Diplomarbeit

Anwendung eines Data Mining-Verfahrens auf Versicherungsvertragsdaten

Jens Fisseler

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

4. Februar 2003

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Stefan Rüping

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen der Wissensentdeckung in Datenbanken **3**
2.1 Data Mining und KDD 3
2.1.1 Data Mining-Verfahren im Überblick 4
2.1.2 Relationales Data Mining 6
2.2 Data Mining in zeitveränderlichen Daten 6
2.2.1 Formen zeitveränderlicher Daten 6
2.2.2 Lernaufgaben für zeitveränderliche Daten 8
2.2.3 Zeitreihenanalyse 8
2.2.4 Zeitveränderliche Assoziationsregeln 15
2.2.5 Lernen aus sequentiellen Daten 16

3 Aufgabenstellung **19**
3.1 Die Swiss Life-Daten 19
3.2 Bisherige Untersuchungen der Swiss Life-Daten 20
3.3 Aufgabenstellung der Diplomarbeit 21

4 Data Mining in Intervallsequenzen **23**
4.1 Kams Algorithmus zum Lernen zeitlicher Intervallmuster 24
4.2 Villafanes Einschlussgraphen 26
4.3 Höppners Algorithmus zum Lernen zeitlicher Intervallmuster 27
4.4 Der gewählte Ansatz zum Lernen häufiger zeitlicher Intervallmuster . 29
4.4.1 Intervallsequenzen und Intervallmuster 30
4.4.2 Umsetzung des Algorithmus 30
4.4.3 Von Mustern zu Regeln 35
4.4.4 Erweiterung der Hypothesensprache 38

5 Wissensentdeckung in Versicherungsvertragsdaten **43**
5.1 Anwendung des Algorithmus 43
5.2 Erste Versuche – Lernen aus den Versicherungsvertragsdaten 45
5.2.1 Einfluss der Parameter auf die Anzahl der gefundenen Regeln . . 45
5.2.2 Analyse der Versicherungsvertragsdaten 47
5.3 Wissensentdeckung in den Tarifkomponentendaten 48
5.3.1 Einfluss der Parameter 49
5.3.2 Laufzeiteigenschaften der Algorithmen 50
5.3.3 Analyse der Tarifkomponentendaten 52
5.3.4 Erweiterung der Hypothesensprache 57
5.3.5 Alternative Regelbewertungskriterien 58
5.3.6 Veränderung der Regeln mit der Zeit 59
5.3.7 Vorhersage von Rückkauf 64
5.3.8 Fazit der ersten Analysen 65
5.4 Analyse der Versicherungsverträge 65
5.4.1 Typische Änderungsmuster in Versicherungsverträgen 66
5.4.2 Vorhersage von Rückkauf 69

6 Zusammenfassung und Diskussion **71**
6.1 Ergebnisse der Wissensentdeckung 71
6.2 Lernen häufiger zeitlicher Intervallmuster 75

Abbildungsverzeichnis

2.1 Der KDD-Prozess 4
2.2 Zeitphänomene 7
2.3 Schlusskurs des DAX im Zeitraum August–September 2002 8
2.4 Fiktive Ereignissequenz 16
2.5 Fiktive Intervallsequenz 18

3.1 Schema der Swiss Life-Daten 20

4.1 Intervallmuster, die die Hypothesensprache aus [43] nicht unterscheiden kann 26
4.2 Ein Einschlussgraph nach [71] 26
4.3 Zwei Intervallmuster nach [37] 27
4.4 Supportberechnung nach [37] 29
4.5 Ablauf des Algorithmus 31
4.6 Illustration der Kandidatenmustererzeugung 32
4.7 Algorithmus zur Kandidatenmustererzeugung (nach [40]) 33
4.8 Algorithmus zur Berechnung der Häufigkeit der Muster in $\mathcal{C}_k$ 34
4.9 Berechnung der Teilmusterrelation (nach [40]) 35
4.10 Der Algorithmus zur Regelerzeugung (nach [5]) 37

5.1 Anzahl der in den Versicherungsvertragsdaten gefundenen Regeln, in Abhängigkeit der Parametereinstellungen 46
5.2 Schema der gefundenen Regeln aus den ersten Versuchen 47
5.3 Die zeitliche Abfolge der Einträge dreier Beispielversicherungen 48
5.4 Anzahl der in den Tarifkomponentendaten gefundenen Regeln, in Abhängigkeit der Parametereinstellungen 49
5.5 Anzahl der von der nichtoptimierten (links) und der optimierten Variante (rechts) des Data Mining-Algorithmus erzeugten Kandidatenmuster in Abhängigkeit der Minimalhäufigkeit 50
5.6 Laufzeit (in Sekunden) der nichtoptimierten (links) und der optimierten Variante (rechts) des Data Mining-Algorithmus in Abhängigkeit der Minimalhäufigkeit 51
5.7 Schema der in den FGV-Tarifkomponenten gefundenen Regeln 56
5.8 Skizze eines Regelhaufens 56
5.9 Ähnliche Regeln, die sich nur in einer Intervallrelation unterscheiden . . 57
5.10 Intervallrelation der erweiterten Hypothesensprache 58
5.11 Anzahl an Änderungen pro Jahr, absolut und durchschnittlich pro laufender Tarifkomponente 61

Tabellenverzeichnis

4.1 Intervallrelationen $\mathcal{R}$ nach [7] 24
4.2 Erweiterte Intervallrelationen $\mathcal{R}_{\text{ext}}$ nach [25] 40

5.1 Durchschnittliche Anzahl an Änderungen der Versicherungen und Tarifkomponenten 44
5.2 Fünf Beispielregeln pro Versicherungsart aus der Analyse der Tarifkomponentenänderungen 52
5.3 Durchschnittliche Anzahl an Änderungen der Versicherungsverträge. . . 65
5.4 Fünf Beispielregeln pro Versicherungsart aus der kombinierten Analyse der Versicherungsvertrags- und Tarifkomponentenänderungen. 66

1 – Einleitung

Welche Produkte eines Supermarktes werden von Kunden häufig zusammen gekauft? Wie wird sich der Aktienkurs von Unternehmen B in den nächsten Minuten entwickeln, wenn der Kurs von Konkurrenzunternehmen A gestiegen ist? Werden Eier und Süßigkeiten im Rest des Jahres genauso häufig zusammen gekauft wie im Frühjahr? Wenn ein Kunde eines Online-Buchhändlers die Bücher „Harry Potter und der Stein der Weisen“ und „Harry Potter und die Kammer des Schreckens“ gekauft hat, welches Buch wird er wahrscheinlich als nächstes kaufen?

Einige dieser Fragen kann jeder durch ein bisschen Nachdenken beantworten, wohingegen die Antworten auf andere Fragen nicht so leicht zu finden sind. Data Mining-Verfahren und Methoden der Wissensentdeckung in Datenbanken, seit ca. 10 Jahren ein zentrales Forschungsthema in der Künstlichen Intelligenz, können helfen, nichttriviale Antworten auf die oben gestellten und ähnliche Fragen zu finden.

Gerade Online-Shops, wie z. B. Buchhändler, müssen ihre Kunden durch besondere Aktionen und interessante Angebote umwerben, da die Konkurrenz nur einen Mausklick entfernt ist und der Kunde leicht die verschiedenen Angebote vergleichen kann. Aber auch die Liberalisierung zum Beispiel des Telefon- und Strommarktes hat dazu geführt, dass Unternehmen, die bis vor einigen Jahren noch ein Monopol in ihrem jeweiligen Markt hatten, dazu gezwungen sind, ihre Kunden zu umwerben und besonders profitable Kunden durch gezielte Werbemaßnahmen oder Sonderrabatte zu halten. Ebenso hat die Einrichtung entsprechender Online-Angebote durch Banken und Versicherungen in den letzten Jahren dazu geführt, dass die Kundenberater und Versicherungsmakler „ihre“ Kunden vielleicht nicht mehr so gut kennen wie in der Vergangenheit. Methoden der Wissensentdeckung in Datenbanken können helfen, die vorhandenen Kunden besser zu beraten und neue zu werben.

Diese Diplomarbeit beschäftigt sich mit der Wissensentdeckung in den Kunden- und Versicherungsvertragsdaten der Rentenanstalt/Swiss Life, der ältesten und größten Lebensversicherungsgesellschaft in der Schweiz. Die Datenmenge umfasst die komplette Lebensgeschichte von über 200 000 Versicherungsverträgen, d. h. die Geschichte aller Änderungen an einem Versicherungsvertrag, angefangen von der Annahme des Vertrags durch die Rentenanstalt bis zum Ablauf des Vertrags bzw. zu dem Datum, an dem die Daten aus der Datenbank der Swiss Life extrahiert wurden.

Das Ziel der Wissensentdeckung in diesen Versicherungsvertragsdaten ist, typische oder interessante Profile zu entdecken. Das können Kundenprofile sein, d. h. Mengen von Versicherungsverträgen, die Kunden häufig gleichzeitig halten oder abschließen, aber auch typische Muster in den Lebensgeschichten der einzelnen Versicherungen; Details stehen in Kapitel 3.

Im Unterschied zu den Aufgaben, die im Rahmen des Workshops „KDD Sisyphus – Data Preparation, Preprocessing and Reasoning for Real-World Data“ ([48],

s. auch [16, 27]) gestellt oder von der PG 402 [10] bearbeitet wurden, soll der im Rahmen dieser Arbeit zur Wissensentdeckung verwendete Algorithmus die zeitliche Information, die in den Daten vorhanden ist, explizit verwenden. In welcher Form die zeitliche Information der Daten in [16] und [27] verwendet wurde ist unklar, und in der Arbeit der PG 402 wurden die Daten vor dem eigentlichen Schritt der Wissensentdeckung so aufbereitet, dass die zeitliche Information auch von Algorithmen verwendet werden kann, die nicht dafür ausgelegt sind. Manche Arten von Mustern können aber nur gefunden werden, wenn die zeitlichen Informationen bei der Wissensentdeckung explizit verwendet werden. Deshalb besteht ein Teil dieser Arbeit darin, einen entsprechenden Data Mining-Algorithmus auszuwählen. Der zweite Teil der Arbeit beschäftigt sich mit der Anwendung dieses Algorithmus auf den vorhandenen Daten und der Auswertung der Ergebnisse.

