Wissuwa, Stefan

Working Paper

Data Mining und XML: Modularisierung und Automatisierung von Verarbeitungsschritten

Wismarer Diskussionspapiere, No. 12/2003

Provided in Cooperation with:
Hochschule Wismar, Wismar Business School

Suggested Citation: Wissuwa, Stefan (2003) : Data Mining und XML: Modularisierung und Automatisierung von Verarbeitungsschritten, Wismarer Diskussionspapiere, No. 12/2003, ISBN 3910102379, Hochschule Wismar, Fachbereich Wirtschaft, Wismar

This Version is available at:
https://hdl.handle.net/10419/39826

Hochschule Wismar
Fachhochschule für Technik, Wirtschaft und Gestaltung
Fachbereich Wirtschaft

Hochschule Wismar
University of Technology, Business and Design
Department of Business

Stefan Wissuwa

Data Mining und XML.
Modularisierung und Automatisierung von Verarbeitungsschritten

Heft 12 / 2003

W D P

Wismarer Diskussionspapiere / Wismar Discussion Papers

Der Fachbereich Wirtschaft der Hochschule Wismar, Fachhochschule für Technik, Wirtschaft und Gestaltung bietet die Studiengänge Betriebswirtschaft, Management sozialer Dienstleistungen, Wirtschaftsinformatik und Wirtschaftsrecht an. Gegenstand der Ausbildung sind die verschiedenen Aspekte des Wirtschaftens in der Unternehmung, der modernen Verwaltungstätigkeit im sozialen Bereich, der Verbindung von angewandter Informatik und Wirtschaftswissenschaften sowie des Rechts im Bereich der Wirtschaft.

Nähere Informationen zu Studienangebot, Forschung und Ansprechpartnern finden Sie auf unserer Homepage im World Wide Web (WWW): http://www.wi.hs-wismar.de/.

Herausgeber: Prof. Dr. Jost W. Kramer
Fachbereich Wirtschaft
Hochschule Wismar
Fachhochschule für Technik, Wirtschaft und Gestaltung
Philipp-Müller-Straße
Postfach 12 10
D – 23966 Wismar
Telefon: ++49 / (0)3841 / 753 441
Fax: ++49 / (0)3841 / 753 131
e-mail: j.kramer@wi.hs-wismar.de

ISSN 1612-0884
ISBN 3-910102-37-9

JEL-Klassifikation C80, Z00

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung und Motivation

Die rasante Entwicklung der Computer- und Messtechnologie in den letzten Jahren und die damit einhergehende Steigerung der Möglichkeiten zur Datengewinnung und Datenspeicherung hat dazu geführt, dass die Kluft zwischen der Menge verfügbarer Daten und dem Verständnis dieser Daten immer grösser wird. In nahezu allen Bereichen des heutigen Lebens werden ständig Daten gesammelt und gespeichert: in der Forschung, Wirtschaft, Handel, beim Einkaufen oder wenn wir im Internet surfen. Ob diese Daten nun gezielt erhoben werden oder als Nebenprodukt anfallen - das in ihnen enthaltenen Wissen kann (oder könnte) vielfältig genutzt werden. Egal, ob die Wissensgewinnung nun dem reinen Selbstzweck dient oder durch konkrete praktische Problemstellungen begründet ist, stellt das gewonnene Wissen in jedem Fall ein wertvolles Kapital da. Leider sind die Datenmengen und erst recht die Möglichkeiten, die sich durch Kombination mit bereits existierendem Wissen ergeben, viel zu umfangreich und zu komplex, als das Menschen sie bewältigen könnten. Erst durch Computer und spezielle Softwareprogramme haben wir die Möglichkeit, diese Datenmenge adäquat zu verarbeiten. Das Data Mining stellt für diesen Zweck Methoden bereit.

In den letzten Jahren ist Data Mining zu einem Schlagwort geworden, das, nicht zuletzt beflügelt durch Werbeaussagen der Hersteller entsprechender Software, neue Erkenntnisse und Problemlösungen für jeden versprach. Data Mining kann zwar die Analyse großer Datenmengen erleichtern oder gar erst ermöglichen, dennoch gehört zum Lösen eines Problems mehr dazu, als eine Data-Mining-Software zu erwerben und sie mit Daten zu füttern. Vielmehr ist das Wissen um die Methoden und die richtige Anwendung selbiger entscheidend, um verwertbare Resultate zu erhalten. Um Data Mining zielorientierter als die bisher weitläufig praktizierte Versuch-und-Irrtum-Methode anzuwenden, ist eine detaillierte Erforschung der Methoden und die Auswirkungen der Daten-Vorverarbeitung nötig, um generelle Handlungsgrundsätze und Richtlinien aufstellen zu können. Die Anwendung des Data Mining ist daher in der Praxis mit vielen Problemen behaftet.

Dennoch sind die Anwendungsgebiete für Data Mining außerordentlich vielfältig. Sie lassen sich sowohl in der Forschung zur Analyse umfangreicher Messdaten einsetzen als auch zur Lösung praktischer Probleme, z.B. im Betriebswirtschaftlichen Umfeld. Der Abschnitt 1.2 verschafft dem interessierten Leser einen Überblick über einige praktische Anwendungsmöglichkeiten aus den Bereichen Forschung, Betriebswirtschaft und Technik und soll einen Eindruck des breit gefächerten Spektrums an Möglichkeiten vermitteln.

Diese Arbeit ist ebenfalls durch eine praktische Anwendung motiviert. Sie entstand im Rahmen der Forschungstätigkeit des Autors im Landesforschungsschwerpunkt “Genomorientierte Biotechnologie” des Landes Mecklenburg Vorpommern und stellt den aktuellen Entwicklungsstand dar. Die im Projekt zu analysierten Daten erforderten eine neue Herangehensweise, da vor allem die Einsatzmöglichkeiten von Künstlichen Neuronalen Netzen untersucht werden sollten. Schnell stellte sich heraus, dass die mangelhafte Unterstützung dieser Technologie durch gängige Data-Mining-Programme auch auf andere Methoden des Data Mining zutraf. Daher wurde mit der Entwicklung eines eigenen Data-Mining-Systems begonnen, dass eine bessere Unterstützung der Experimente ermöglicht. Im Folgenden wird ein flexibles System zur Automatisierung von Data-Mining-Experimenten vorgestellt, um den Data-Mining-Prozess zu beschleu-

nigen und dem Anwender zeitraubende Routinearbeiten abzunehmen. Das System besteht aus eigenständigen Modulen zur Durchführung verschiedener Data-Mining-Algorithmen sowie einem auf XML basierenden Protokoll für die Prozesskommunikation und die Darstellung von Data-Mining-Abläufen (Workflows). Die offene Architektur des Systems erlaubt eine ständige Weiterentwicklung durch neue Methoden und begünstigt damit den Einsatz in der Forschung, während es sich durch seine hohe Flexibilität, die einfache Anwendung und die Möglichkeit zur Automatisierung auch für den praktischen Einsatz und die automatische Verarbeitung von Massendaten eignet.

1.1 Data Mining

Als ein Teilgebiet des Knowledge Discovery in Databases (KDD) befasst sich das Data Mining mit der Gewinnung von (neuem) Wissen aus vorhandenen Daten. Wie aus dem Englischen Wort "Mining" abzuleiten ist, handelt es sich, bildlich gesprochen, um das Graben und Suchen in großen Datenmengen, um darin versteckte interessante Sachverhalte aufzuspüren und neue Erkenntnisse zu gewinnen. Die in den Daten implizit enthaltenen Informationen werden durch Anwendung verschiedener Data-Mining-Verfahren extrahiert und in explizites, und damit nutzbares, Wissen umgewandelt. Dieses Wissen kann z.B. durch Regeln oder Diagramme ausgedrückt und zur Lösung von Problemstellungen herangezogen werden.

Als "Data Mining" wird sowohl die Anwendung der Methoden auf Daten, als auch das Forschungsgebiet bezeichnet. Data Mining ist ein stark experimentelles Forschungsgebiet. Dazu gehört sowohl die Erforschung und Entwicklung der Data-Mining-Methoden selbst als auch die Untersuchung, wie diese in der Praxis angewandt werden können. Data Mining als Anwendung ist ein Prozess, der in mehreren Phasen abläuft. Eine strikte Trennung einzelner Phasen ist dabei kaum möglich, da sie sich teilweise überschneiden und aufeinander aufbauen. Es kann folgende grobe Unterteilung erfolgen:

- Daten-Vorverarbeitung (Aufbereitung, Kodierung)
- Analyse
- Daten-Nachverarbeitung (Auswertung, Visualisierung)
- Interpretation

Dem Data-Mining-Prozess vor- und nachgelagert sind die Datenerhebung sowie die Validierung. Beide sind stark anwendungsbezogen und daher nur bedingt dem Data Mining selbst zuzuordnen. Jedoch haben sie großen Einfluss auf den Erfolg des Data Mining. Die Datenerhebung hat dafür Sorge zu tragen, dass die einen Sachverhalt beschreibenden Informationen auch in der Datenmenge enthalten sind und andererseits irrelevante, aber signifikante Informationen die Datenmenge nicht verfälschen. Ebenso können ungenaue, fehlende oder falsche Werte das Ergebnis beeinträchtigen oder unbrauchbar machen.

Die Validierung dient der Überprüfung des Data-Mining-Ergebnis auf tatsächliche Korrektheit. Dies geschieht meist durch Anwenden des erstellten Modells auf ausgewählte Testdaten.

Für ein erfolgreiches Data Mining sind innerhalb der einzelnen Phasen mehrere Faktoren entscheidend:

1. Die *Vorverarbeitung* dient der Aufbereitung der verfügbaren Daten für die nachfolgenden Analysen. Neben allgemein notwendigen Verarbeitungsschritten wie z.B. Import, Auswahl der Attribute oder Datenbereinigung gehört dazu vor allem die Datenkodierung, die speziell für die gewählte Analysemethode vorzunehmen ist. Eine geeignete Vorverarbeitung ist entscheidend für den Erfolg oder Misserfolg des Data Mining, da die Wahl ungeeigneter Methoden die Daten verfälschen kann. Da die Vorverarbeitung 75- 85% des gesamten Zeitaufwands beanspruchen kann [1], ist dies ein Ansatz für eine Rationalisierung der Experimente.

2. Die *Analyse* umfasst die eigentliche Anwendung der Data-Mining-Methoden. Es existiert zwar eine Vielzahl mehr oder weniger gut dokumentierter Algorithmen, dennoch gibt es keine allgemeingültigen Ansätze, wie diese unter welchen Bedingungen und unter Verwendung welcher Parameter auf eine bestimmte Art von Daten anzuwenden sind, um ein gewünschtes Ergebnis zu erhalten. Die in der Literatur beschriebenen Fälle sind auf spezielle Anwendungsgebiete zugeschnitten und daher nicht allgemeingültig. Zudem wird die Wahl der Methoden selten motiviert.

3. Die *Nachverarbeitung* bildet die Schnittstelle zwischen Data-Mining-Methode und Wissensgewinnung. Sie dient der Darstellung der Data-Mining-Ergebnisse und deren Überführung in explizites Wissen, z.B. in Form von Regeln. Dies kann automatisch oder durch Interaktion mit einem Anwender geschehen. In letzterem Fall ist die Transformation der Data-Mining-Ergebnisse in eine menschengerechte, in der Regel grafische Darstellungsform notwendig, gegebenenfalls unter Berücksichtigung des jeweiligen Anwendungsgebietes.

