P. and Zantinge, D. (1996). *Data Mining.* Addison-Wesley.

Angele, J., Fensel, D., and Studer, R. (1996). Domain and task modelling in mike. In Sutcliffe, A., van Assche, F., and Benyon, D., editors, *Domain Knowledge for Interactive System Design, Proceedings of the IFIP WG8.1/13.2 Joint Working Conference on Domain Knowledge for Interactive System Design.* Chapman & Hall.

Benjamins, R., Fensel, D., and Straatman, R. (1996). Assumptions of problem-solving methods and their role in knowledge engineering. *Proceedings of the 12th European Conference on Artificial Intelligence.*

Breuker, J. and van de Velde, W. (1994). *CommonKADS Library for Expertise Modelling.* IOS Press.

Brodley, C. and Smyth, P. (1995). Applying classification algorithms in practice. In Aha, D., editor, *Proceedings of the Workshop on Applying Machine Learning in Practice at the ICML-95*, Tahoe City, CA.

Chandrasekaran, B., Johnson, T. R., and Smith, J. W. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124–137.

Consortium, M. (1993). Final public report. Technical report. Esprit II Project 2154.

Craw, S., Sleeman, D., Granger, N., Rissakis, M., and Sharma, S. (1992). Consultant: Providing advice for the machine learning toolbox. In Bramer, M. and Milne, R., editors, *Research and Development in Expert Systems*, pages 5–23.

Decker, S., Erdmann, M., and Studer, R. (1996). A unifying view on business process modelling and knowledge engineering. *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW-96)*, I:1–16.

Engels, R. (1996). Planning tasks for knowledge discovery in databases; performing task-oriented user-guidance. In Simounis, E., Han, J., and Fayyad, U., editors, *Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases*, pages 170–175, Portland, Oregon. AAAI-Press.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurasamy, R. (1996). *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, London.

Hand, D. (1994a). Deconstructing statistical questions. *Journal of the Royal Statistical Society*, pages 317–356.

Hand, D. (1994b). Statistical strategy: step 1. In Cheeseman, P. and Oldford, R. W., editors, *Selecting Models from Data: Artificial Intelligence and Statistics IV*, volume 89, pages 3–9. Lecture Notes in Statistics, Springer Verlag, Berlin.

Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Piatetsky-Shapiro, G., Brachman, R., Khabaza, T., Kloesgen, W., and Simoudis, E. (1996). An overview of issues in developing industrial data mining and knowledge discovery applications. In Simoudis, E., Han, J., and Fayyad, U., editors, *Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases*, pages 89–95, Portland, Oregon. AAAI press.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Reinartz, T. and Wirth, R. (1995). Towards a task model for kdd. In Kodratoff, Y., Nakhaeizadeh, G., and Taylor, C., editors, *Proceedings of the workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases. MLNet Familiarisation Workshop*, pages 19–24, Heraklion, Crete.

Schreiber, A., Wielinga, B., and Breuker, J. (1993). *KADS: A Principled Approach to Knowledge-Based System Development*. Academic Press, London.

Shearer, C. (1996). User driven data mining. In *Unicom Data Mining Conference*.

Simon, H. (1985). Information-processing theory of human problem-solving. In Aitkenhead, A. and Slack, J., editors, *Issues in Cognitive Modeling*, pages 253–278. Lawrence Erlbaum Ass., London.

Wielinga, B., Schreiber, A., and Breuker, J. (1992). Kads: A modelling approach to knowledge engineering. special issue the kads approach to knowledge engineering“. *Knowledge Acquisition*, 4(1):5–53.

Wirth, R., Shearer, C., Grimmer, U., Reinartz, T., Schloesser, J., Breitner, C., Engels, R., and Lindner, G. (1997). Towards process-oriented tool support for kdd. In *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97)*.