In Kapitel 2 erfolgt zunächst eine allgemeine Einführung in das Data Mining und den Prozess der Wissensentdeckung in Datenbanken. Danach werden die unterschiedlichen Arten zeitlicher Information, die Daten enthalten können, vorgestellt und verschiedene Verfahren erläutert, mit denen Wissensentdeckung in zeitlichen Daten betrieben werden kann. In Kapitel 3 werden die für diese Arbeit benutzten Daten und die Aufgaben und Ziele der Wissensentdeckung in diesen Daten nochmals ausführlich beschrieben. Zudem werden einige Anforderungen an einen anwendbaren Data Mining-Algorithmus formuliert. In Kapitel 4 werden verschiedene Lernverfahren vorgestellt und miteinander verglichen. Zudem wird die Implementation des letztendlich verwendeten Data Mining-Algorithmus detailliert erläutert.

Kapitel 5 beschreibt die verschiedenen Experimente, die im Rahmen des Prozesses der Wissensentdeckung durchgeführt wurden, und schildert die erzielten Ergebnisse. Kapitel 6 fasst die Ergebnisse der Arbeit zusammen und gibt Anregungen und Ideen für weitergehende Forschungen.

2 – Grundlagen der Wissensentdeckung in Datenbanken

Die Einführung des Rechners und die immer größer werdenden Speicherkapazitäten von Festplatten und anderen Speichermedien haben zu einer Anhäufung riesiger Datenbestände in den Datenbanken von Wirtschaft und Wissenschaft geführt. Solche Datenmengen können nicht mehr „von Hand" durchgesehen und analysiert werden, und somit werden Werkzeuge benötigt, die den Anwender bei der Analyse der Daten unterstützen und ihm helfen, nützliche Informationen in der Datenflut zu finden. Die hierzu verwendeten Methoden fallen in das Gebiet der Wissensentdeckung in Datenbanken (Knowledge Discovery in Databases, KDD), und in diesem Kapitel soll zuerst eine kurze Einführung in die „traditionelle" Wissensentdeckung gegeben werden (vgl. [24]), bevor in einem weiteren Abschnitt Methoden und Verfahren zur Wissensentdeckung in zeitveränderlichen Daten vorgestellt werden.

2.1 Data Mining und KDD

Data Mining und KDD haben sich in den letzten Jahren zu einem zentralen Forschungsthema in der Künstlichen Intelligenz entwickelt, wobei aber multidisziplinär Methoden und Verfahren aus dem Bereich des maschinellen Lernens, der Mustererkennung und der Statistik eingesetzt und weiterentwickelt werden. Von Interesse ist das Gebiet aber auch für Forscher aus dem Bereich der Datenbanken, da die riesigen Datenmengen effizient verwaltet werden müssen. Außerdem können die Daten mit Verfahren der Computergraphik visualisiert werden, was oftmals schon eine große Hilfe für die Analyse ist.

Fayyad et al. [24] definieren den Begriff der *Wissensentdeckung in Datenbanken* als

> *[...] den nichttrivialen Prozess der Identifikation gültiger, neuer, möglicherweise nützlicher und letztendlich verständlicher Muster in Daten.*

Die Daten $\mathcal{D}$ bestehen dabei im Allgemeinen aus einer Menge von Objekten, z. B. Tupel einer Tabelle in einer Datenbank, die durch bestimmte Merkmale oder Variablen charakterisiert sind. Im vorherigen Beispiel würde jede Spalte der Tabelle ein Merkmal der Tupel beschreiben.

Die gesuchten Muster P sind Ausdrücke in einer Hypothesensprache $\mathcal{L}_H$, die eine Teilmenge $\mathcal{D}_P \subseteq \mathcal{D}$ abdecken. Die Muster sollten einfacher sein als eine reine Aufzählung aller durch sie abgedeckten Objekte, da sie eine Verallgemeinerung dieser Objekte darstellen und deren gemeinsame Charakteristika widerspiegeln sollen.

Weiterhin sollen die Muster natürlich nicht nur für die Daten $\mathcal{D}$ gelten, in denen sie gefunden wurden, sondern auch für neue Daten. Zudem sollten sie für den Menschen, der die Wissensentdeckung durchführt, neue, bisher unbekannte Erkenntnisse darstellen und für ihn von Nutzen sein. Die verschiedenen Kriterien wie Nützlichkeit, Verständlichkeit und Gültigkeit werden der Einfachheit halber häufig zu einem einzigen Bewertungskriterium zusammengefasst.

Die Anwendung eines Data Mining-Algorithmus auf eine Datenmenge ist nur ein Schritt im Wissensentdeckungsprozess, wenn auch der interessanteste und der bisher am intensivsten erforschte Teil dieses Prozesses. Zur Wissensentdeckung in Datenbanken gehören aber auch eine Reihe weiterer Schritte wie die Aufbereitung der vorliegenden Rohdaten für den Data Mining-Algorithmus und das Auswerten der erzielten Ergebnisse. Abbildung 2.1 zeigt die grundlegenden Schritte im KDD-Prozess (nach [24]).

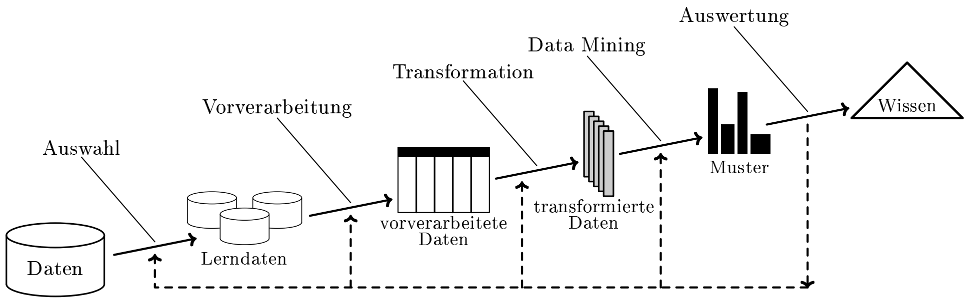


Abbildung 2.1: Der KDD-Prozess

Der Wissensentdeckungsprozess ist interaktiv und iterativ, d. h. er erfordert viele Entscheidungen des Benutzers, der die Wissensentdeckung durchführt, und er muss meistens mehrmals wiederholt werden, um brauchbare Ergebnisse zu erhalten. Der Anwender braucht bereichsspezifisches Wissen und klar definierte Ziele, um im Folgenden die richtigen Entscheidungen zu treffen und die Ergebnisse richtig bewerten zu können. Die vorliegenden Rohdaten müssen ausgewählt und bereinigt werden, um Ausreißer und fehlende Werte zu korrigieren, die das Ergebnis negativ beeinflussen können. Danach wird die vorbehandelte Datenmenge eventuell nochmals reduziert und transformiert, vielleicht weil das ausgewählte Data Mining-Verfahren nicht mit den vorliegenden Daten arbeiten kann oder um einen Teil der Daten zum Testen der gelernten Muster zurückzuhalten. Dann folgt der eigentliche Schritt des Data Minings. Der hierbei verwendete Algorithmus ist vielleicht schon zuvor an Hand der Ziele der Wissensentdeckung ausgewählt worden. Oftmals hat man aber mehrere Verfahren zur Auswahl und wird auch einige ausprobieren, um das für diese konkrete Aufgabe am besten geeignete Verfahren herauszufinden. Nach dem Data Mining erfolgt die Auswertung der Ergebnisse. Der ganze Wissensentdeckungsprozess wird, wenn nötig, mehrmals wiederholt, und am Ende werden die Ergebnisse in einem Bericht zusammengefasst und evtl. in einem Computerprogramm oder auf eine andere Art und Weise weiterverwendet.

2.1.1 Data Mining-Verfahren im Überblick

Im vorangegangenen Abschnitt wurde Data Mining sehr allgemein und im Zusammenhang mit dem Prozess der Wissensentdeckung in Datenbanken vorgestellt. In diesem Abschnitt sollen deshalb die Lernaufgaben, die mit Hilfe von Data Mining-Methoden gelöst werden können, vorgestellt werden.

Definition 1 (Lernaufgabe (nach [31])) *Eine Lernaufgabe wird definiert durch eine Beschreibung der dem lernenden System zur Verfügung stehenden Eingaben (ihrer Art, Verteilung, Eingabezeitpunkte, Darstellung und sonstigen Eigenschaften), der vom lernenden System erwarteten Ausgaben (ihrer Art, Funktion, Ausgabezeitpunkte, Darstellung und sonstigen Eigenschaften) und den Randbedingungen des Lernsystems selbst (z. B. maximale Laufzeiten oder Speicherverbrauch).*

Die Lernaufgaben, die mit Data Mining-Verfahren gelöst werden können, unterteilen sich in prädiktive und deskriptive Lernaufgaben. Bei einer *prädiktiven Lernaufgabe* ist jedem Objekt der Datenmenge der Wert einer unbekannten Funktion zugeordnet und die Aufgabe besteht darin, auf Grundlage der vorliegenden Daten eine möglichst gute Approximation dieser Funktion, d. h. ein globales Modell, zu finden. Das Ziel einer *deskriptiven Lernaufgabe* hingegen ist, für Teilmengen der Daten lokal interessante Muster zu finden.

Die beiden sehr allgemeinen Lernaufgaben der Prädiktion und Deskription lassen sich noch weiter aufgliedern:

Klassifikation ■ Jedes Objekt der Eingabedaten ist einer von mehreren vordefinierten Klassen zugeordnet, und es soll eine Funktion gelernt werden, die diese Zuordnung auf Grundlage der Merkmale der Objekte möglichst gut approximiert.

Clustering ■ Jedes Objekt wird einer oder mehreren Klassen (*Cluster*) zugeordnet, wobei die Klassen nicht vordefiniert sind, sondern ebenfalls aus den Daten bestimmt werden müssen. Die Objekte innerhalb einer Klasse sollen möglichst homogen sein, und die Objekte aus zwei verschiedenen Klassen möglichst heterogen.

Regression ■ Jedem Objekt der Eingabe ist der Wert einer reellwertigen Funktion zugeordnet, und diese Funktion soll mit Hilfe der Merkmale der Objekte approximiert werden.

Modellierung von Abhängigkeiten ■ Hierbei werden lokale Abhängigkeiten zwischen den Merkmalen gesucht.

Assoziationen ■ Zusammenhänge zwischen mehreren Merkmalen werden meist durch sogenannte *Assoziationsregeln* dargestellt.