1.2 Anwendungen

Die Anwendungsmöglichkeiten des Data Mining sind außerordentlich vielfältig. Es lassen sich drei grundlegende Formen der Anwendung unterscheiden:

1. Klassifikation

2. Assoziation

3. Clustering

Klassifikation ist ein überwachtes Lernverfahren, bei dem anhand bereits klassifizierter Beispiele gelernt wird, unbekannte Beispiele ebenfalls korrekt zu klassifizieren. Eine typische Anwendung ist die Schrifterkennung. Als Methoden können zum Beispiel Entscheidungsbäume oder Neuronale Netze zum Einsatz kommen.

[1] [Alp00] S. 7

Die Assoziation unterscheidet sich von der Klassifikation dadurch, dass nicht nur die Klasse, sondern beliebige Attribute und Attributkombinationen vorhergesagt werden können. Diese Aufgabe kann ebenfalls sehr gut durch Neuronale Netze gelöst werden.

Das Clustering ist ein Verfahren zur Bildung von Gruppen (Cluster) einander ähnlicher Elemente. Es kann verwendet werden, um die Attribute herauszufinden, die wesentliche Merkmale einer Gruppe darstellen oder durch die sie sich von anderen Gruppen unterscheiden. Eine der bekanntesten Formen eines Clustering-Algorithmus sind die von Teuvo Kohonen entwickelten Selbstorganisierenden Karten[2]. Die mathematisch motivierten Support Vector Machines stellen ebenfalls einen erfolgversprechenden Clustering-Ansatz dar.

Die praktischen Anwendungsmöglichkeiten des Data Mining sind vielfältig und sollen anhand ausgewählter Beispiele aufgezeigt werden. Für weitere Anwendungsfälle sei auf [Alp00, Nak98] verwiesen.

1.2.1 Genexpressions-Analyse

Die Anwendung von Data-Mining-Methoden, speziell der künstliche Neuronalen Netze, zur Analyse von Daten, die durch die Differentiellen Genexpression gewonnen werden, stellt einen Schwerpunkt unserer Forschungstätigkeit im Rahmen des Förderprogramms "Genomorientierte Biotechnologie" des Landes Mecklenburg Vorpommern dar.

Durch die Entwicklung der Microarray-Technologie ist es erstmals möglich geworden, die Aktivitäten tausender Gene gleichzeitig zu messen. Dies geschieht indirekt durch Messung der Konzentration der entsprechenden m-RNA. Diese Methode erlaubt es, einen biologischen "Fingerabdruck" von Zellen zu erzeugen. Ziel ist es, den Zusammenhang und die Funktionsweise von Genen genauer erforschen zu können. Moderne Microarrays sind in der Lage, über 11000 Gene gleichzeitig zu Messen. Die dabei entstehenden Datenmengen entziehen sich aufgrund ihres Umfangs und ihrer Komplexität der Analyse durch einen Menschen. Hier sind spezielle Computerprogramme notwendig, um die Massendaten zu handhaben. Auf dem Markt ist bereits eine Reihe von Software verfügbar, die für diesen Zweck mehr oder weniger umfangreiche Data-Mining-Methoden bereitstellt. Allerdings leidet diese ebenfalls an den unter Punkt 1.4 dargestellten Problemen.

Bei der Anwendung von Data-Mining-Methoden in der Biotechnologie, aber auch in anderen Bereichen, ist es notwendig, Domänenwissen einzubinden, um eine effektive Analyse der Data-Mining-Ergebnisse zu ermöglichen. In der Praxis kommt es leider immernoch vor, dass Forscher umfangreiche und zeitaufwändige Recherchen manuell vornehmen müssen, die jedoch technisch leicht integrierbar wären.

1.2.2 Marketing und Verkauf

Marketing und Verkauf gehören wohl zu den dem Data Mining gegenüber aufgeschlossensten Anwendungsgebieten. Unternehmen verfügen heute über sehr viele Informationen über Kunden, Lieferanten und Geschäftsabläufe. Sie umfassen beispielsweise das Kauf- und Zahlungsverhalten, Reaktion auf Werbung und sogar das soziale Umfeld wie Familienstand, Wohnge-

[2] auch: SOM; Self-Organizing Map; Kohonen Feature Map

gend oder Bildungsniveau. Daraus resultierend gibt es viele Zielstellungen, die durch Data Mining erreicht werden können. Hier zwei Beispiele:

- Die Warenkorbanalyse dient der Optimierung der Angebotspalette hinsichtlich Umfang und strategischer Platzierung in Regalen oder Katalogen. Dabei sind nicht nur die absoluten Verkaufszahlen von Bedeutung, sondern z.B. auch die Kombination verschiedener Produkte.
- Beim Direktmarketing werden Reaktionsprofile von Personen oder Haushalten anhand bisheriger Reaktionen auf Werbung erstellt und zur Identifikation potentieller Ziele von Werbemaßnahmen genutzt. Dadurch sollen z.B. Streuverluste minimiert und so die Kosten von Werbeaktionen gesenkt werden, während gleichzeitig die Erfolgsquote erhöht wird.

1.2.3 Bildanalyse

Bildanalyse wird auf viele Themenbereiche wie z.B. Schrift- und Gesichtserkennung, Auswertung von Radar- und Satellitenbildern oder die Erkennung von Bewegungen angewandt. Dies erfolgt größtenteils durch den Einsatz künstlicher Neuronaler Netze. Durch überwachte Lernverfahren wird dem Neuronalen Netz anhand von Beispielen das gewünschte Verhalten, z.B. die Erkennung von Buchstaben oder Gesichtern auf Pixelbildern (Muster), antrainiert. Ein trainiertes Netz kann nun zur Erkennung unbekannter Muster eingesetzt werden. Die Stärke der Verfahren liegt darin, auch Muster, die sich vom ursprünglich gelernten unterscheiden, z.B. durch Unschärfe, Rauschen oder abweichende Form, innerhalb gewisser Toleranzen korrekt zu klassifizieren. Diese Methoden lassen sich auf viele andere Gebiete anwenden und werden z.B. auch in der optischen Qualitätskontrolle von Produkten eingesetzt.

1.2.4 Web Mining

Als Web-Mining wird die Anwendung des Data Mining auf Web-basierte[3] Daten bezeichnet. In [GC01] werden drei Anwendungsbereiche unterschieden, die sich durch die Art der analysierten Informationen unterscheiden:

1. Web Content Mining
2. Web Usage Mining
3. Web Structure Mining

[3]Dies bezieht sich auf alle Informationen, die direkt oder indirekt mit der Nutzung des Internet zusammenhängen, wobei hauptsächlich die Informationen gemeint sind, die über das http-Protokoll übertragen werden. In abgewandelter Form lässt sich dies auch auf andere Netzwerk-bezogene Bereiche übertragen, wie z.B. Auslastung von Servern oder Teilen von Firmennetzwerken.

Als Web Content Mining wird die Analyse von Inhalten bezeichnet, die z.B. als Text oder Grafik vorliegen können. Ziel ist es, Aussagen über die Thematik von Web-Inhalten zu treffen und sie zu katalogisieren. Diese Verfahren werden in der Praxis extensiv eingesetzt. Das wohl bekannteste Beispiel sind Internet-Suchmaschinen.

Web Usage Mining bezieht sich auf die Analyse von Nutzungs- und Zugriffsdaten von Web-Auftritten, um das Nutzerverhalten zu erforschen. Das Ziel ist die Optimierung des Web-Auftritts in vielerlei Hinsicht, wie z.B. die zielgruppengerechte Ausrichtung, thematische Strukturierung und Navigation oder dem Ladeverhalten.

Das Web Structure Mining hat schließlich die topographische Struktur der durch Hyperlinks verbundenen Web-Sites als Untersuchungsgegenstand. Es wird ebenfalls von vielen Internet-Suchmaschinen angewendet, um z.B. die Relevanz von Sites zu beurteilen.

1.3 Auswirkungen des Data Mining

Data Mining ermöglicht das Erkennen komplexer Zusammenhänge in großen Datenmengen. Dadurch stehen Informationen zur Verfügung, deren Nutzung viele Fragen aufwirft. Es ist zu klären, ob die Nutzung rechtlich, moralisch und ethisch vertretbar ist, wobei sich die Frage nach der rechlichen Zulässigkeit am ehesten beantworten lässt. Den gleichen Fragen hat sich nicht nur die Nutzung der durch Data Mining gewonnenen Erkenntnisse zu unterwerfen, sondern es betrifft bereits die Erhebung der Daten. Da das Gebiet Datenschutz sehr umfangreich ist und nicht zum Thema dieser Arbeit gehört, sollen hier mögliche Probleme nur kurz angerissen werden, um die generelle Problematik zu verdeutlichen. Für weitere Informationen sei auf einschlägige Literatur, insbesondere das Bundes-Datenschutz-Gesetz verwiesen.

1.3.1 Rechtliche Aspekte

Dem Rechtlichen Aspekt der Erhebung und Verarbeitung von Daten widmen sich die Datenschutzgesetze des Bundes und der Länder, im Folgenden einfach Datenschutzgesetz genannt. Es hat die Aufgabe, das Recht des Einzelnen zu schützen, grundsätzlich selbst über Preisgabe und Verwendung seiner Daten zu bestimmen. Dies betrifft alle Daten, die mit einer Person in Verbindung gebracht werden können, sogenannte Personenbezogene Daten. Dies sind “Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren Person.” (§1 Abs. 3 Satz 1 BDSG) Der “bestimmbare Person” kommt eine nicht unerhebliche Bedeutung zu, denn selbst scheinbar unverfängliche Informationen lassen sich durch Kombination einzelnen Individuen zuordnen. So ließe sich zum Beispiel durch folgende Angaben: Informatiker, 25 bis 35 Jahre Alt, Wohnhaft in Norddeutschland und Mitglied in einem Segelclub, den Personenkreis von zunächst vielleicht einigen Tausend auf ein Minimum oder vielleicht sogar eine Einzelperson reduzieren.

Durch Anonymisierung lässt sich jedoch ein Datenbestand derart verändern, dass er datenschutzrechtlichen Bestimmungen genügt. Jedoch sind hier ebenfalls Maßnahmen erforderlich, die eine Deanonymisierung verhindern. Auch aus einem solchen Datenbestand lassen sich erfolgreich wichtige Informationen ableiten. Beispielsweise können für bestimmte Kundengrup-

pen wichtige verkaufsrelevante Informationen gewonnen werden. Die Zuordnung einzelner Personen zu diesen Gruppen ist wiederum datenschutzrechtlich problematisch.

Ein weiteres Problem stellt die Nutzung von Daten dar, die ursprünglich für andere Zwecke erhoben wurden. Mit nur wenigen Ausnahmen ist die schriftliche Zustimmung der betroffenen Personen für jeden Verwendungszweck einzuholen, sowohl bei der Ersterhebung als auch für jede weitere Nutzung, falls sie über die usprünglich geplante hinausgeht.