Diese Liste der Lernaufgaben des Data Minings ist natürlich nicht vollständig, und für jede der Aufgaben gibt es eine Reihe von Verfahren, die sie lösen können. Viele der Verfahren sind nicht neu und stammen aus dem Bereich des maschinellen Lernens, der Statistik oder der Datenbanktheorie. Gerade im Data Mining ist es aber wichtig, dass die verwendeten Algorithmen mit großen Datenmengen und verrauschten oder fehlenden Merkmalen zurecht kommen. Die im Folgenden erwähnten Verfahren sind die bekanntesten und gebräuchlichsten Methoden, um die angegebenen Lernaufgaben zu lösen.

Ein häufig angewandtes Verfahren zur Klassifikation ist C4.5 [61], mit dem sogenannte Entscheidungsbäume und -regeln gelernt werden können. Auch mit Neuronalen Netzen (vgl. Kapitel 7 in [14]) und der Stützvektormethode [70] kann klassifiziert werden, wobei sich beide Methoden auch zur Regression eignen. Beschreibungen von Clusteringverfahren wie k-MEANS finden sich häufig in Lehrbüchern der Statistik. Einen Überblick über verschiedene Clusteringtechniken, die in der Wissensentdeckung Verwendung finden, bietet [12]. Methoden zur Modellierung von Abhängigkeiten sind graphische Modelle wie die Bayesschen Netze [35], und das bekannteste Verfahren zum Lernen von Assoziationsregeln ist APRIORI [5, 3].

2.1.2 Relationales Data Mining

Eine Erweiterung des Data Minings ist das sogenannte relationale Data Mining. Wie oben schon erwähnt, können die analysierten Objekte als Tupel einer Tabelle in einer Datenbank angesehen werden, und die Merkmale als Spalten dieser Tabelle. Die Daten liegen somit in Attribut-Werte-Darstellung oder *propositionaler Form* vor. Data Mining-Algorithmen suchen dann nach Mustern in dieser einen Tabelle. Häufig hat man es aber mit Daten zu tun, die in verschiedenen Tabellen einer Datenbank liegen, die mit Hilfe von Fremdschlüsseln miteinander verknüpft sind und somit in *relationaler Form* vorliegen. Ein Beispiel sind die Kundendaten eines Händlers und die Daten über die Einkäufe der Kunden. Man wird nicht alle Informationen in eine große Tabelle packen und so bei jedem Einkauf die Kundendaten wiederholt speichern. Vielmehr wird eine Tabelle mit den Kundendaten angelegt und eine Tabelle mit den Daten der Einkäufe. Zu jedem Einkauf wird dann nur die Kundennummer mitabgespeichert, über die man die entsprechenden Kundendaten bekommen kann.

Eine Möglichkeit Data Mining in relationalen Daten durchzuführen ist, die Daten so vorzuverarbeiten, dass sie in propositionaler Form vorliegen und so den oben vorgestellten Data Mining-Algorithmen zugänglich sind. Diese Vorverarbeitung muss sorgfältig durchgeführt werden, um keine wichtigen Informationen zu verlieren. Eine Alternative ist, die Data Mining-Verfahren so anzupassen, dass sie aus relationalen Daten lernen können. *Relationales Data Mining* ist somit eine Erweiterung des (propositionalen) Data Minings, und viele Algorithmen zur Klassifikation, zum Clustering oder zum Lernen von Assoziationen sind vom propositionalen auf den relationalen Fall übertragen worden. Einen einführenden Überblick gibt [22].

2.2 Data Mining in zeitveränderlichen Daten

Die in 2.1.1 vorgestellten Data Mining-Algorithmen können nur mit statischen Daten, d. h. Daten ohne zeitliche Merkmale, umgehen. Enthalten die Lerndaten dennoch zeitliche Informationen, so werden diese von den meisten Algorithmen einfach als weitere Merkmale interpretiert, ohne besondere Berücksichtigung ihrer speziellen Semantik. Die meisten Daten jedoch enthalten zeitliche Informationen, und viele Daten lassen sich erst unter Einbeziehung dieser Informationen analysieren. Somit stellt das Data Mining aus zeitveränderlichen Daten[1] eine wichtige Erweiterung des herkömmlichen Data Minings dar, denn erst die explizite Verwendung der Zeit ermöglicht es, zeitliche oder vielleicht sogar kausale Zusammenhänge in Daten zu finden. Im restlichen Teil dieses Kapitels wird deshalb ein Überblick über Methoden und Verfahren zur Wissensentdeckung in zeitveränderlichen Daten gegeben, bevor im nächsten Kapitel die Anforderungen an einen im Rahmen dieser Diplomarbeit anwendbaren Algorithmus formuliert werden.

2.2.1 Formen zeitveränderlicher Daten

Bevor die Verfahren zum Lernen aus zeitveränderlichen Daten vorgestellt werden können wird ein Rahmen benötigt, in den die verschiedenen Methoden eingeordnet werden. Dieser wird im Folgenden festgelegt, sodass die in den nachfolgenden Abschnitten vorgestellten Verfahren vor einem einheitlichen Hintergrund präsentiert werden können.

Roddick und Spiliopoulou definieren in [63] vier Kategorien zeitveränderlicher Daten:

[1] In [63] passender als *temporal data mining* bezeichnet.

Statisch ■ Die zu analysierenden Daten enthalten keinerlei zeitliche Informationen. Auf statischen Daten können Methoden zum zeitlichen Data Mining nicht sinnvoll angewandt werden, wohl aber auf den Ergebnissen einer „klassischen" Wissensentdeckung. Liegen zum Beispiel Ergebnisse zu mehreren Datenmengen vor, die zu verschiedenen Zeiten erfasst wurden, so kann nach Veränderungen in den Ergebnissen gesucht werden [8].

Sequenzen ■ Auf den Merkmalen der Objekte ist eine Ordnungsrelation definiert, d. h. die zu analysierenden Daten sind *Sequenzen* oder *Folgen* von Merkmalen.

Wenn die Zeit auch nicht quantitativ erfasst ist, so erlaubt die Ordnungsrelation doch eine qualitative Darstellung der zeitlichen Relationen der Merkmale untereinander. Die einfachsten dieser Relationen sind *before* und *after*, die einfach die zeitliche Abfolge der Merkmale widerspiegeln, aber auch ausdrucksstärkere Relationen, wie sie von Allen [7] oder Freksa [25] definiert werden, können dargestellt werden.

Zeitstempel ■ Für jedes Merkmal ist der Zeitpunkt vermerkt, an dem sein Wert ermittelt wurde. Die betrachteten Objekte sind somit wieder Sequenzen, aber neben der qualitativen Betrachtung der Zeit können jetzt auch quantitative Aspekte wie die Zeitdauer in die Analyse miteinbezogen werden.

zeitliche Relation ■ Eine *zeitveränderliche Datenbank* (*temporal database*, [69]) speichert zeitveränderliche Daten. Werden in dieser Tupel einer Tabelle geändert oder gelöscht, so werden die alten Daten nicht einfach überschrieben, sondern die komplette Historie bleibt erhalten. Zu diesem Zweck können zu jedem Tupel verschiedene zeitliche Informationen gespeichert sein [41]: Die *Abwicklungszeit* (*transaction time*) ist der Zeitpunkt, an dem der Tupel in die Datenbank eingetragen wurde. Die *Gültigkeitsdauer* (*valid time*) ist die Zeitspanne, während der ein Tupel die durch die Datenbank modellierte Realität widerspiegelt. Die *benutzerdefinierte Zeit* (*user-defined time*) schließlich ist ein beliebiges zeitliches Attribut, dass von der Datenbank wie ein weiteres benutzerdefiniertes Attribut behandelt wird.

Die meisten der im Folgenden vorgestellten Methoden und Verfahren benutzen nur eine zeitliche Dimension, normalerweise die Abwicklungszeit.

Die Sichtweise von Roddick und Spiliopoulou ist stark datenbankorientiert, Morik hingegen strukturiert in [54] zeitliche Phänomene entlang der Zeitachse und entlang des Abstraktionsniveaus, vgl. Abbildung 2.2

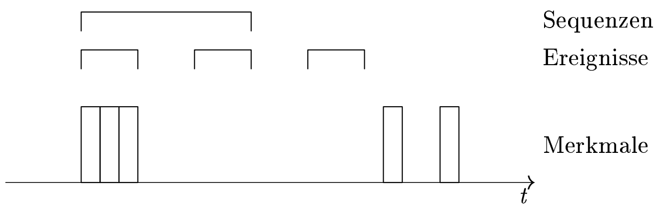


Abbildung 2.2: Zeitphänomene

Auf dem niedrigsten Abstraktionsniveau betrachtet man nur die Anordnung der erfassten Merkmale entlang der Zeitachse. Die meisten statistischen Ansätze zur Analyse zeitveränderlicher Daten arbeiten auf diesem Niveau.

Auf höheren Abstraktionsniveaus werden Folgen von Merkmalen in immer abstrakteren Kategorien zusammengefasst. Ereignisse und Intervalle abstrahieren zum Beispiel von den einfachen Merkmalen auf dem niedrigsten Abstraktionsniveau, und

Sequenzen umfassen wiederum mehrere Ereignisse oder Intervalle. Die zeitliche Information der Elemente in den höheren Kategorien wird aber aus den ihnen zu Grunde liegenden Merkmalen auf dem niedrigsten Abstraktionsniveau gewonnen.

2.2.2 Lernaufgaben für zeitveränderliche Daten

Data Mining-Lernaufgaben lassen sich grob in prädiktive und deskriptive Aufgaben unterteilen, vgl. Abschnitt 2.1.1. In [63] wird eine ähnliche Unterteilung für die Lernaufgaben des zeitlichen Data Minings vorgeschlagen, wobei ein Beispiel aus der Zeitreihenanalyse gegeben wird: Eine mögliche prädiktive Lernaufgabe wäre hier, die zukünftige Entwicklung einer Zeitreihe vorherzusagen. Ähnlichkeiten zwischen vielen Zeitreihen zu finden ist dagegen eine deskriptive Lernaufgabe. Beide Aufgaben überlappen sich jedoch zum Teil, denn eine Möglichkeit zur Vorhersage einer Zeitreihe ist, ähnliche Zeitreihen zu beobachten und daraus Schlüsse über die Entwicklung der vorherzusagenden Zeitreihe zu machen. Morik stellt in [54] ebenfalls eine Reihe von Lernaufgaben vor.

Im Folgenden werden die wichtigsten Formen sequentieller, zeitveränderlicher Daten und die auf ihnen lösbaren Lernaufgaben vorgestellt. Dabei unterscheiden sich nur die Art der erfassten Merkmale und die Repräsentation der Zeit. Zeitreihen zum Beispiel sind zeitliche geordnete Folgen quantitativer Beobachtungswerte. Die Zeit wird hierbei sowohl qualitativ als auch quantitativ erfasst, je nach zu lösender Lernaufgabe (vgl. Abschnitt 2.2.3). Stellen die erfassten Merkmale qualitative Beobachtungswerte dar, so kann man nach zeitveränderlichen Assoziationsregeln oder häufigen Sequenzen suchen, vgl. die Abschnitte 2.2.4 und 2.2.5.

2.2.3 Zeitreihenanalyse

Zeitreihen sind die bekannteste Form zeitveränderlicher Daten. Abbildung 2.3 zeigt als ein Beispiel den Schlusskurs des DAX in den Monaten August 2002 bis Oktober 2002. Sie beschreiben aber auch in vielen anderen Bereichen den zeitlichen Ablauf gewisser Phänomene: Luftdruck und -temperatur sowie Windrichtung und -stärke erlauben zusammen mit weiteren Daten eine kurzfristige Vorhersage des Wetters. In der Medizin wird mit Hilfe regelmäßig erhobener physiologischer Daten die Genesung eines Patienten kontrolliert. In der Mobilfunktechnik schließlich kann die Datenmenge, die bei einem Telefonat übertragen werden muss, mit Hilfe statistischer Methoden erheblich reduziert werden.

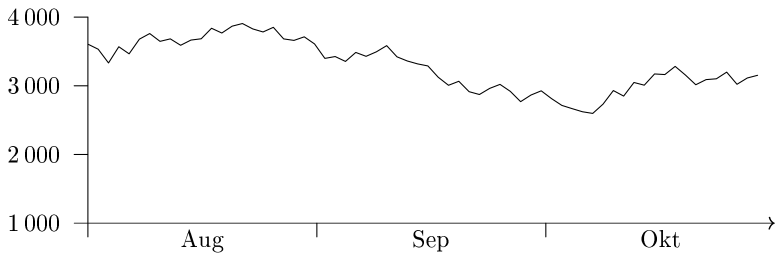


Abbildung 2.3: Schlusskurs des DAX im Zeitraum August–September 2002

Statistische Verfahren

Die Statistik hat eine Fülle von Verfahren zur Analyse von Zeitreihen hervorgebracht. Hier kann nur ein kurzer einführender Überblick über die wichtigsten Methoden zur Beschreibung, Modellierung und Vorhersage gegeben werden. Dieser folgt im Wesentlichen ausgewählten Teilen aus [65, 34, 72].

Eine *Zeitreihe* ist eine (zeitlich) geordnete Folge $(x_t)_{t \in T}$ quantitativer Beobachtungswerte, wobei die *Parametermenge* T meist eine endliche, diskrete Menge von äquidistanten Zeitpunkten ist. Man nummeriert dann die Zeitpunkte durch und setzt $T = \{1, 2, \ldots, N\}$. Die Beobachtungen sind im Folgenden einzelne Werte, man spricht von einer *univariaten* Zeitreihe. Wird diese Einschränkung fallengelassen und werden auch Vektoren als Beobachtungswerte zugelassen, so handelt es sich um eine *multivariate* Zeitreihe. Bei der Analyse multivariater Zeitreihen ist u. a. der Zusammenhang der beobachteten Größen von Interesse. Häufig werden diese Zusammenhänge bei der Untersuchung multivariater Zeitreihen aber ignoriert und die multivariate Zeitreihe wird zur Analyse in ihre univariaten Komponenten zerlegt.

Beschreibung von Zeitreihen. Im Komponentenmodell der Zeitreihenanalyse (s. Abschnitt 1.3 in [65]) wird eine Zeitreihe in vier Teilkomponenten zerlegt:

i) Der *Trend* stellt eine langfristige systematische Veränderung des mittleren Niveaus der Zeitreihe dar.

ii) Längere, nicht notwendigerweise regelmäßige Schwankungen werden in der *Konjunkturkomponente* erfasst.

iii) Die *Saisonkomponente* umfasst in regelmäßigen Zeitabständen relativ unverändert wiederkehrende Phänomene.

iv) In der *Restkomponente* werden nicht zu erklärende Einflüsse oder Störungen erfasst.

Trend und Konjunkturkomponente werden häufig zu einer einzigen, der sogenannten *glatten Komponente* zusammengefasst. Im additiven Modell, das den weiteren Ausführungen zu Grunde liegt, wird unterstellt, dass sich die glatte Komponente g_t, die Saisonkomponente s_t und die Restkomponente r_t additiv überlagern:

$$x_t = g_t + s_t + r_t.$$

Häufig steigt aber zusammen mit der glatten Komponente auch die Streuung der Werte um die glatte Komponente an. Bei solchen Zeitreihen geht man dann von einem multiplikativen Modell aus:

$$x_t = g_t \cdot s_t \cdot r_t.$$

Dieses kann durch logarithmieren auf ein additives Modell zurückgeführt werden:

$$\log x_t = \log (g_t \cdot s_t \cdot r_t) = \log (g_t) + \log (s_t) + \log (r_t) .$$

Ein erster Schritt in der Zeitreihenanalyse besteht häufig darin, den Verlauf der glatten Komponente zu schätzen und diese zu eliminieren. Der Grund ist, dass die meisten Zeitreihenmodelle nur mit stationären Reihen umgehen können, d. h. mit Zeitreihen die – grob gesagt – keine systematischen Veränderungen im Gesamtbild aufweisen.

Zunächst wird davon ausgegangen, dass sich die Zeitreihe nur aus einer Funktion der Zeit m_t und Zufallseinflüssen u_t zusammensetzt: $x_t = m_t + u_t$. Die einfachste Funktion zur Trendapproximation ist dann eine Gerade:

$$m_t = \beta_1 + \beta_2 t.$$

Ihre Parameter β_1, β_2 können mit der Methode der kleinsten Quadrate geschätzt werden, vgl. Abschnitt 1.4.1 in [65]. Eine Erweiterung dieses einfachen Trendmodells ist das *lineare Regressionsmodell*, dass von k bekannten Funktionen $m_1(t), m_2(t), \ldots, m_k(t)$ der Zeit ausgeht und den Trend mit Hilfe einer Linearkombination schätzt:

$$m_t = \beta_1 m_1(t) + \beta_2 m_2(t) + \cdots + \beta_k m_k(t).$$

Die Parameter $\beta_1, \beta_2, \ldots, \beta_k$ dieses Modells können ebenfalls mit der Methode der kleinsten Quadrate geschätzt werden. Ein umfassender Überblick über die beiden hier vorgestellten und weitere Trendmodelle wird in [65], Abschitt 1.4, gegeben.

Lässt man wieder saisonale Schwankungen zu und kommt somit zurück zum vollständigen additiven Zeitreihenmodell, so können die glatte und die saisonale Komponente mit Hilfe von Filtern geschätzt und eliminiert werden, vgl. Abschnitt 1.3 in [34], Kapitel XII und Abschnitt 1.5 in [65].

Ein *linearer Filter* einer Zeitreihe x_t ist eine lineare Abbildung der Form

$$y_t = \sum_{u=-q}^{s} a_u x_{t+u} \quad t = q+1, \ldots, N-s.$$

Die a_u für $u = -q, \ldots, 0, \ldots, s$ sind die *Gewichte* des Filters, und y_t heißt auch *gefilterte Zeitreihe*.

Ein linearer Filter a_u mit $\sum a_u = 1$ heißt *gleitender Durchschnitt* (*moving average*). Im Fall $a_u = \frac{1}{2q+1}$, $u = -q, \ldots, q$, spricht man auch von einem *einfachen gleitenden Durchschnitt*. Mit gleitenden Durchschnitten kann bei Zeitreihen ohne bzw. mit konstanter Saisonkomponente die glatte Komponente approximiert werden.

Eine andere Möglichkeit zur Elimination der glatten und der saisonalen Komponente ist die *Differenzenbildung*. Weist eine Zeitreihe x_t einen linearen Trend auf, so kann dieser durch Bildung der *ersten Differenzen*

$$\Delta x_t = x_t - x_{t-1}, \quad t = 2, \ldots, N,$$

eliminiert werden. Bei einem polynomialen Trend vom Grad d kann die Trendbereinigung durch Bildung der *d-ten Differenzen*

$$\Delta^d x_t = \Delta^{d-1} x_t - \Delta^{d-1} x_{t-1}, \quad t = d+1, \ldots, D,$$

erfolgen.

Liegt schließlich eine stationäre Zeitreihe $(x_t)_{t=1,\ldots,N}$ vor, so bilden die *empirischen Momente* wie das *arithmetische Mittel*

$$\overline{x} = \frac{1}{N} \sum_{t=1}^{N} x_t$$

und die *Varianz*

$$s^2 = \frac{1}{N} \sum_{t=1}^{N} (x_t - \overline{x})^2$$

eine Grundlage für die weitere Analyse. $\overline{x}$ ist der mittlere Wert, um den die Beobachtungen schwanken, und die Varianz s^2 misst die Stärke der Schwankungen. Wichtig ist auch die Stärke des linearen Zusammenhangs aufeinanderfolgender Werte der Zeitreihe. Die *Autokovarianzfunktion* c_τ einer Zeitreihe ist definiert als

$$c_\tau = \frac{1}{N} \sum_t (x_t - \overline{x})(x_{t+\tau} - \overline{x}).$$

In Abhängigkeit vom *Lag* τ, $\tau = -(N-1), \ldots, -1, 0, 1, \ldots, (N-1)$, misst sie die Stärke des linearen Zusammenhangs zwischen Beobachtungen, die τ Schritte auseinanderliegen. Normiert man die Werte der Autokovarianzfunktion mit c_0, so erhält man die *Autokorrelationsfunktion* $r_\tau = \frac{c_\tau}{c_0}$. Der Graph dieser Funktion wird als *Korrelogram* bezeichnet und enthält die wesentlichen Informationen über die zeitlichen Abhängigkeiten in der analysierten Zeitreihe.