1.3.2 Ethisch-Moralische Aspekte

Neben den rechtlichen Bestimmungen ist zusätzlich zu berücksichtigen, inwiefern die Nutzung gewonnener Erkenntnisse moralisch vertretbar ist. Ethisch-moralische Aspekte sind zwangsläufig dann zu berücksichtigen, wenn Data Mining auf Daten angewendet wird, die direkt oder indirekt mit Menschen oder ihrem Verhalten zu tun haben. Die Nutzung von Data-Mining-Techniken bedeutet, dass die Nutzung der Daten weit über das hinausgehen kann, was bei der Aufnahme der Daten ursprünglich geplant war. (Witten, 36) Das prinzipielle Problem dabei ist, dass durch Data Mining komplexe Zusammenhänge erkannt werden können, die ein Mensch niemals oder nur mit extremen Aufwand entdeckt hätte, so dass die Gesellschaft bzw. der Mensch durch die vermeindliche “Intelligenz” von Computern faktisch überrumpelt wird.

Ein gutes Beispiel stellen die durch die Genforschung oder in der Medizin bereits oder zukünftig gewonnenen Erkenntnisse dar. Zum einen kann es für die Betroffenen einen Vorteil bedeuten, wenn genetische Veranlagungen für Krankheiten erkannt werden können und so die präventive Behandlungen ermöglicht wird. Andererseits kann es einen Nachteil bedeuten, da diese Veranlagung potentiell ein Ausschlusskriterium bei Versicherungen oder Arbeitgebern sein könnte.

Einen weiteren Aspekt stellt die Glaubwürdigkeit der Data-Mining-Ergebnisse dar. Komplexe Zusammenhänge anhand einiger weniger Indikatoren absolut korrekt zu erkennen, ist nicht möglich. So kann es zu scheinbar eindeutigen Ergebnissen kommen, die jedoch Falsch sind und lediglich Eigenheiten der erhobenen Datenmenge repräsentieren, die durch Verwendung von für das Problem eigentlich irrelevanter Informationen hervorgerufen wurden, die jedoch durch das Data Mining zwangsläufig ebenfalls Berücksichtigung finden. Ein fiktives Beispiel wäre, wenn erkannt würde, dass rote Autos besonders häufig in Wildunfälle verwickelt sind, wenn der Fahrer einer bestimmten ethnischen Gruppe angehört. Wie aussagekräftig wäre dieses Ergebnis und wäre dessen Nutzung vertretbar?

1.4 Probleme

Die Verfügbarkeit von Data-Mining-Methoden ist sehr hoch. Jedoch ist Verfügbarkeit nicht mit Anwendbarkeit gleichzusetzen. Obwohl es bereits viele kommerzielle und auch nicht-kommerzielle Data-Mining-Anwendungen gibt, treten ständig neue Problemstellungen auf, für die jedes Mal ein neuer Lösungsansatz entwickelt werden muss. Die Gründe liegen sowohl auf technischer als auch auf methodischer Ebene.

Eine große Zahl Data-Mining-Verfahren sind bereits hinreichend bekannt und auch in der Literatur detailliert beschrieben, wogegen die praktische Anwendung der Verfahren weiter-

hin Probleme bereitet. Daher liegt das Ziel unserer Arbeit im Rahmen des Landesforschungsschwerpunktes nicht in den Verfahren selbst, sondern in der Erforschung der Anwendungsmöglichkeiten und -bedingungen.

1.4.1 Technische Probleme

Es existiert eine Vielzahl von Data-Mining-Anwendungen, angefangen bei teilweise sehr teuren Produkten bis hin zu freier Software. Trotz dieses Spektrums treten ständig neue Problemstellungen auf, für deren Lösung die vorhandene Software schlichtweg untauglich ist. Mangelnde Transparenz und Verfügbarkeit der Methoden sowie unzureichende Unterstützung von Experimenten, die über das reine Anwenden von Methoden hinausgehen, machen sie für einen wissenschaftlichen Einsatz untauglich. Andere erfüllen diese Kriterien weitestgehend, sind aber so kompliziert und umfangreich, dass sie von Fach-Anwendern nicht bedient werden können. Zu den größten Problemen gehört somit die Diskrepanz zwischen den Fähigkeiten der Software und den Anforderungen und Fähigkeiten der Anwender. Es lassen sich grob zwei Szenarien für den Einsatz von Data-Mining-Anwendungen unterscheiden:

1. Experten-Szenario: Die für Data-Mining-Experten entwickelte Software stellt eine große Vielfalt unterschiedlicher Verfahren bereit. Durch komplexe Parametrisierung lassen sich diese Verfahren optimal auf eine Problemstellung anpassen. Die Entwickelten Lösungen können jedoch nur bedingt in der Praxis eingesetzt werden, da die Software ein hohes Maß an Expertenwissen voraussetzt, das von “normalen” Anwendern und Experten anderer Fachrichtungen in der Regel nicht zu erwarten ist und teils sehr hohe Einarbeitungszeiten verlangt.

2. Anwender-Szenario: Domainspezifische Software bietet eine gute Unterstützung des Anwenders bei leichten bis mittelschweren Data-Mining-Problemen. Damit einhergehend nimmt die Anzahl verfügbarer Methoden stark ab. Die Möglichkeiten der Konfiguration sowie die Transparenz der einsetzbaren Verfahren ist ebenfalls begrenzt, was die Software für einen wissenschaftlichen Einsatz disqualifiziert. Daher lässt sich diese Software nur für einfache Analysen einsetzen, die Verfahren können nicht ausgereizt werden.

1.4.2 Methodische Probleme

Nicht nur die Einsetzbarkeit verfügbarer Data-Mining-Anwendungen ist mit Problemen verbunden, sondern auch die Methoden des Data Mining selbst. In der Literatur wird zwar wiederholt gezeigt, wie sich ein bestimmtes Problem durch den Einsatz von Data Mining lösen lässt, jedoch sind diese Lösungen auf einen eng begrenzten Problemkreis beschränkt. Überwachte Lernverfahren zum Beispiel benötigen in der Regel sehr viele aufbereitete Trainingsdaten, um ein Problem zu lösen. Ändern sich die zu verarbeitenden Daten geringfügig, ist oft ein komplett neues Modell notwendig. Ähnlich verhält es sich bei unüberwachten Lernverfahren. Hier ist das entstehende Modell sehr stark von den verwendeten Daten abhängig. Zwei Modelle, selbst wenn sie mit thematisch gleichen Daten erzeugt wurden, sind oft vollkommen verschieden.

Es gibt keine allgemeingültigen Herangehensweisen oder Handlungsgrundsätze, anhand derer zur Lösung von neuen Problemstellungen vorgegangen werden kann. Data Mining wird oft nach dem Versuch-und-Irrtum-Prinzip durchgeführt, was keinesfalls effizient sein kann. Ziel sollte es sein, während der Anwendung des Data Mining Informationen über die verwendeten Daten und Methoden zu sammeln, um zu ermitteln, welche Eigenschaften Daten aufweisen müssen, damit sie durch bestimmte Methoden erfolgreich verarbeitet werden können. Wenn es gelingt, wenigstens grundlegende Merkmale zu finden, ließe sich das Versuch-und-Irrtum Prinzip stark einschränken und die Anwendung des Data Mining erfolgreicher und vor allem zeiteffizienter gestalten.

2 Anforderungen

Im Folgenden sollen grundlegende Anforderungen an ein flexibles System zur Durchführung von Data-Mining-Experimenten dargestellt werden. Der Schwerpunkt des Einsatzes liegt in der Eigenschaft des Systems als Experimentierwerkzeug, da primär die Eignung verschiedener Methoden des Data Mining sowie Möglichkeiten und Auswirkungen der Datenvorverarbeitung untersucht werden sollen. Die mit diesem System entwickelten Lösungen sollen direkt in der Praxis angewandt werden können.

Die Anforderungen an eine Data-Mining-Software lassen sich aus den Kriterien für gute Software ableiten, wie sie in der Softwaretechnik beschrieben werden: Flexibilität, Funktionalität, Stabilität, Verständlichkeit und Effizienz. In der Literatur werden teils abweichende Definitionen verwendet, die Kernaussage ist jedoch dieselbe. Dabei ist zu beachten, dass die einzelnen Punkte nicht losgelöst voneinander betrachtet werden können. Vielmehr gibt es Überschneidungen und gegenseitige Abhängigkeiten. Durch die Eigenschaft des Systems als Experimentierwerkzeug hat vor allem die Flexibilität einen hohen Stellenwert.

2.1 Flexibilität

Die Flexibilität bezieht sich auf die Fähigkeit der Software, unter verschiedenen Bedingungen einsetzbar zu sein. Es können dabei sowohl technische als auch methodische Anforderungen variieren.

Um sowohl die experimentelle als auch praktische Anwendbarkeit der Methoden zu gewährleisten, müssen die Voraussetzungen geschaffen werden, um den unterschiedlichen Ansprüchen zu genügen:

- Anwendbarkeit: ein Data-Mining-Experte muss detaillierte Konfigurationen der Methoden durchführen können, um deren Leistungsfähigkeit voll auszunutzen, denn der Untersuchungsgegenstand sind die Methoden selbst. Ein Endanwender muss dagegen in der Lage sein, auch ohne umfangreiche Data-Mining-Kenntnisse und langwierige Einarbeitung Datenanalysen erfolgreich durchzuführen.

- Effizienz der Bedienung / Anwendung: wiederholt auftretende Arbeitsschritte müssen gespeichert werden können, um bei Bedarf automatisch und somit zeitsparend ausgeführt

zu werden. Dies gilt ebenfalls für komplexe Konfigurationsvorgänge der Methoden.

- Praxistauglichkeit: Im Rahmen des praktischen Einsatzes muss das System komplexe, definierte Data-Mining-Abläufe automatisch durchführen können, um die Verarbeitung von Massendaten zu vereinfachen. Auch das semi-automatische Durchführen von Data-Mining-Experimenten im Rahmen vorher definierter Kriterien ist zu ermöglichen.

- Erweiterbarkeit: das System muss sich leicht anpassen und um neue Methoden erweitern lassen. Dies ist zum einen notwendig, um die Aktualität des Systems sicherzustellen. Zum anderen dient es der effizienten Entwicklung, da die zeitnahe Fertigstellung eines funktionsfähigen Systems wichtig ist. Eine Auswahl der wichtigsten Methoden genügt, um erste Resultate zu erzielen. Weitere Methoden können zu einem späteren Zeitpunkt nach Bedarf ergänzt werden.

Die technische Flexibilität des Systems soll das Zusammenspiel mit anderen Soft- und Hardware-Systemen sicherstellen.

- Kompatibilität: der Datenaustausch mit vorhandenen Anwendungen muss ermöglicht werden. Entsprechend umfangreiche Methoden für den Im- und Export von Daten müssen bereitgestellt werden.

- Portabilität: das System muss mit wenig Aufwand an gängige Rechnerarchitekturen und Betriebssysteme angepasst werden können.

2.2 Funktionalität

Funktionalität, oder auch Korrektheit, bedeutet, dass die Software die an sie gestellten Funktionalen Anforderungen exakt erfüllt. Für den geplanten experimentell-wissenschaftlichen und praktisch-wissenschaftlichen Einsatzbereich muss die Software folgende Eigenschaften aufweisen:

- Transparenz der Methoden: Die Funktionsweise der zur Verfügung stehenden Methoden muss für den Anwender transparent und nachvollziehbar sein. Nur durch Kenntnis der Eigenschaften einer Methode ist gewährleistet, dass die Plausibilität eins Ergebnisses sichergestellt werden kann.