Stochastische Prozesse. Nimmt man bei der Analyse von Zeitreihen eine andere Sichtweise an, so kann man eine Zeitreihe $(x_t)_{t\in T}$ als endliche Realisierung einer Folge korrelierter Zufallsvariablen $(X_t)_{t\in T}$ ansehen, ein sogenannter *stochastischer Prozess*. Dieser heißt *schwach stationär*, wenn alle Zufallsvariablen denselben Erwartungswert μ und dieselbe Varianz σ^2 besitzen. Der Grund für diese geänderte Sichtweise ist, dass im Folgenden einige Zeitreihenmodelle vorgestellt werden, die einen schwach stationären Prozess durch sich selbst und/oder einen Prozess ε_t erklären, der ein sogenanntes weißes Rauschen darstellt. Ein stochastischer Prozess ε_t heißt *weißes Rauschen*, wenn er schwach stationär ist und die Zufallsvariablen ε_t alle unkorreliert sind.

Ein stochastischer Prozess X_t heißt *Moving-Average-Prozess der Ordnung* q, kurz MA(q), wenn er sich in der Form

$$X_t = \varepsilon_t + \beta_1\varepsilon_{t-1} + \cdots + \beta_q\varepsilon_{t-q}$$

darstellen lässt. ε_t ist dabei ein weißes Rauschen und $\beta_1,\ldots,\beta_q$ sind die Parameter des Prozesses.

Die Idee hinter den Moving-Average-Prozessen ist, dass der beobachtbare Prozess X_t als gewichtetes Mittel der vorherigen Werte eines unzugänglichen stochastischen Prozesses angesehen wird.

Ein stochastischer Prozess X_t heißt *Autoregressiver Prozess der Ordnung* p, kurz AR(p), wenn er der Gleichung

$$X_t = \alpha_1X_{t-1} + \alpha_2X_{t-2} + \cdots + \alpha_pX_{t-p} + \varepsilon_t$$

genügt. ε_t ist wieder ein weißes Rauschen und $\alpha_1,\ldots,\alpha_p$ sind die Parameter.

Autoregressive Prozesse entsprechen einer Regression über die vorherigen Werte von X_t und einem zufälligen Rest.

MA(q)- und AR(p)-Prozesse kann man zu den ARMA(p,q)-Prozessen zusammenfassen:

$$X_t = \alpha_1X_{t-1} + \alpha_2X_{t-2} + \cdots + \alpha_pX_{t-p} + \varepsilon_t + \beta_1\varepsilon_{t-1} + \cdots + \beta_q\varepsilon_{t-q}.$$

Mit ARMA-Prozessen können viele Zeitreihenphänomene modelliert werden. Um eine Zeitreihe x_t zu modellieren, muss zuerst ein entsprechendes Modell ausgewählt werden, d. h. die Parameter p und q müssen festgelegt werden. In einem weiteren Schritt werden die Parameter $\alpha_1,\ldots,\alpha_p$ und $\beta_1,\ldots,\beta_q$ des Prozesses geschätzt und die Adäquatheit des Modells wird überprüft. Details zur Anpassung von ARMA-Prozessen an gegebene Zeitreihen werden in [34], Kapitel XII, Abschnitt 2.5 und [65], Kapitel 6 beschrieben. Dort wird auch erklärt, wie mit Hilfe eines angepassten ARMA-Prozesses die Entwicklung einer Zeitreihe prognostiziert werden kann.

Weitere Methoden. Die bisher vorgestellten Verfahren zur Zeitreihenanalyse können nur lineare Zusammenhänge in den untersuchten Zeitreihen modellieren. Diese Einschränkung fällt weg, wenn man *Zustandsraummodelle* betrachtet, bei denen von der Annahme ausgegangen wird, dass der eigentlich interessierende Zustand nicht direkt beobachtet werden kann. Die durch die Zeitreihe erfassten Beobachtungen stellen nur eine von Rauschen überlagerte Version des Zustands dar. Viele im Rahmen der Zeitreihenanalyse betrachteten Modelle lassen sich durch diesen Ansatz erfassen, u. a. die ARMA-Modelle, vgl. [65], Abschnitt 2.5. Mit Zustandsraummodellen können aber auch nichtlineare Zusamenhänge erfasst und analysiert werden, s. [14], Kapitel 5 und [72].

Eine weitere Methode zur Analyse von Zeitreihen ist, diese im Frequenzbereich zu untersuchen, vgl. Kapitel 3 und 7 in [65]. Diese Verfahren wurden hier aus Platzmangel nicht vorgestellt.

Datenbankorientierte Verfahren

Neben den statistischen Verfahren zur Analyse und Beschreibung von Zeitreihen wurden in den letzten Jahren viele Methoden entwickelt, die zu einer gegebenen Zeitreihe ähnliche Zeitreihen in einer Datenbank finden können. Die Aufgabe des Vergleichs zweier Zeitreihen hört sich zunächst trivial an, da aber einfache Algorithmen im Allgemeinen polynomielle Laufzeit in der Länge der zu vergleichenden Zeitreihen haben, spielt die zum Vergleich benötigte Rechenzeit bei großen Datenbanken eine wichtige Rolle. Viele der im Folgenden vorgestellten Verfahren verwenden ausgeklügelte Indizierungsmethoden, um die einer gegebenen Zeitreihe ähnlichsten Zeitreihen möglichst schnell zu finden.

Um die verschiedenen Verfahren vor einem gemeinsamen Hintergrund beschreiben zu können, wird zuerst die zu lösende Lernaufgabe beschrieben: Gegeben sind im Folgenden eine Datenbank $\mathcal{D}$ mit Zeitreihen x_t, eine weitere Zeitreihe q_t sowie ein Abstandsmaß d und ein Schwellwert $\varepsilon \geq 0$. Gesucht ist die Menge aller Sequenzen aus $\mathcal{D}$, die zu q_t höchstens den Abstand ε aufweisen:

$$R := \{x_t \in \mathcal{D} \,|\, d(q_t, x_t) \leq \varepsilon\}.$$

Wie eingangs schon erwähnt, verwenden viele Algorithmen zur Suche ähnlicher Zeitreihen Indizierungsmethoden, um nicht alle Zeitreihen in $\mathcal{D}$ vollständig mit q_t vergleichen zu müssen. Damit Indexstrukturen die Suche nach ähnlichen Zeitreihen beschleunigen, müssen sie folgende Eigenschaften erfüllen [23]:

- Sie sollten schneller sein als der naive Vergleich von q_t mit jeder Zeitreihe $x_t \in \mathcal{D}$.
- Sie müssen vollständig sein, d. h. eine Obermenge aller Zeitreihen $x_t \in \mathcal{D}$ mit $d(q_t, x_t) \leq \varepsilon$ zurückgeben. Die Zeitreihen mit einem größeren Abstand als ε können dann in einem Nachbearbeitungsschritt entfernt werden.
- Die Indexstruktur sollte nur wenig Speicherplatz benötigen.
- Es muss möglich sein, dynamisch Zeitreihen hinzuzufügen und zu entfernen.
- Es sollte möglich sein, sowohl Zeitreihen unterschiedlicher Länge in der Indexstruktur zu speichern als auch Suchabfragen mit Zeitreihen unterschiedlicher Länge durchzuführen.

Eine Zeitreihe der Länge n kann als Vektor in einem n-dimensionalen Raum angesehen werden, sodass zur Indizierung räumliche Indexstrukturen wie R-Bäume [33, 11] verwendet werden können. Da die Zeitreihen aber im Allgemeinen aus hunderten oder tausenden von Beobachtungswerten bestehen, und da die Performanz räumlicher Indexstrukturen für Dimensionen größer als 15 stark abnimmt, werden Methoden benötigt, niedrigdimensionale Signaturen $\tilde{x}$ für die Zeitreihen x_t zu berechnen und diese Signaturen zu indizieren. Da die Signaturen der Zeitreihen indiziert werden, muss auch das Abstandsmaß d auf die Signaturen übertragen werden, und um die Vollständigkeit der Berechnung von R zu garantieren, muss das Abstandsmaß der Signaturen den wahren Abstand zweier Zeitreihen *unterschätzen*:

$$d_{\text{sig}}(\tilde{x}, \tilde{y}) \leq d(x_t, y_t).$$

Um zu einer Zeitreihe q_t nun alle Zeitreihen $x_t \in \mathcal{D}$ mit $d(q_t, x_t) \leq \varepsilon$ zu berechnen, werden mit Hilfe der Indexstruktur zuerst alle Zeitreihen $x_t \in \mathcal{D}$ mit $\tilde{d}(\tilde{x}, \tilde{q}) \leq \varepsilon$ gesucht. Hierbei werden auch einige Zeitreihen herausgesucht, deren wahrer Abstand zu q_t größer ist als ε. Deshalb wird in einem Nachbearbeitungsschritt für jede herausgesuchte Zeitreihe der wirkliche Abstand $d(x_t, q_t)$ berechnet.

Agrawal et al. [1] verwenden die Diskrete Fourier Transformation um die Signatur der Zeitreihen zu berechnen, wobei als Abstandsmaß der Euklidische Abstand verwendet wird. Die Signatur einer Zeitreihe besteht aus ihren ersten k Fourierkoeffizienten, die, da sie komplexe Zahlen sind, in einer $2k$-dimensionalen Indexstruktur (F-Index) abgelegt werden. Der Ansatz funktioniert, da sich viele Zeitreihen sehr gut durch ihre ersten Fourierkoeffizienten approximieren lassen, und da der Euklidische Abstand im Frequenzbereich erhalten bleibt. Ein Nachteil ist aber, dass alle Zeitreihen exakt die gleiche Länge haben müssen.

Faloutsos et al. [23] erweitern den Ansatz aus [1] und lassen die Einschränkung fallen, dass alle Zeitreihen gleichlang sein müssen. Sie gehen davon aus, dass die Zeitreihe q_t eine minimale Länge von $w \geq 1$ hat. Zur Indizierung wird ein gleitendes Zeitfenster der Länge w durch alle Zeitreihen $x_t \in \mathcal{D}$ geschoben, und der Inhalt jedes Fensters wird indiziert, d. h. eine Zeitreihe der Länge $n \geq w$ erzeugt $n - w + 1$ Einträge in der Indexstruktur. Die große Menge an Einträgen würde die Verwendung der Indexstruktur aber ineffizient machen. Um die Suche im Index zu beschleunigen, nutzen die Autoren die Tatsache aus, dass aufeinanderfolgende Zeitfenster ähnliche Einträge im Index erzeugen und die Zeitfenster einer Zeitreihe somit einen „Pfad“ im Index hinterlassen. Dieser Pfad wird mit Hilfe einer Heuristik in Hüllrechtecke unterteilt, und diese werden anstelle des Pfades im Index gespeichert. Somit ist es möglich, Zeitreihen einer beliebigen Länge $n \geq w$ zu indizieren und auch Suchanfragen einer beliebigen Länge $n \geq w$ zu beantworten.