- Umfang der Methoden: Es müssen Methoden aus den Bereichen Datenbereinigung, Datenkodierung, Datenanalyse (Data Mining) und Nachverarbeitung (z.B. Visualisierung) zur Verfügung stehen.

- Bei der Durchführung komplexer Experimente ist es notwendig, die Nachvollziehbarkeit der Ergebnisse und Arbeitsabläufe zu gewährleisten. Dazu ist ein Mechanismus notwendig, der die durchgeführten Experimente bei Bedarf automatisch und detailliert protokolliert. Dies führt gleichzeitig zur Beschleunigung der Experimente, da dem Experten zeitaufwändige Protokollierungsarbeiten abgenommen werden.

- Die Integration externer Informationen ist für die Auswertung der Analyseergebnisse essentiell. Es müssen Metainformationen mit den Daten verknüpft werden können, um diese in einen Kontext zu bringen und mit “Inhalt” zu versehen.

- Die praktische Nutzbarkeit des Systems ist nicht zuletzt von der Domainspezifischen Ausrichtung abhängig. Das bedeutet, dass für den Anwender eine Umgebung geschaffen wird, die ihm das Arbeiten mit den ihm vertrauten Begriffen ermöglicht.

2.3 Stabilität

Das korrekte Funktionieren des Systems auch unter Ausnahmebedingungen (wie z.B. Falsche oder fehlende Eingaben, Fehlfunktion in Modulen) muss soweit sichergestellt sein, dass kein Datenverlust eintritt. Dies kann zum Beispiel durch Zerlegung des Systems in unabhängige Module erreicht werden. Die Anforderungen an die Stabilität eines Data-Mining-Systems sind ansonsten die selben wie für jede andere Software auch sollen daher hier nicht weiter erläutert werden.

2.4 Verständlichkeit

Die Verständlichkeit des Systems bezieht sich sowohl auf die Funktionen als auch die Nutzerführung. Wie auch unter Punkt 2.2 erläutert, beinhaltet dies die Transparenz der Data-Mining-Methoden, speziell hinsichtlich der Klarheit ihrer Bedeutung.

Die Nutzerführung ist für die praktische Nutzbarkeit des Systems von Bedeutung. Sie umfasst unter anderem die logische und klare Gliederung von Menüs und Dialogen sowie die Eindeutigkeit von Begriffen. Nicht zuletzt ist sie von der Domainspezifischen Anpassung der Software abhängig. Für den Anwender muss eine Umgebung bereitgestellt werden, die das Arbeiten mit den ihm vertrauten Begriffen ermöglicht.

2.5 Effizienz

Die Effizienz des Systems betrifft sowohl die Anwendbarkeit (siehe Abschnitt 2.2) des Systems als auch den angemessenen Umgang mit Systemressourcen. Die Hauptkriterien sind Komplexität und Speicherbedarf der Methoden, wobei letzteres zunehmend durch ständig steigende verfügbare Speicherkapazität in den Hintergrund tritt und erst bei sehr Umfangreichen Datenmengen wieder an Bedeutung gewinnt. Die Komplexität ist jedoch weiterhin von großer Bedeutung, da sie ein Ausschlusskriterium darstellt, das die Anwendung bestimmter Methoden auf Daten, die einen gewissen Umfang überschreiten, mit sowohl der aktuellen Computertechnik als auch auf absehbare Zeit unmöglich macht. Hier besteht die Aufgabe darin, durch effiziente Implementierung der Algorithmen die Komplexität auf ein Minimum zu reduzieren.

Ein effizienter Umgang mit Systemressourcen bedeutet auch, auf unnötigen Ballast zu verzichten und die Anforderungen für die Ausführbarkeit auf einem Computersystem auf ein notwendiges Maß zu reduzieren. Konkret bedeutet dies, dass nur die für die jeweilige Aufgabe notwendigen Komponenten des Systems zum Einsatz kommen. Beispielsweise kann für eine

Batch-Verarbeitung oder beim parallelen Rechnen auf mehreren Systemen auf eine grafische Benutzeroberfläche verzichtet werden. Dadurch wird der Einsatz des Systems auch auf weniger leistungsstarken Rechnern ermöglicht.

2.6 Kompatibilität

Die Sicherstellung der Kompatibilität ist ein weiterer wichtiger Aspekt. Die Notwendigkeit, den Datenaustausch mit anderen Anwendungen, die oft als Datenquelle dienen, sicherzustellen, bedarf sicher keiner weiteren Erklärung und sollte selbstverständlich sein. Auch der Im- und Export von Data-Mining-Modellen, zum Beispiel in PMML[4], ist zu berücksichtigen.

Ein oft vernachlässigter Aspekt ist auch die Kompatibilität des Systems mit sich selbst, d.h. die Sicherstellung der Kompatibilität zwischen verschieden weit entwickelten Systemen. Zwei Systeme können sich dabei in zweifacher Hinsicht voneinander unterscheiden:

1. Versionsspezifische Unterschiede
2. Ausstattungsspezifische Unterschiede

Versionsspezifische Unterschiede entstehen durch Weiterentwicklung z.B. Methoden oder der Architektur des Systems. Ausstattungsspezifische Unterschiede resultieren aus der Möglichkeit, ein System (fast) beliebig um neue Komponenten erweitern zu können bzw. es um Komponenten zu reduzieren. In jedem Fall muss die Nutzung bestehender Daten soweit gewährleistet sein, wie das aktuelle System die benötigte Funktionalität unterstützt. Das Fehlen eines Moduls bzw. das Vorhandensein einer älteren oder neueren Version darf nicht zur völliger Unbrauchbarkeit von Daten und somit der bisherigen Ergebnisse führen.

3 Ein Konzept für ein Modulares Data-Mining-System

Im Folgenden wird ein Modell für die Durchführung von Data-Mining-Experimenten vorgestellt. Dabei wird auf eine Architektur aus der Unix-Welt zurückgegriffen, das sich in der Praxis sehr gut bewährt hat: das System besteht nicht aus einer einzelnen monolithischen Applikation, sondern aus einer Vielzahl eigenständiger Programme, die eine genau definierte Aufgabe erfüllen.

3.1 Motivation

Die Modularisierung von Software ist ein gängiges Verfahren zur Erhöhung der Flexibilität und Stabilität. Die Umsetzung erfolgt meist in Form einer Haupt- oder Kernkomponente, die durch das Einbinden von Modulen, die in Form von Funktionsbibliotheken vorliegen, zur Laufzeit

[4] Predictive Model Markup Language; ein XML-Basierte Beschreibungssprache für Data-Mining-Modelle; Homepage im Internet unter: http://www.dmg.org

um Funktionen erweitert werden kann. Die Module sind dabei nicht vollkommen autark, sondern von der einbindenden Software abhängig, da sie ihr untergeordnet sind. Sie definiert die Schnittstellen und somit die Art der möglichen Funktionen, die ein Modul übernehmen kann.

Ein weiteres Konzept der Modularisierung, das aus der Unix-Welt stammt, ist der Aufbau eines Systems aus einer Vielzahl vollkommen eigenständiger Elemente, deren Zusammenspiel das System ergeben. Ein Unix-System besteht aus einem Kernel, der lediglich Basisfunktionen bereitstellt und der das eigentliche Betriebssystem darstellt, sowie sehr vielen kleinen Programmen, die auf eine bestimmte Aufgabe spezialisiert sind. Zusammen bilden sie das, was wir unter einem Unix-System verstehen.

Dieses Konzept bietet eine Reihe von Vorteilen und lässt sich auch zur Realisierung eines Data-Mining-Systems nutzen. Dabei werden sämtliche Methoden durch einzelne Programme realisiert, die untereinander Daten austauschen (können). Die Aufteilung der Funktionalität auf mehrere eigenständige Programme bietet viele Vorteile, z.B:

- Jedes Modul ist unabhängig von anderen Modulen nutzbar. Da die Module eigenständige Programme sind, können sie z.B. in Skripten verwendet werden.
- Durch Verteilung der Module auf mehrere Rechner lässt sich sehr leicht ein einfaches Parallel-Computing realisieren.
- Es wird ressourceneffizientes Arbeiten ermöglicht, da nur die wirklich benötigten Module Speicher und Rechenzeit beanspruchen. Auch leistungsschwache Rechner können so sinnvoll in das Data Mining einbezogen werden.
- Das System wird leichter Skalierbar, da jederzeit neue Module hinzugefügt werden können.
- Das Gesamtsystem ist sehr fehlertolerant, da die Module sich nicht gegenseitig beeinflussen können.
- Die Datensicherheit wird erhöht, da der Datenfluss zwischen den Modulen nach Bedarf umgelenkt werden kann und so jederzeit eine Sicherung des Arbeitsfortschritts ermöglicht wird.
- Module können bei Bedarf sehr komplex sein. Auf diese Weise ist auch die Einbindung anderer Applikationen möglich, sofern diese einen Automatismus (z.B. Skripting) unterstützen. Ein Modul dient dabei der Kapselung der Applikation und stellt nach Außen hin dessen gesamten Funktionsumfang zur Verfügung. Der von uns entwickelte System-Prototyp (siehe Abschnitt 4) bindet auf diese Weise den SNNS[5] ein, der eine sehr große Auswahl an Architekturen und Lernverfahren unterstützt.
- Auf das herkömmliche Einbinden von Funktionsbibliotheken, gerade bei Verwendung wenig komplexer oder bereits implementierter Algorithmen, muss nicht verzichtet werden, da ein Modul auch diese Aufgabe übernehmen kann.

[5] Stuttgarter Neuronale Netze Simulator; Entwickelt am Institut für Parallele und Verteilte Höchstleistungsrechner IPVR der Universität Stuttgart; Literatur: [Zel00]

- Die Implementierung der Module kann in der jeweils optimalen Programmiersprache erfolgen.

3.2 SXML Module

SXML-Module sind Programme, deren Ein- und Ausgabeströme dem SXML-Protokoll genügen.

Durch die sehr allgemein gehaltene Definition kann prinzipiell jeder Algorithmus als SXML-Modul implementiert werden. Sie definiert die zwei wichtigsten Merkmale des SXML-Systems: die Modularität und das SXML-Protokoll.

Da sich die Methoden der Datenvorverarbeitung, Analyse und Nachverarbeitung auf ein Black-Box-Modell zurückführen lassen und sich aus informationstechnischer Sicht nicht voneinander unterscheiden, ist die Spezifikation von SXML so ausgelegt, dass prinzipiell alle Module miteinander verbunden werden können. Komplexe Data-Mining-Abläufe lassen sich durch Verkettung von Modulen realisieren.

3.3 SXML Protokoll

Das SXML-Protokoll definiert Syntax und Grammatik der Ein- und Ausgabeströme von SXML-Modulen. Prinzipiell sind Ein- und Ausgabeströme kompatibel, d.h. ein Ausgabestrom kann einem anderen Modul als Eingabe dienen. Das SXML-Protokoll übernimmt vier Aufgaben:

1. Konfigurationsdefinition eines Moduls
2. Konfiguration eines Moduls
3. Übertragen der Daten
4. Definition von Verarbeitungs-Richtlinien

Das SXML-Protokoll ist dynamisch erweiterbar. Das bedeutet, dass die Syntax eines Eingabestroms erst während seiner Verarbeitung endgültig festgelegt wird. Das Protokoll definiert lediglich die Grundstruktur, die durch jedes Modul individuell erweitert werden kann. Das SXML-Protokoll wird im folgenden detailliert erläutert.