Viele neuere Verfahren zur Suche nach ähnlichen Zeitreihen basieren auf den Ansätzen aus [1] und [23]. Chan und Fu [19] zum Beispiel verwenden die Diskrete Wavelet Transformation zur Berechnung der Signatur einer Zeitreihe. Keogh et al. [45] unterteilen die Zeitreihen einfach in N gleichlange Abschnitte, berechnen für jeden Abschnitt den Mittelwert der Beobachtungswerte der Zeitreihe und verwenden diesen Vektor der Länge N als Signatur. Trotz dieser sehr einfachen Vorgehensweise erlaubt diese Indizierungsmethode Suchanfragen variabler Länge und ist genauso schnell oder schneller als die Ansätze aus [1, 23, 19].

Ein Nachteil der bisher vorgestellten Verfahren ist die Verwendung des Euklidischen Abstandsmaßes. Es ist sehr empfindlich gegenüber Verzerrungen entlang der Zeitachse oder Ausreißern in den Beobachtungswerten, und so können Zeitreihen, die bei visuellem Vergleich als sehr ähnlich angesehen werden, einen großen Euklidschen Abstand aufweisen. In der Literatur gibt es verschiedene Ansätze, die Nachteile des Euklidischen Abstands zu umgehen.

Der Ansatz aus [2] sieht zwei Zeitreihen genau dann als ähnlich an, wenn sie genug ähnliche nichtüberlappende Teilzeitreihen enthalten. Um die Ähnlichkeit zweier Zeitreihen x_t und y_t zu berechnen, werden zuerst alle Paare ähnlicher Zeitfenster der Länge w aus x_t und y_t berechnet. Die Beobachtungswerte werden dabei auf den Bereich $[-1, +1]$ normiert, sodass unterschiedliche Skalierungen der Zeitreihen oder Verschiebungen entlang der Werteachse bei der Berechnung der Ähnlichkeit keine Rolle spielen. Aus Paaren ähnlicher Zeitfenster werden dann längere, zusammenhängende Teilzeitreihen gebildet, und in einem letzten Schritt werden für x_t und y_t nichtüberlappende Paare von Teilzeitreihen bestimmt, die letztendlich die Ähnlichkeit zwischen x_t und y_t definieren.

Das et al. [21] verfolgen einen anderen Ansatz und sehen zwei Zeitreihen x_t und y_t genau dann als ähnlich an, wenn sich eine beliebige Teilzeitreihe einer bestimmten Länge von x_t mit Hilfe einer linearen Abbildung f so transformieren lässt, dass der Abstand zu einer beliebigen Teilzeitreihe von y_t höchsten ε beträgt.

Eine anderes, robusteres Abstandsmaß ist das *Dynamic Time Warping* (DTW) [13]. Dieses Verfahren minimiert den Abstand zweier Zeitreihen mittels Dynamischer Programmierung, wobei die beiden Zeitreihen entlang der Zeitachse verschoben, gedehnt und gestaucht werden. Die Rechenzeit des DTW beträgt bei zwei Zeitreihen der Länge n allerdings $O(n^2)$, sodass dieses Verfahren zur Beschleunigung der Suche nach ähnlichen Zeitreihen zunächst ungeeignet ist. KEOGH hat in [47] allerdings gezeigt, wie das DTW durch lokale Approximation der Zeitreihen beschleunigt werden kann, und in [44] stellt er ein Verfahren vor, mit dem DTW zur Indizierung von Zeitreihen genutzt werden kann.

Landmarks sind eine weitere Möglichkeit, Zeitreihen zu indizieren [59]. Landmarks sind charakteristische Beobachtungswerte einer Zeitreihe wie Minima, Maxima oder Wendepunkte. Werden diese charakteristischen Beobachtungswerte geschickt gewählt, so ist diese Indizierungsmethode invariant unter bestimmten Transformationen der Zeitreihen wie Verzerrungen entlang der Zeitachse.

Neben der datengesteuerten Suche nach ähnlichen Zeitreihen gibt es auch Verfahren, bei denen der Benutzer die Form der gesuchten Zeitreihe spezifizieren kann. AGRAWAL et al. stellen in [4] die Beschreibungssprache $\mathcal{SDL}$ vor, mit deren Hilfe der Benutzer die gesuchte Form beschreiben kann.

Weitere Lernaufgaben

Die im vorherigen Abschnitt vorgestellte Aufgabe des Suchens nach ähnlichen Zeitreihen in einer Menge $\mathcal{D}$ von Zeitreihen ist an sich schon eine anspruchsvolle Lernaufgabe. Sie ist aber auch eine Teilaufgabe, die beim Clustering von Zeitreihen gelöst werden muss. Eine andere Möglichkeit Zeitreihen zu gruppieren, stellen OATES et al. in [55, 56] vor. Sie verwenden das Dynamic Time Warping, um Anfangscluster zu finden, die in einem weiteren Schritt mit Hilfe von Hidden Markov Modellen verfeinert werden. Hidden Markov Modelle werden auch von GE und SMYTH [29] verwendet, um ein bestimmtes Muster in einer Zeitreihe zu finden. Ihre Methode ist aber den prädiktiven Lernaufgaben zuzurechnen. POVINELLI versucht in [60] ebenfalls, Abschnitte von Zeitreihen zu klassifizieren. Er verwendet dazu aber eine Einbettung in einen Zustandsraum und klassifiziert die resultierenden Vektoren. Weitere Arbeiten, die sich mit der Klassifikation von Zeitreihen beschäftigen, sind zum Beispiel [50] und [28].

Die vorgestellten Arbeiten zur Klassifikation und zum Clustering von Zeitreihen stellen natürlich nur einen kleinen Ausschnitt dar, da es aus Platzgründen hier nicht möglich ist, alle Arbeiten vorzustellen. Bibliographien wie [62] geben hier einen weiteren Überblick. Allgemein kann man sowohl die Verfahren zur Klassifikation als auch die zum Clustering aufteilen in solche, die *ganze* Zeitreihen klassifizieren bzw. clustern und solche, die nur mit Teilen oder Ausschnitten von Zeitreihen arbeiten.

Neben den Lernaufgaben der Klassifikation und des Clustering können Zeitreihen auch segmentiert werden, um vom niedrigen Abstraktionsniveau der einzelnen Beobachtungen zu einem höheren Abstraktionsniveau zu gelangen, auf dem die Zeitreihe zum Beispiel in steigende, fallende oder konstante Abschnitte unterteilt wird. GURALNIK und SRIVASTAVA [32] unterteilen eine Zeitreihe in Abschnitte, indem sie versuchen, die Zeitreihe lokal durch Funktionen zu approximieren. Jedes Segment wird dabei durch eine Funktion beschrieben, und die Grenzen der Segmente werden auf Grund des Fehlers, den die approximierende Funktion macht, bestimmt. Die Bestimmung der Segmente erfolgt top-down, falls die gesamte Zeitreihe schon vorliegt, oder inkrementell, falls die Beobachtungswerte erst während der Laufzeit des Algorithmus erhoben werden. Die Qualität der Segmentierung des inkrementellen Algorithmus ist im Allgemeinen schlechter als die des top-down Ansatzes, da der Algorithmus seine Entscheidungen auf Grundlage der ihm bisher bekannten Beobachtungswerte treffen muss. Der top-down vorgehende Algorithmus hat dagegen die vollständige Zeitreihe vorliegen und kann somit bessere Entscheidungen treffen. KEOGH [46] versucht diesen Nachteil zu kompensieren, indem er einfach einige Beobachtungswerte puffert. Auf den gepufferten Beobachtungswerten kann sein Segmentierungsverfahren semiglobale Entscheidungen treffen und erreicht damit fast die gleiche Qualität wie ein global arbeitendes Verfahren. Weitere Segmentierungsverfahren werden in [64] und [51] vorgestellt. Hervorzuheben ist noch die Arbeit von Bauer [9]. Er passt mit Hilfe einer Zustandsraumeinbettung ein statistisches Modell an die Zeitreihe an und kann so Ausreißer oder level changes in den Beobachtungswerten der Zeitreihe erkennen, mit denen diese ebenfalls segmentiert werden kann, die aber auch selber interessante Ereignisse in der Zeitreihe darstellen.

2.2.4 Zeitveränderliche Assoziationsregeln

Das Entdecken von *Assoziationsregeln* [5] ist eine der bekanntesten deskriptiven Lernaufgaben. Ihre bekannteste Anwendung ist wahrscheinlich die sogenannte *Warenkorbanalyse*. Ein Warenkorb enthält alle Produkte, die ein Kunde bei einem Einkauf (*Transaktion*), z. B. in einem Supermarkt, erstanden hat. Ein Beispiel für eine Assoziationsregel ist

$$\text{Bier, Pizza} \rightarrow \text{Chips}.$$

Diese Regel besagt, dass falls bei einem Einkauf Bier und Pizza zusammengekauft werden, wahrscheinlich auch Chips gekauft werden. Assoziationsregeln können zum Beispiel dazu genutzt werden, die Anordnung der Waren in einem Supermarkt zu verbessern oder Kunden eines Online-Shops bestimmte Angebote zu machen, die ihrem Kundenprofil entsprechen.

Die Lernaufgabe des Findens von Assoziationsregeln kann folgendermaßen definiert werden (vgl. [31]): Sei $\mathcal{I}$ eine Menge von Objekten (*items*) und $\mathcal{T} = \{t \mid t \subseteq \mathcal{I}\}$ eine Menge von Transaktionen. Sei weiterhin $\mathit{supp}_{\min} \in [0,1]$ eine benutzerdefinierte Minimalhäufigkeit und $\mathit{conf}_{\min} \in [0,1]$ eine benutzerdefinierte Minimalkonfidenz. Dann sind alle Regeln der Form $X \rightarrow Y$ mit $X \subseteq \mathcal{I}$, $Y \subseteq \mathcal{I}$ und $X \cap Y = \emptyset$ zu finden, für die gilt:

$$\mathrm{supp}(X \rightarrow Y) := \frac{|\{t \in \mathcal{T} \mid X \cup Y \subseteq t\}|}{|\mathcal{T}|} \geq \mathit{supp}_{\min}$$

und

$$\mathrm{conf}(X \rightarrow Y) := \frac{|\{t \in \mathcal{T} \mid X \cup Y \subseteq t\}|}{|\{t \in \mathcal{T} \mid X \subseteq t\}|} \geq \mathit{conf}_{\min}.$$

Häufig enthalten die Transaktionsdaten auch eine Angabe des Zeitpunkts, zu dem diese Transaktion durchgeführt wurde. Zieht man die Transaktionszeitpunkte mit in Betracht, dann können die gefundenen Assoziationsregeln zu verschiedenen Zeiten unterschiedliche Häufigkeiten und Konfidenzen aufweisen. So könnten Kaffee und Brötchen morgens weitaus häufiger zusammen gekauft werden als am Rest des Tages, oder Eier und Süssigkeiten im März öfters als in den restlichen Monaten. Deshalb wurde die Aufgabe des Findens von Assoziationsregeln gehingehend erweitert, *zeitveränderliche Assoziationsregeln* zu lernen.