3.3.1 Konfigurationsdefinition

Um Data-Mining-Experimente zu automatisieren ist es notwendig, dass ein SXML-Modul seiner Umwelt mitteilen kann, auf welche Art es zu konfigurieren ist. Dies erfolgt durch eine formale Beschreibung seiner Parameter und Restriktionen, z.B.

- Art und Anzahl der Parameter
- zulässige Wertebereiche

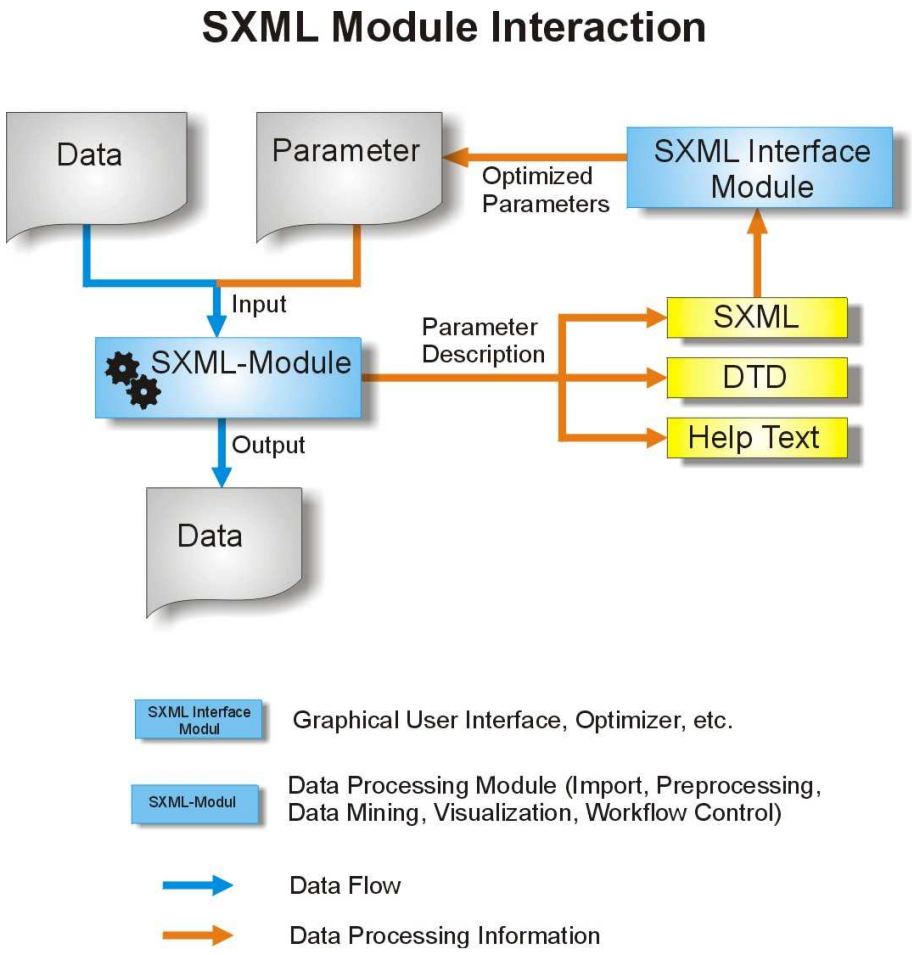

Abbildung 1: Interaktion von SXML-Modulen

- Vorauswahl
- Standardwerte etc.

Abhängig vom Empfänger dieser Informationen findet eines der folgenden Formate Anwendung:

- Als *Hilfe-Text* um die Anwendung der Module von der Kommandozeile aus zu unterstützen.
- Als *Document Type Definition (DTD)* für die Kompatibilität mit XML
- Als *XML* in einem Format ähnlich den XML-Schemas. Diese Informationen können von einem anderen Programm (bzw. Modul) verwendet werden, um eine automatische Konfiguration vorzunehmen, und so z.B. Parameter zu optimieren. Eine Einbindung des Moduls in eine Benutzeroberfläche ist so ebenfalls möglich.

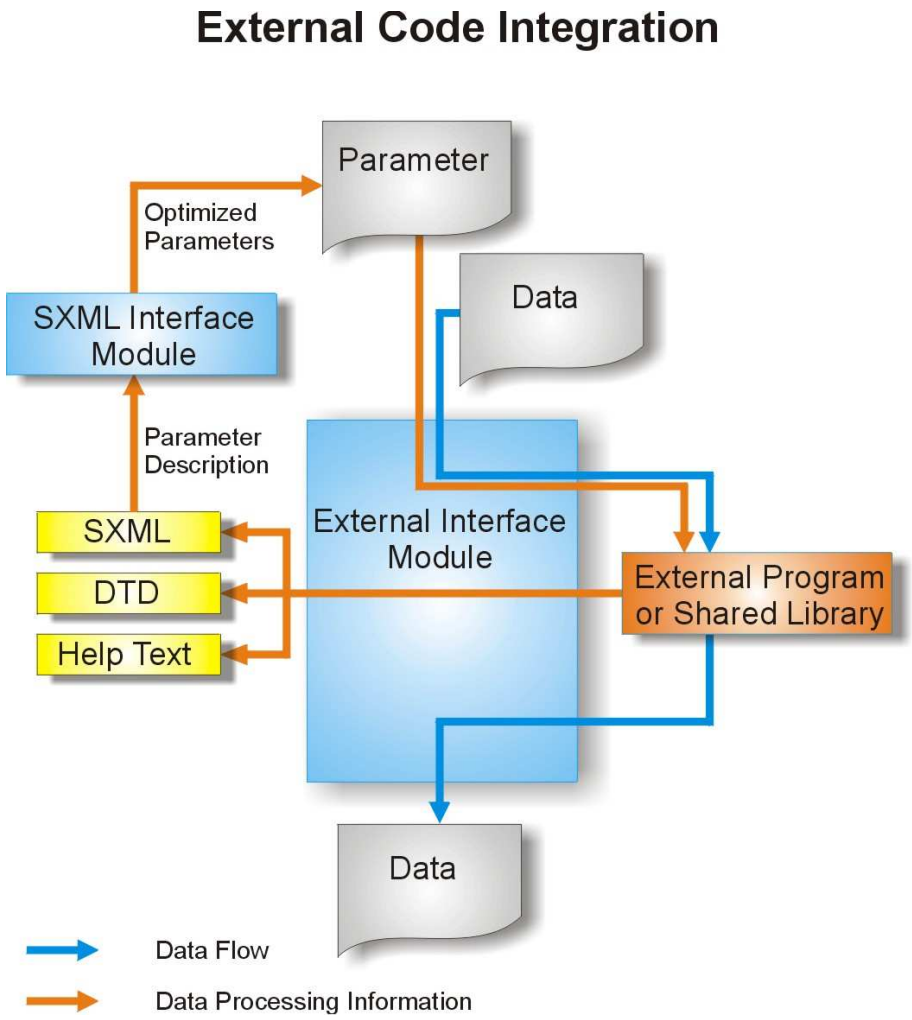

Abbildung 2: Kapselung

3.3.2 Konfiguration

Die Konfiguration eines Moduls geschieht entweder über Kommandozeilen-Parameter in der Form:

```
parameter[.attribut]=Wert
```

oder in einem Konfigurationsabschnitt des Eingabestroms als XML, unterhalb eines <*config*>-Knotens. Die dabei zu verwendenden Bezeichner für Parameter und Attribute sind nicht vorgegeben, sondern können von jedem Modul frei definiert werden (siehe 3.3.1). Da das SXML-Protokoll dynamisch ist, ist es möglich, dass Parameter erst während der Verarbeitung des Eingabestroms in Abhängigkeit der Werte anderer Parameter definiert werden.

Zur Verdeutlichung des Sachverhaltes soll das Modul *enc_lib* des System-Prototypen dienen. Es wurde für die Kapselung von Kodierungsfunktionen, die als Shared Library vorliegen,

entwickelt. In Abhängigkeit von der gewählten Library wird die Konfigurationsdefinition zur Laufzeit um andere Parameter erweitert. Das folgende Listing zeigt die DTD des Moduls. Neben den vordefinierten Parametern “config”, “data” und “p”, die zum SXML-Protokoll gehören, unterstützt das Modul zunächst nur einen Parameter “lib”, der die zu verwendende Library angibt:

```
> enc_lib -dtd
<!ELEMENT config (lib)>
<!ELEMENT lib (#PCDATA)>
<!ELEMENT data (p*)>
<!ELEMENT p (#PCDATA)>
```

Sobald die zu verwendende Library dem Modul bekannt ist, also eine Parameterzuweisung erfolgt, wird sie geladen und es werden bei Bedarf weitere Parameter hinzugefügt. Im Fall der Verwendung der Library *enc_lib_range.so* für die Kodierung von Wertebereichen erweitert sich die Konfigurationsdefinition um den Parameter “attr” mit den Attributen “from” und “to” für Ober- und Untergrenze des Wertebereichs:

```
> enc_lib lib=enc_lib_range.so -dtd
<!ELEMENT config (lib, attr*)>
<!ELEMENT lib (#PCDATA)>
<-- Beginn der neuen Elemente-->
<!ELEMENT attr (#PCDATA)>
<!ATTLIST attr
  from CDATA " " #REQUIRED
  to CDTATA " " #REQUIRED
>
<-- Ende -->
<!ELEMENT data (p*)>
<!ELEMENT p (#PCDATA)>
```

Ein Konfigurationsabschnitt könnte dann wie folgt aussehen:

```
<config>
  <lib>enc_lib_range.so</lib>
  <attr from=1 to=3>A</attr>
  <attr from=4 to=7>B</attr>
  <attr from=8 to=10>C</attr>
  ...
</config>
```

Konfigurations-Abschnitte und Daten-Abschnitte können sich beliebig oft abwechseln. Ein einmal verwendeter Parameter behält seine Gültigkeit, bis er überschrieben wird. Wird ein Parameter über die Kommandozeile festgelegt, so kann dieser nicht überschrieben werden.

3.3.3 Datenspeicherung und -übertragung

Daten werden satzweise innerhalb eines Daten-Abschnitts abgelegt. Die verwendeten Knoten sind:

- <*data*> für einen Datenabschnitt
- <*p*> für einen Datensatz

In jedem Datenabschnitt kann ein Konfigurationsabschnitt enthalten sein, der eine Kopie der originalen Konfiguration desjenigen Moduls ist, das die Daten erzeugt hat. Dies dient der Protokollierung und Nachvollziehbarkeit von Ergebnissen. Metainformationen (siehe auch 6) können durch Attribute jedem Datenabschnitt und Datensatz zugeordnet werden. Diese sind prinzipiell frei definierbar. Vom SXML-Protokoll vordefinierte Attribute sind:

- *id* : zur eindeutigen Kennzeichnung eines Datenabschnitts/Datensatzes
- *src* : der Dateiname des Moduls, von dem die Daten stammen
- *src_name* : der Name des Moduls, von dem die Daten stammen
- *type* : Bezeichner des Datentyps; Syntax: [Typ|Separator[|Dimension...]]