Özden et al. [57] betrachten zyklische zeitveränderliche Assoziationsregeln. Sie gehen von einer vorgegebenen feinsten Zeiteinheit aus und berechnen für jeden Zeitabschnitt t_i die in ihm vorhandenen Assoziationsregeln. In diesen Assoziationsregeln wird dann nach Zyklen gesucht, d. h. es wird überprüft, ob eine Assoziationsregel wiederholt in Zeiteinheiten gefunden wird, die einen konstanten Abstand voneinander haben.

Chen und Petrounias [20] suchen ebenfalls nach zeitveränderlichen Assoziationsregeln, gehen aber davon aus, dass die gesuchten Assoziationen schon bekannt sind. Sie suchen nach den längsten zusammenhängenden Zeitintervallen, in denen eine gegebene Assoziationsregel gilt, und nach allen Periodizitäten, die eine gegebene Assoziationsregel aufweist.

Li et al. [52] verwenden bei der Suche nach zeitveränderlichen Assoziationsregeln nicht nur *eine* Zeiteinheit, sondern eine Hierarchie von Zeiteinheiten, einen *Kalendar*. Dieser besteht aus n Zeiteinheiten D_i, $1 \leq i \leq n$, wobei jedes Element des

Zeitbereichs D_i eindeutig durch ein Element des Zeitbereichs D_{i+1} abgedeckt werden muss. Ein zulässiger Kalender wäre zum Beispiel (*Monat*, *Tag*), nicht zulässig ist aber (*Jahr*, *Monat*, *Woche*), da eine Woche durchaus teilweise durch zwei Monate abgedeckt werden kann. Li et al. suchen nun nach zeitveränderliche Assoziationsregeln, die in Zeitperioden gültig sind, die durch sogenannte *kalendarische Ausdrücke* definiert werden. Kalendarische Ausdrücke sind immer in Bezug zu einem festen Kalender definiert. Zum Beispiel umfasst der kalendarische Ausdruck $\langle *, 1, 10 \rangle$ in Bezug zu dem Kalender (*Woche*, *Tag*, *Stunde*) die Zeitperioden „die zehnte Stunde des ersten Tages jeder Woche“. Kalendarische Ausdrücke können sowohl feste Zeiträume definieren als auch Periodizitäten beschreiben und können die zeitlichen Änderungen von Assoziationsregeln somit präziser und verständlicher beschreiben als dies bei Verwendung nur einer Zeiteinheit möglich wäre.

2.2.5 Lernen aus sequentiellen Daten

Eine weitere Form zeitlicher Daten sind *Ereignissequenzen*, zeitliche Folgen qualitativer Beobachtungswerte eines bestimmten Vorgangs. Abbildung 2.4 zeigt als Beispiel eine fiktive Sequenz von Ereignissen, die teilweise auch zeitgleich auftreten. Ereignissequenzen und Zeitreihen ähneln sich, da beide zeitliche Folgen von Beobachtungswerten darstellen. Allerdings handelt es sich bei den Zeitreihen um quantitative, d. h. nummerische und somit geordnete Beobachtungswerte, wohingegen Ereignisse qualitative Beobachtungen darstellen. Während in Zeitreihen Trends und typische „Formen“ gesucht werden können (vgl. Abschnitt 2.2.3) können Ereignissequenzen nur nach Mustern durchsucht werden, ähnlich wie Zeichenketten. Weiterhin werden die Beobachtungen bei Zeitreihen meist in regelmäßigen Abständen erhoben, während Ereignisse in unregelmäßig auftreten und keine zeitliche Ausdehnung besitzen.

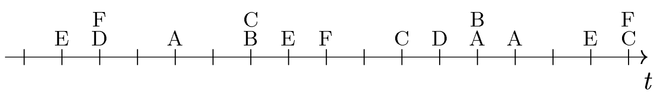


Abbildung 2.4: Fiktive Ereignissequenz

Beispiele für Ereignissequenzen sind Alarmmeldungen in Telekommunikationsnetzen oder Einkäufe, die ein Kunde nacheinander bei einem Online-Händler macht. Im Fall der Alarmmeldungen kann man nach Ereignissen suchen, die größeren Ausfällen des Netzwerks vorausgehen, um diesen vorzubeugen. In den Einkaufstransaktionen könnte man nach typischen Folgen von Einkäufen suchen, die von vielen Kunden gemacht werden, um so anderen Kunden entsprechende Angebote machen zu können und den Umsatz zu steigern.

Agrawal und Srikant [6] haben den Apriori-Ansatz auf Ereignissequenzen übertragen, um in großen Transaktionsdatenbanken nach häufigen sequentiellen Mustern zu suchen. Gegeben ist eine Menge $\mathcal{I}$ von Objekten (*items*) und $\mathcal{T} = \{t \,|\, t \subseteq \mathcal{I}\}$ eine Menge von Transaktionen mit den dazugehörigen Transaktionszeiten, die über eine Identifikationsnummer einem Kunden zugeordnet sind. Die Transaktionen eines Kunden stellen somit eine Sequenz $\langle t_1 \; t_2 \; \ldots \; t_n \rangle$ von Itemmengen $t_j \subseteq \mathcal{I}$ dar, und es werden alle maximalen Sequenzen gesucht, die eine bestimmte Minimalhäufigkeit erreichen. Dabei ist eine häufige Sequenz genau dann maximal, wenn sie nicht Teilsequenz einer anderen häufigen Sequenz ist. Der Grund für die Beschränkung auf die maximalen häufigen Sequenzen liegt darin, dass jede Teilsequenz einer häufigen Sequenz ebenfalls häufig ist, sodass der Anwender mit weitaus mehr häufigen Sequenzen konfrontiert werden würde, wenn nicht nur die maximalen Sequenzen gesucht würden.

In [68] erweitern Srikant und Agrawal ihr Verfahren aus [6]. Sie erlauben zum einen die Definition einer Taxonomie auf der Itemmenge $\mathcal{I}$, sodass zum Beispiel aussagekräftigere oder allgemeinere Ergebnisse gefunden werden können. Zum anderen erlauben Zeitfenster und andere zeitliche Nebenbedingungen eine feinere Parametrisierung des Algorithmus. So können z. B. Transaktionen, die innerhalb eines Zeitfensters einer festen Länge auftreten, zu einer einzigen Transaktion zusammengefasst werden. Damit werden Kunden, die viele kleine Transaktionen kurz hintereinander tätigen genausogut erfasst wie solche Kunden, die eine große Transaktion durchführen. Trotz dieser Erweiterung des Verfahrens hat der neue Algorithmus eine bessere Laufzeit als die Verfahren aus [6].

Im Gegensatz zu Agrawal und Srikant, die häufige Sequenzen in vielen Ereignissequenzen suchen und so die allgemeinen Eigenschaften dieser Ereignissequenzen analysieren, suchen Mannila et al. [53] häufige Sequenzen in einer langen Ereignissequenz. Die gesuchten Muster sind sogenannte *Episoden*, partiell geordnete Mengen von Ereignissen, wobei die Ordnungsrelation die zeitliche Abfolge definiert, in der die einzelnen Ereignisse in der Episode auftreten. Um die Häufigkeit einer Episode zu bestimmen wird ein gleitendes Zeitfenster durch die Eingabesequenz geschoben. Die Häufigkeit einer Episode ist der Anteil der Zeitfenster, die die Episode enthalten, gemessen an der Menge aller Zeitfenster. Aus den häufigen Episoden werden noch Regeln berechnet, die Vorhersagen über die weitere Entwicklung der Eingabesequenz machen.

Einen interessanten Ansatz verfolgen Joshi et al. [42]. Sie definieren eine Hypothesensprache, mit der sich sowohl die häufigen Sequenzen aus [6, 68] als auch die Episoden aus [53] darstellen lassen. Die gesuchten Muster werden dabei wieder als gerichtete azyklische Graphen dargestellt, wobei die Knoten die einzelnen Ereignisse oder Mengen zeitgleich stattfindender Ereignisse darstellen und die Kanten die zeitliche Abfolge der Ereignisse festlegen. Eine gerichtete Kante von Knoten A zu Knoten B drückt dann aus, dass Ereignis A vor Eregnis B stattfindet. Mit den Knoten und den Kanten können noch weitere Nebenbedingungen verbunden sein. So kann eine Kante mit der Nebenbedingung verknüpft sein, dass der zeitliche Abstand zwischen den beiden ajazenten Knoten in einem bestimmten Intervall liegt. Ein Knoten kann ebenfalls mit einer Nebenbedingung verknüpft sein, die den Zeitfenstern aus [68] entspricht und somit erlaubt, Ereignisse, die in einem Zeitfenster der Größe w liegen, zu einer Ereignismenge zusammenzufassen. Neben dieser sehr flexiblen Hypothesensprache stellen die Autoren verschiedene Möglichkeiten der Häufigkeitsberechnung vor. Insgesamt ist der in [42] vorgestellte Algorithmus sehr flexibel anwendbar und subsummiert die Ansätze aus [6, 68] und [53].

Die bisher vorgestellten Ansätze suchen nach *häufigen* Mustern in Ereignissequenzen. In manchen Anwendungen stellen aber gerade seltene Ereignisse die gesuchte Information dar. Ein Beispiel hierfür ist der Bereich der Intrusion Detection, d. h. der möglichst frühen Aufdeckung und Vermeidung von Einbrüchen in Rechnernetze. Einbrüche in Rechnernetze sind im Vergleich zum normalen Netzwerkverkehr relativ selten, stellen aber wichtige Ereignisse dar, da sie Hinweise auf Schwachstellen in der Netzinfrastruktur liefern. Würde man die Protokolle der Firewalls der betroffenen Rechner mit herkömmlichen Methoden zur Analyse von Ereignissequenzen untersuchen, so würde man mit einer großen Menge von uninteressanten häufigen Sequenzen konfontriert werden.