3.3.4 Referenzierung

Die speziell bei der Erstellung von Workflows erforderliche Weiterleitung und Verteilung von Daten an Module geschieht durch Namensräume. Jeder Datenstrom lässt sich über einen eindeutigen Namen ansprechen. Datenströme können aus SXML-Dateien bestehen oder die Ausgabe von Modulen sein. Teile eines Datenstroms können extrahiert und an Module geleitet werden. Ebenso lassen sich mehrere Datenströme zusammenfassen.

Die Benennung von Datenströmen erfolgt durch das Attribut *id*, das entweder in einem Datenabschnitt

```
<data id="name">
```

vorkommen kann, oder einem Modul zugeordnet wird:

```
<proc cmd="modul" id="Daten">
```

Einem Modul kann eine Datenquelle durch das Attribut *data* zugewiesen werden:

```
<proc cmd="modul" data="Daten">
```

Datenströme können durch Angabe eines Datenfeldes aufgesplittet werden. Dies kann wahlweise durch Angabe des Spaltennamens oder der Spaltennummer erfolgen:

```
<proc cmd="modul1" data="MeineDaten:Datumsfeld">
<proc cmd="modul2" data="MeineDaten:#1">
<proc cmd="modul3" data="MeineDaten:#2">
```

Ebenso lassen sich Datenströme zu neuen zusammenfassen:

```
<proc cmd="modul1" id="NeueDaten:#2">
<proc cmd="modul2" id="NeueDaten:#1">
<proc cmd="modul3" id="NeueDaten:#3">
```

Im letzten Fall bestehen die Datensätze des neuen Datenstroms “NeueDaten” aus den Ausgabedaten von modul2+modul1+modul3 in dieser Reihenfolge.

Falls Module als Quelle für Datenströme dienen, müssen sie Definiert werden, bevor eine Referenzierung stattfindet. Für Datenströme in Datenabschnitten gilt diese Beschränkung nicht, d.h. ein aktives Modul wartet so lange auf die zugewiesenen Daten, bis sie im Eingabestrom auftreten.

4 Der SXML Prototyp

Das vorgestellte Konzept wird in einem Prototypen umgesetzt und beständig erweitert. Die Entwicklung erfolgt unter Linux unter GNU C/C++, die Wahl der Programmiersprache ist jedoch prinzipiell frei. Es existieren Module für die Prozess-Steuerung[6], den Datenimport, die Vorverarbeitung sowie ein Schnittstellenmodul für die Integration des Stuttgarter Neuronale Netze Simulators SNNS für die Datenanalyse.

Durch die Modularität sowie den jeweils klar definierbaren Rahmen für ein Modul eignet sich die Erweiterung des Systems sehr gut als Projektarbeit, z.B. für Studentische Hilfskräfte oder Praktikanten.

4.1 Implementation von SXML Modulen

Module können prinzipiell in jeder beliebigen Programmiersprache implementiert werden. Um die Entwicklung von SXML-Modulen so einfach wie möglich zu gestalten, wurden alle protokollspezifischen Funktionen in einer C++ - Klassenbibliothek gekapselt, die auch durch den Prototypen des Systems verwendet wird. Für andere Programmiersprachen müssen entsprechende Routinen selbst entwickelt werden, da hier vorerst keine Unterstützung besteht. Bei Nutzung der Bibliothek für die Implementierung eigener Module sind keine speziellen Kenntnisse des SXML-Protokolls erforderlich. Folgendes gilt daher für die Verwendung von C/C++ für die Implementierung von Modulen.

Da einige Funktionen des SXML-Protokolls, wie z.B. die Konfiguration, programmtechnisch sehr eng mit den zu implementierenden Algorithmen zusammenhängen, sind einige Regeln bei der Entwicklung von SXML-Modulen zu beachten:

[6] Siehe dazu Kapitel 5

1. Jedes SXML-Modul muss eine von SXML_IO abgeleitete Klasse besitzen. Diese Klasse übernimmt sämtliche Funktionen des SXML-Protokolls.
2. Die Kontrolle muss beim Programmstart dieser Klasse übergeben werden.
3. SXML_IO stellt virtuelle Methoden bereit, die während der Verarbeitung des Eingabestroms aufgerufen werden, um bestimmte Ereignisse zu signalisieren. Diese können (und sollen) bei Bedarf überschrieben werden, um auf die Ereignisse je nach Bedarf zu reagieren.

Das folgende Listing stellt die grundlegende Struktur eines SXML-Moduls dar. Die Implementierung eines SXML-Programms ist sehr stark vom jeweiligen Algorithmus abhängig. Um ein verdecken der SXML-Typischen Konstrukte durch den Algorithmus zu vermeiden, wurde auf ein vollständiges Beispiel verzichtet und Pseudo-Code verwendet:

```
class MySXMLClass: public SXML_IO{
  // constructor
  MySXMLClass(){
    doSomeInitialization();
    registerParameter(...);
    registerAttribute(...);
  }
  // signal handler
  void paramChange() { ... }
  void configStart() { ... }
  void configEnd()   { ... }
  void dataStart()   { ... }
  void dataEnd()     { ... }
  // data processing
  bool pcdata(...){ doSomething(); writeOutput(); }
}
int main(int argc, char * argv[]){
  MySXMLClass io;
  io.args(argc,argv);
  return io.parse(); // transfer control to SXML_IO
}
```

4.1.1 Konfiguration

Die Konfiguration eines SXML-Moduls wird vollständig und ausschließlich durch SXML_IO vorgenommen. Das bedeutet, dass die durch Kommandozeile oder Eingabestrom übergebenen Werte der Parameter in Variablen gespeichert werden. Dazu muss ein SXML-Modul seine Parameter vollständig registrieren. Dies geschieht durch Verwendung der Methoden *registerParam()* bzw. *registerAttribute()*. Der Aufruf beider Methoden ist identisch, mit der Ausnahme,

dass *registerAttribute()* einen zusätzlichen Parameter für die Zuordnung des Attributes zu einem Parameter benötigt. Die Syntax lautet:

```
registerParam(
  Parameter Name,
  Parameter Tag,
  Parameter Description,
  Default Value / Value List,
  Number of Occurences,
  Parameter Type | Input Type,
  Destination Variable)
```

Parameter Name: definiert den Namen eines Parameters, wie er z.B. in einem Dialogfeld verwendet werden kann.

Parameter Tag: definiert das zu verwendende XML-Tag.

Parameter Description: kann einen kurzen Hilfetext enthalten.

Default Value/Value List: Hier kann eine durch "|" getrennte Liste möglicher Werte angegeben werden, die der Parameter annehmen kann. Der erste Wert ist jeweils der Standardwert.

Number of Occurences: definiert, wie oft ein Parameter maximal auftreten darf. Auf diese Weise können Listen übergeben werden.

Parameter Type|Input Type: hier kann der Datentyp des Parameters festgelegt werden, ebenso die Möglichkeiten, wie die Eingabe erfolgen kann, z.B. freie Eingabe, Einfache oder Mehrfache Auswahl. Der *Input Type* legt fest, wie die Werte der *Value List* interpretiert werden.

Destination Variable: die Programmvariable, in der die Speicherung des Parameters erfolgt.

Für registrierte Parameter werden bei Bedarf durch SXML_IO Konfigurationsdefinitionen erzeugt und ausgegeben. Ebenso werden die den Parametern zugeordneten Variablen automatisch mit Werten belegt. Dabei wird eine Umwandlung in folgende Datentypen unterstützt:

- Text (char*)
- Ganzzahl (int)
- Fließkommazahl (float,double)
- Boolsche Werte (bool)

- Aufzählungen von Werten der genannten Datentypen (vector[char*|int|float|bool])

Während der Verarbeitung von Konfigurations-Abschnitten werden folgende virtuelle Funktionen aufgerufen:

- configStart() : signalisiert den Beginn eines neuen Konfigurationsabschnitts
- configEnd() : signalisiert das Ende des aktuellen Konfigurationsabschnitts
- paramChange() : signalisiert die Änderung eines Parameters

Diese Methoden können bei Bedarf überschrieben werden. Besonders paramChange() ist sehr vielseitig Einsetzbar und eignet sich z.B. für die Übergabe mehrerer Parameter des selben Typs. In diesem Fall wird jedes Mal paramChange() aufgerufen und der entsprechende Parameter kann z.B. in eine Liste eingetragen werden. Die dem Parameter zugeordnete Variable dient hierbei als Puffer. Diesen Mechanismus nutzt z.B. das Modul *enc_lib_range*, um die Zuordnung mehrerer Wertebereiche zu Mustern zu ermöglichen. Die Puffer-Variable *dummy* ist dabei dem Parameter “attr” und dessen Attributen “from” und “to” zugeordnet:

```
void paramChange(char * pname){
  if(pname!=NULL){
    if(strcmp(pname,"attr")==0) EncodingTable->push_back(dummy);
  }
}
```

4.1.2 Datenübergabe

Die Datenübergabe geschieht durch folgende Methoden:

- startData() : signalisiert den Beginn eines neuen Datenabschnitts
- endData() : signalisiert das Ende des aktuellen Datenabschnitts
- pcdata() : wird aufgerufen, um den aktuellen Datensatz zu übergeben

4.2 Module des Rahmensystems

Neben Modulen für die Eigentliche Datenverarbeitung sind weitere Module erforderlich, um folgende Aufgaben zu übernehmen:

- Datenimport
- Datenexport
- Prozess-Steuerung

Module für den Im- und Export von Daten verhalten sich wie normale SXML-Module, mit der Ausnahme, dass entweder Eingabe- oder der Ausgabestrom nicht SXML-Konform sein müssen.

Module für die Prozess-Steuerung dienen dem Starten und Verbinden von anderen Modulen. Dies ist ein sehr komplexer Vorgang und wird daher separat behandelt.

5 Automatisierung von Data Mining Abläufen

Für die Steuerung komplexer Verarbeitungshierarchien bzw. Workflows wurde das SXML-Modul *io_process* entwickelt. Es ist in der Lage, andere Module zu starten sowie deren Ein- und Ausgabeströme untereinander zu verbinden und somit den Datenfluss zu steuern. Da es selbst ein Modul ist, kann es auch weitere Instanzen seiner selbst ausführen und somit auch ganze Workflows miteinander verknüpfen. Auf die genaue Funktionsweise soll hier nicht eingegangen werden.

5.1 Darstellung von Workflows

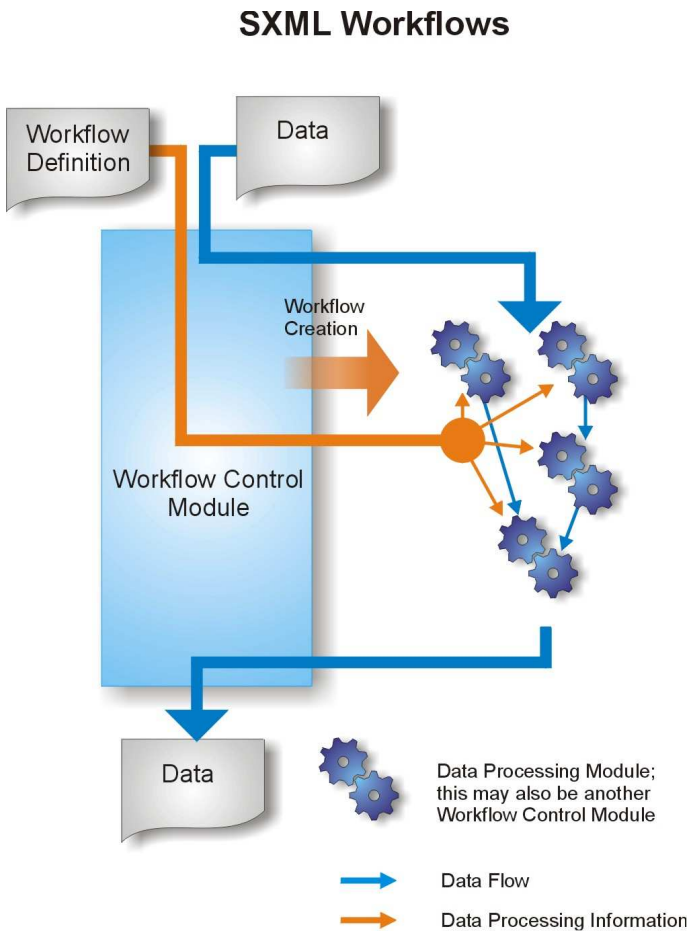

Abbildung 3: SXML Workflows

Da ein Parameter beliebige Daten enthalten kann, ist es möglich, ganze Konfigurations-Abschnitte als Parameter zu übergeben. Die somit mögliche Schachtelung von Konfigurationen erlaubt die

Darstellung komplexer Workflows. Eine Schematische Darstellung der Schachtelung und der daraus resultierende Workflow sind in Abbildung 4 zu sehen.

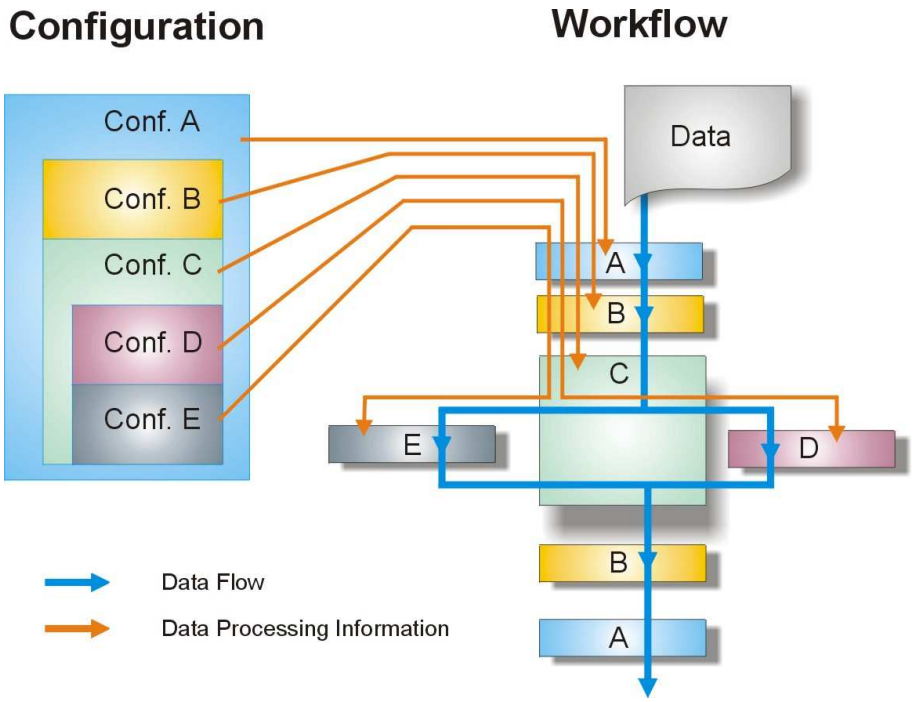

Abbildung 4: Schachtelung von Konfigurationen

5.2 Beispiel

Die Erstellung eines Workflows soll anhand eines Beispiels erläutert werden, das häufig verwendet wird: auf einem Golfplatz wird anhand von Wettermerkmalen entschieden, ob gespielt wird oder nicht. Es stehen Informationen über Bewölkung, Temperatur, Luftfeuchtigkeit und Windgeschwindigkeit an verschiedenen Tagen bereit und die jeweilige Entscheidung der Jury, ob gespielt wird oder nicht, ist bekannt. Ziel ist es, aus den vorhandenen Fakten Regeln abzuleiten, um die Entscheidung in Zukunft durch einen Computer fällen zu lassen.

Die Daten liegen im ARFF-Format in der Datei “wetter.arff” vor:

```
@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {true, false}
@attribute whatIdo {will_play, may_play, no_play}

@data
```

```
overcast,75,55,false,will_play
sunny,85,85,false,will_play
sunny,80,90,true,may_play
overcast,83,86,false,no_play
rainy,70,96,false,will_play
...
```

Diese muss zunächst Importiert werden. Dazu kann wird Tabellen-Importmodul *imptab* verwendet werden. Die Konfiguration des Moduls lautet: (Auszug)

```
<config>
  <sep>,</sep>
  <ignorespc>true</ignorespc>
  <header>0</header>
  <export>1,2,3,4,5</export>
  ...
</config>
```

Das Modul erzeugt nun folgende Ausgabe:

```
<data src="imptab">
  <p>overcast,75,55,false,will_play</p>
  <p>sunny,85,85,false,will_play</p>
  ...
</data>
```

Diese Daten sollen nun Spaltenweise in Binäre Muster für ein Neuronales Netz kodiert werden. Die Konfiguration des Moduls für diskrete Werte lautet entsprechend:

```
<config>
  <lib>enc_lib_direct.so</lib>
  <attr val="overcast">0 0 1</attr>
  <attr val="sunny">0 1 0</attr>
  ...
</config>
```

Die Temperaturwerte können z.B. als Wertebereiche kodiert werden. Die Konfiguration des Moduls lautet entsprechend:

```
<config>
  <lib>enc_lib_range.so</lib>
  <attr from=60 to=70>0 0 1</attr>
  <attr from=71 to=80>0 1 0</attr>
  ...
</config>
```

Die Importierten Daten müssen an die Module weitergeleitet werden. Dazu wird der Ausgabestrom des Import-Moduls mit “Wetter” benannt. Die Ausgabeströme der Kodierungs-Module sollen zu einem Datenstrom namens “Pattern” zusammengefügt werden. Anschließend sollen der Datenstrom “Pattern” an ein Neuronales Netz geleitet werden. Die komplette Konfiguration für diesen Workflow lautet wie folgt:

```
<config>
  <proc cmd=”imptab” id=”Wetter”>
    <tabstart>7</tabstart>
    <sep>,</sep>
    <ignorespc>true</ignorespc>
    <header>0</header>
    <export>1,2,3,4,5</export>
    ...
  </proc>
  <proc cmd=”enc_lib” data=”Wetter:1” id=”Pattern:1”>
    <lib>enc_lib_direct.so</lib>
    <attr val=”overcast”>0 0 1</attr>
    <attr val=”sunny”>0 1 0</attr>
    ...
  </proc>
  <proc cmd=”enc_lib” data=”Wetter:2” id=”Pattern:2”>
    <lib>enc_lib_range.so</lib>
    <attr from=60 to=70>0 0 1</attr>
    <attr from=71 to=80>0 1 0</attr>
    ...
  </proc>
  <proc cmd=”neuralnet” data=”Pattern”>
    <type>Feed Forward</type>
    <learn>Backpropagation</learn>
    ...
  </proc>
</config>
```

Dieser Konfiguration kann nun in einer Datei als Workflow gespeichert werden, z.B. datamining_wetter.sxml. Durch Verwendung des Moduls *process* für die Prozess-Steuerung kann dieser Workflow ausgeführt, auf die vorhandenen Daten angewandt und das Resultat z.B. in einer Datei gespeichert werden:

```
> cat wetter.arff | process datamining_wetter.sxml
  > ergebnis.sxml
```

Das so gewonnene Ergebnis kann manuell inspiziert oder wiederum als Eingabestrom an ein weiteres SXML-Modul geleitet werden, das z.B. eine Bewertung des Ergebnisses oder eine Vi-

sualisierung vornimmt. Für das konkrete Beispiel würde sich z.B. eine Visualisierung anbieten, die je nach Ergebnis ein Rotes oder Grünes Signal gibt.

6 Zusammenfassung und Ausblick

Diese Arbeit basiert auf der Forschungsarbeit des Autors in dem vom Land Mecklenburg-Vorpommern im Rahmen des Landesforschungsschwerpunktes “Genomorientierte Biotechnologie” geförderten Projektes “Analyse von Proteomdaten mittels Künstlicher Neuronaler Netze” an der Hochschule Wismar. Es ist zugleich die Grundlage nachfolgender Projekte im Rahmen des Themenschwerpunktes “Data Mining Engineering”.

Das vorgestellte System behebt die aus wissenschaftlicher Sicht mangelnde Funktionalitäten herkömmlicher Data-Mining-Anwendungen und schließt die Kluft zwischen Data-Mining-Experten und Endanwendern, indem es sowohl extrem umfangreiche Anpassungen erlaubt und gleichzeitig durch das Erzeugen vordefinierter Workflows in beliebiger Komplexität bei Bedarf sehr einfach anzuwenden ist. Durch den geringen Ressourcenaufwand erlaubt es die Analyse umfangreicher Datenmengen auch in kleinen Labors und Unternehmen.

Das vorgestellte XML-Basierte Format für die Prozesskommunikation und die Datenspeicherung bietet eine gute Basis für weitere Entwicklungen. Es ist flexibel genug, um nahezu beliebige Informationen aufzunehmen und mit den Daten zu verknüpfen, wie zum Beispiel Metadaten und Kontextbezogenes Wissen. Im Rahmen weiterer Forschungen wird ein Konzept entwickelt werden, das zunächst auf die Integration Bio-Medizinischer Daten ausgerichtet ist.

Die weitere Entwicklung des Prototypen zielt auf die Ausweitung der verfügbaren Data-Mining-Methoden und die Enwicklung einer modularen grafischen Benutzeroberfläche, die domain-spezifische Anpassungen erlaubt, um das System praktisch einsetzen zu können. Dabei wird eine enge Zusammenarbeit mit bisherigen und künftigen Projektpartnern angestrebt, um eine möglichst hohe praktische Anwendbarkeit zu gewährleisten.

Einige zukünftige Entwicklungsschwerpunkte sollen im Folgenden kurz vorgestellt werden.

Integration von Domänenwissen

Für eine erfolgreiche Anwendung des Data Mining ist es notwendig, Meta-Informationen mit den Daten zu verknüpfen. Dies können sowohl Informationen über die Struktur der Daten und Zusammenhänge / Abhängigkeiten von Attributen sein, als auch kontextbezogenes Wissen über die Daten. Ersteres ist vor allem für die Verarbeitung der Daten durch Data-Mining-Methoden relevant. Letzteres stellt eine wichtige Vorraussetzung für die Interaktion mit Anwendern dar, da diese Informationen es dem Anwender erlauben, losgelöst von der abstrakten Daten-Sicht mit den ihm vertrauten Begriffen zu arbeiten. Wie sich dies mit dem vorgestellten System umsetzen lässt, wird im Folgenden beschrieben. Als Beispiel dient die Anwendung des Systems in der Biotechnologie zur Analyse von Genexpressionsmustern. Eine Übertragung auf andere Bereiche ist prinzipiell möglich.

Das Ziel der Integration von Domänenwissen ist es, die Auswertung der durch das Data Mining gewonnenen Modelle durch Anwender zu erleichtern und ihm Routinearbeiten ab-

zunehmen. Bisher ist ein ständiges “Umschalten” des Anwenders zwischen der Data-Mining-Software, gedruckter Literatur und Online-Datenbanken notwendig, um die Ergebnisse des Data Mining auszuwerten und zu überprüfen. Es ist zwar sehr schön, wenn die Data-Mining-Software die Datensätze irgendwelchen Clustern zuordnet. Das wirklich interessante ist jedoch, warum dies geschieht und welche charakteristischen Eigenschaften die einzelnen Cluster aufweisen. Man kann nun die Datensätze der Cluster untersuchen und wird feststellen, dass z.B. in einem Cluster ein bestimmtes Attribut immer einen bestimmten Wertebereich aufweist. Ohne Kontextinformationen ist dieses Ergebnis jedoch Wertlos, da der Wert an sich nichts aussagt. Vielmehr muss die Bedeutung, die Aussage des Wertes und des zugehörigen Datensatzes ermittelt werden. Durch Hinzuziehen von Literatur oder Online-Datenbanken kann durch zeitaufwändige manuelle Recherche nun ermittelt werden, dass Datensätze dieses Clusters z.B. zu Genen gehören, die sich alle auf demselben Chromosom befinden, eine bestimmte Gruppe von Proteinen kodieren oder einer bekannten Erbkrankheit zugeordnet werden. Allein durch diese Informationen ist der Anwender in der Lage, das Ergebnis des Data-Mining-Experiments fachbezogen zu Beurteilen und seine Schlussfolgerungen zu ziehen. Das Ziel ist es, diese Informationen schnell verfügbar zu machen und dem Anwender so die manuellen Recherche- und Vergleichsarbeiten abzunehmen. Dazu müssen diese Informationen mit den Daten verknüpft werden können und Methoden bereitgestellt werden, die diese Informationen auswerten. So können auch solche Informationen berücksichtigt werden, die ein Mensch niemals berücksichtigen könnte, beispielsweise komplette Gen-Sequenzen.

Die Integration von Domänen-Wissen kann durch Erweiterung der Daten- und Datensatz-Knoten um weitere Attribute geschehen. Diese können Schlüssel enthalten, die von speziellen Modulen ausgewertet werden und für weitere Aktionen wie z.B. das Suchen in einer Online-Datenbank herangezogen werden. Eine entsprechende Implementierung für genomorientierte Informationen ist derzeit in der Entwicklung.

Benutzeroberfläche und Visualisierung

Für die Anwendung der Data-Mining-Komponenten durch den Endanwender ist eine Grafische Benutzeroberfläche notwendig. Die Interaktionsmöglichkeiten mit den Modulen sind mit Ausnahme der Kommandozeilen-Optionen nicht auf die Manuelle Nutzung durch den Anwender ausgelegt, sondern Zielen auf die Integration mit anderen Modulen. Durch die Möglichkeit, umfangreiche Konfigurationsinformationen automatisch bereitzustellen ist es möglich, eine automatische Anpassung der Nutzeroberfläche an neue Module zu realisieren und somit die Entwicklung neuer Module zu entkoppeln. Das nächste Ziel ist daher die Entwicklung eines (grafischen) Konfigurations-Tools, das für jedes beliebige Modul genutzt werden kann. Anschließend kann darauf aufbauend eine Domain-Spezifische Anpassung umgesetzt werden, vorzugsweise nach dem selben Mechanismus.

Nicht zuletzt sind umfangreiche Visualisierungsmöglichkeiten in Form von Modulen bereitzustellen, um eine menschengerechte Auswertung der Data-Mining-Ergebnisse zu ermöglichen.

Portierung

Das System ist bisher nur in C/C++ verfügbar und auf die Anwendung unter Linux ausgelegt. Um eine breitere Anwendungsbasis zu schaffen, ist eine Portierung der Basisbibliotheken nach Win32 und Solaris notwendig. Die Entwicklung von Bibliotheken oder Interfaces für weitere Programmiersprachen, vor allem Java und Perl, stellen weitere anzustrebende Ziele dar.

Literatur

[Alp00] ALPAR, PAUL: *Data Mining im praktischen Einsatz*. Vieweg, 2000.

[GC01] GEORGE CHANG, ET.AL.: *Mining the World Wide Web*. Kluwer Academic Publishers, 2001.

[HL01] H. LIU, H. MOTODA: *Instance Selection and Construction for Data Mining*. Kluwer Academic Publishers, 2001.

[IW01] I.H. WITTEN, E. FRANK: *Data Mining*. Hanser, 2001.

[Koh01] KOHONEN, TEUVO: *Self-Organizing Maps*. Springer, 2001.

[Leu01] LEUNG, M.L. WONG; K.S.: *Data Mining using Grammar Based Genetic Programming and Applications*. Kluwer Academic Publishers, 2001.

[Nak98] NAKHAEIZADEH, G.: *Data Mining - Theoretische Aspekte und Anwendungen*. Physica-Verlag, 1998.

[PM00] P. MERTENS, H.W. WIECZORREK: *Data X Strategien*. Springer, 2000.

[Zel00] ZELL, ANDREAS: *Simulation Neuronaler Netze*. Oldenbourg, 2000.

Anhang

A: Konfigurationsdefinitionen

Die Konfigurationsdefinition der Module können zum einen sehr umfangreich sein und sind in der Regel nicht für den Menschen gedacht. Sie sind daher nur Auszugsweise dargestellt, um die grundlegenden Prinzipien zu verdeutlichen.

Listing 1: DTD des Moduls enc_lib_range

```
>enc_lib lib=enc_lib_range.so -dtd
<--  DTD generated by IO_SXML. Do not modify! -->
<!ELEMENT doc (config*,data*)>
<!ELEMENT config (lib,attr*)>
<!ELEMENT lib (#PCDATA)>
<!ELEMENT attr (#PCDATA)>
<!ATTLIST attr
        from CDATA ” ” #REQUIRED
        to CDATA “ “ #REQUIRED >
<!ELEMENT data (p*)>
<!ATTLIST data
        id CDATA “ ” #IMPLIED
        src CDATA “ ” #IMPLIED
        src_name CDATA “ ” #IMPLIED
        type CDARA “ ” #IMPLIED
>
<!ELEMENT p (#PCDATA)>
```

Listing 2: SXML - Konfigurationsdefinition des Moduls enc_lib_range

```
>enc_lib lib=enc_lib_range.so -h
<param>
        <name>Library</name>
        <tag>lib</tag>
        <desc>Enter your library name here.</desc>
        <type>TEXT</type>
        <input>USER</input>
        <value>enc_lib_range.so</value>
 </param>
<param>
        <name>Pattern</name>
        <tag>attr</tag>
        <desc>Enter your pattern here.</desc>
        <type>TEXT</type>
```

```
        <input>USER</input>
        <value> </value>
  <attr>
        <name>From</name>
        <tag>from</tag>
        <desc>lower bound</desc>
        <type>FLOAT</type>
        <input>USER</input>
        <value> </value>
  </attr>
  <attr>
        <name>To</name>
        <tag>to</tag>
        <desc>upper bound</desc>
        <count>1</count>
        <type>FLOAT</type>
        <required>true</required>
        <input>USER</input>
        <value> </value>
  </attr>
</param>
```

Listing 3: SXML-Konfigurationsdefinition des Moduls imptab

```
>imptab -h
<param>
  <name>Separator</name>
  <tag>sep</tag>
  <desc>Spaltentrenner</desc>
  <type>TEXT</type>
  <input>USER</input>
  <value> </value>
</param>
...
<param>
  <name>export</name>
  <tag>export</tag>
  <desc>folgende Spalten exportieren:</desc>
  <type>INT|VECTOR</type>
  <input>USER</input>
  <value>0</value>
</param>
...
<param>
  <name>export_header</name>
  <tag>export_header</tag>
  <desc>Tabellen-Kopf exportieren?</desc>
```

```
    <type>BOOL</type>
    <input>SELECT</input>
    <value>true</value>
    <value>false</value>
  </param>
```

B: Module des Prototypen

- imptab: Modul für den Import von Textbasierten Tabellen
- io_process: Ausführung und Steuerung von Workflows
- enc_lib: Kapselung von Kodierungs-Funktionen in Shared Libraries
- enc_lib_direct.so: Shared Library zur Direkten Kodierung von Attributen
- enc_lib_range.so: Shared Library zur Kodierung von Wertebereichen
- enc_direct: Direktes Kodierung von Attributen
- enc_range: Kodierung von Wertebereichen
- io_sxml2net: Konvertierung von XML-Basierten Neuronalen Netzen in das SNNS-Format
- io_net2sxml: Konvertierung von SNNS-Dateien nach XML
- io_sxml2pat: Konvertierung von XML-Basierten Musterdateien in das SNNS-Format
- io_pat2sxml: Konvertierung von SNNS-Musterdateien nach XML
- netgen: Erzeugt Neuronale Netze in Form von SXML-Dateien
- io_snnsbat: Schnittstellen-Modul zur Einbindung des SNNS
- sxmlconf: GUI zur Konfiguration von Modulen

Autorenangaben

Dipl. Wirt. Inf. (FH) Stefan Wissuwa
Fachbereich Wirtschaft
Hochschule Wismar
Philipp-Müller-Straße
Postfach 12 10
D – 23966 Wismar
Telefon: ++49 / (0)3841 / 753 519
Fax: ++49 / (0)3841 / 753 131
E-mail: s.wissuwa@wi.hs-wismar.de

WDP - Wismarer Diskussionspapiere / Wismar Discussion Papers

Heft 01/2003	Jost W. Kramer: Fortschrittsfähigkeit gefragt: Haben die Kreditgenossenschaften als Genossenschaften eine Zukunft?
Heft 02/2003	Julia Neumann-Szyszka: Einsatzmöglichkeiten der Balanced Scorecard in mittelständischen (Fertigungs-)Unternehmen
Heft 03/2003	Melanie Pippig: Möglichkeiten und Grenzen der Messung von Kundenzufriedenheit in einem Krankenhaus
Heft 04/2003	Jost W. Kramer: Entwicklung und Perspektiven der produktivgenossenschaftlichen Unternehmensform
Heft 05/2003	Jost W. Kramer: Produktivgenossenschaften als Instrument der Arbeitsmarktpolitik. Anmerkungen zum Berliner Förderungskonzept
Heft 06/2003	Herbert Neunteufel/Gottfried Rössel/Uwe Sassenberg: Das Marketingniveau in der Kunststoffbranche Westmecklenburgs
Heft 07/2003	Uwe Lämmel: Data-Mining mittels künstlicher neuronaler Netze
Heft 08/2003	Harald Mumm: Entwurf und Implementierung einer objektorientierten Programmiersprache für die Paula-Virtuelle-Maschine
Heft 09/2003	Jost W. Kramer: Optimaler Wettbewerb - Überlegungen zur Dimensionierung von Konkurrenz
Heft 10/2003	Jost W. Kramer: The Allocation of Property Rights within Registered Co-operatives in Germany
Heft 11/2003	Dietrich Nöthens/Ulrike Mauritz: IT-Sicherheit an der Hochschule Wismar
Heft 12/2003	Stefan Wissuwa: Data Mining und XML. Modularisierung und Automatisierung von Verarbeitungsschritten