Zaki et al. [73] stellen einen Ansatz vor, mit dem die Ursachen für das Scheitern eines Evakuierungsplans analysiert werden können. Die Methode ist jedoch auf andere Anwendungsbereiche übertragbar. Zuerst werden die Evakuierungspläne in erfolgreiche und gescheiterte Pläne unterteilt und in beiden Datenmengen wird nach häufigen Sequenzen gesucht. Dann werden die in den Daten der gescheiterten Pläne gefundenen häufigen Sequenzen, die auch in den Daten der erfolgreichen Pläne häufig sind, herausgefiltert, da sie keine neuen Erkenntnisse darüber liefern, warum ein Plan gescheitert ist. In einem weiteren Schritt werden redundante Muster entfernt. Eine häufige Sequenz ist genau dann redundant, wenn sie eine kürzere häufige Teilsequenz mit derselben Häufigkeit wie sie selbst enthält. Schließlich werden noch dominierte häufige Sequenzen entfernt, d. h. häufige Sequenzen, die eine kürzere Teilsequenz mit geringerer Häufigkeit in den Daten der erfolgreichen und einer größeren Häufigkeit in den Daten der gescheiterten Pläne enthalten.

Mit ihrem Ansatz können ZAKI et al. aus der Menge aller häufigen Sequenzen diejenigen herausfiltern, die eine Erklärung dafür liefern, warum ein Evakuierungsplan gescheitert ist. Ihre Vorgehensweise lässt sich aber auch auf andere Anwendungen übertragen, in denen eine Vorhersagefunktion für seltene Ereignisse gesucht wird.

Es gibt noch eine ganze Reihe weiterer Verfahren, die nach Mustern in Ereignissequenzen suchen. Für weitere Details sei hier aber wieder auf [62, 63] verwiesen.

Eine weitere Form zeitveränderlicher Daten sind *Intervallsequenzen*. Diese sind den Ereignissequenzen recht ähnlich, da sie auch zeitliche Folgen qualitativer Beobachtungen eines bestimmten Vorgangs darstellen. Allerdings haben die Beobachtungen in diesem Fall eine zeitliche Ausdehnung und stellen somit bestimmte Merkmale oder Charakteristika des beobachteten Vorgangs dar. Abbildung 2.5 zeigt ein fiktives Beispiel.

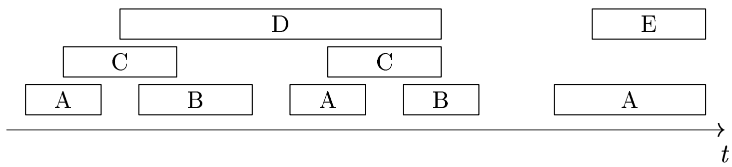


Abbildung 2.5: Fiktive Intervallsequenz

Versicherungsverträge sind ein Beispiel für Intervallsequenzen. Ein Vertrag ist durch verschiedene Merkmale wie Höhe der Prämie oder die im Versicherungsfall begünstigte Person gekennzeichnet. In einer bestimmten Kombination der beschreibenden Merkmale ist ein Vertrag eine gewissen Zeit lang gültig. Ändern sich jedoch einige Merkmale, so beginnt ein neuer Zeitabschnitt, in dem der Vertrag eine andere versicherungstechnische Form hat. Ein anderes Beispiel für eine Intervallsequenz ist der Krankheitsverlauf bei einem chronisch Kranken. Eine Zeit lang geht es dem Kranken recht gut, während es später vielleicht eine Periode mit einem Krankheitsschub gibt. Hier wäre es interessant, in allen Patientenakten chronisch Kranker nach häufigen Intervallsequenzen zu suchen, um so vielleicht zu einem besseren Verständnis der Krankheit zu kommen und die Patienten besser behandeln zu können.

Verschiedene Verfahren, die Muster in Intervallsequenzen suchen können, werden in Kapitel 4 vorgestellt. Die einzelnen Verfahren werden miteinander verglichen und die Vorgehensweise des letztendlich im Rahmen dieser Arbeit eingesetzten Verfahrens wird detailliert erläutert.

3 – Aufgabenstellung

Nach der allgemeinen Einführung in den Bereich der Wissensentdeckung in Datenbanken im vorherigen Kapitel wird im Folgenden die Aufgabenstellung der Diplomarbeit konkretisiert. Dazu werden zuerst die zu analysierenden Daten vorgestellt und erläutert. Anschließend erfolgt ein Überblick über die in vorherigen Analysen erzielten Ergebnisse und es werden mögliche Lernaufgaben formuliert. Im letzten Abschnitt werden einige Anforderungen an ein anwendbares Data Mining-Verfahren aufgestellt.

3.1 Die Swiss Life-Daten

Die vorliegenden Daten stellen einen Auszug aus dem Data Warehouse der Rentenanstalt/Swiss Life dar, der ältesten und größten Lebensversicherungsgesellschaft der Schweiz. Sie umfassen die gesamten Informationen zu über 210 000 Versicherungsverträgen und über 160 000 Kunden, im Folgenden *Partner* genannt (Erklärung s. u.). Die Daten liegen in 18 Tabellen vor und enthalten neben den Informationen zu den Versicherungsverträgen auch demographische Daten zu den Partnern. Diese Informationen sind allerdings anonymisiert worden, sodass nur die Daten zu den Versicherungsverträgen und grundlegende Daten zu den Partnern, wie eine eindeutige Identifikationsnummer oder Geschlecht, verwendet werden können. Abbildung 3.1 zeigt das Datenbankschema der Tabellen mit auswertbaren Daten, das aus den Daten und deren Beschreibung rekonstruiert wurde.

Die Tabellen sind durch die Rechtecke, die den Namen der Tabelle enthalten, repräsentiert, wobei die Namen von Tabellen, die nur aus technischen Gründen benötigt werden, kursiv dargestellt sind. Die Verbindungslinien zwischen den Rechtecken stellen die Beziehungen zwischen den Tabellen dar, wobei zusätzlich noch die Namen der Fremdschlüssel und die Navigationsrichtung angegeben sind.

Tabelle MO_VVERT enthält die komplette Geschichte aller Versicherungsverträge, angefangen von der Annahme durch die Versicherungsgesellschaft bis zum Ablauf des Vertrags bzw. dem Datum, an dem der Data Warehouse-Auszug erstellt wurde (Anfang 2002). Bei den Versicherungsverträgen handelt es sich größtenteils um Kapitallebensversicherungen, weitere vorhandene Versicherungsarten sind Rentenversicherungen, Krankenversicherungen, Erwerbsunfähigkeitsversicherungen und Fondsgebundene Lebensversicherungen.

Ein Versicherungsvertrag besteht aus einem oder mehreren Vertragsteilen, sogenannten *Tarifkomponenten*, die in Tabelle MO_TFKOMP gespeichert sind. Jeder Eintrag in MO_TFKOMP bzw. MO_VVERT entspricht einer Änderung (*Mutation*) einer Tarifkomponente oder eines ganzen Versicherungsvertrags, wobei zu jeder Änderung eine eindeutige Änderungsnummer, der Änderungsgrund und ihr Gültigkeitzeitraum

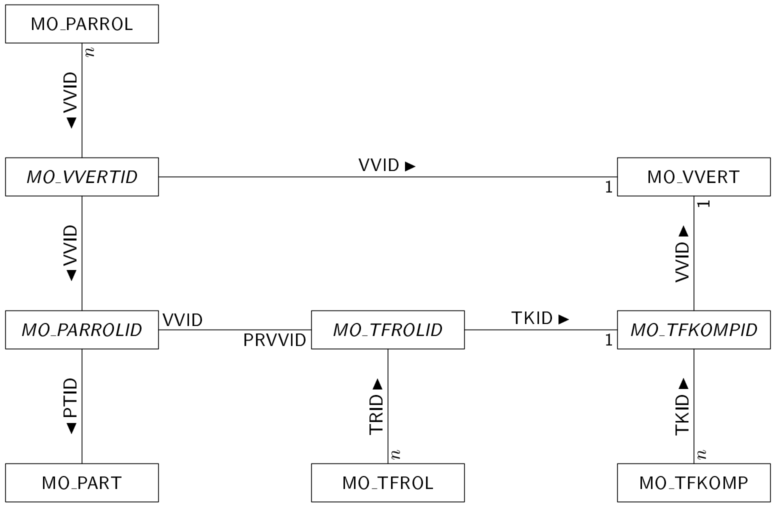


Abbildung 3.1: Schema der Swiss Life-Daten

gespeichert sind.

Die Daten über die Partner der Swiss Life sind in Tabelle MO_PART gespeichert. Hierzu gehören Angaben wie eine eindeutige Identifikationsnummer, Geschlecht, Geburtsjahr und Familienstand des Partners. Jeder Partner kann in einem Versicherungsvertrag verschiedene *Rollen* annehmen, z. B. „Versicherte Person“, „Prämienzahler“ oder „Begünstigter“. Deshalb wird auch von Partner und nicht von Kunde gesprochen, da mit einem Kunden meistens nur der Prämienzahler eines Versicherungsvertrags assoziiert wird, wohingegen verschiedene Personen im Rahmen eines Vertrags unterschiedliche Rollen annehmen können, oder aber eine Person mehrere Rollen. Die wichtigsten Rollen innerhalb eines Vertrags sind (vgl. [26, 30]):

Versicherungsnehmer ▪ Der Partner, der mit dem Versicherungsunternehmen den Versicherungsvertrag abschließt.

Versicherte Person ▪ Derjenige Partner, auf dessen Risiko[1] die Versicherung abgeschlossen wurde.

Prämienzahler ▪ Der Partner, der die Prämien bezahlen muss.

Begünstigter ▪ Der Partner, der bei Fälligkeit die Versicherungsleistung erhält.

Die Zuordnung zwischen den Partnern, ihren verschiedenen Rollen und dem dazugehörigen Versicherungsvertrag geschieht mit Hilfe der Tabelle MO_PARROL. Handelt es sich bei einem Partner um die Versicherte Person, so enthält die Tabelle MO_TFROL weitere Informationen über die der Versicherten Person zugeteilten Tarifkomponenten.

3.2 Bisherige Untersuchungen der Swiss Life-Daten

Ein dem vorliegenden ähnlicher, wenn auch weitaus kleinerer Auszug aus dem Data Warehouse der Swiss Life war 1998 Gegenstand eines Workshops im Rahmen

[1] Die Versicherungslehre versteht unter einem *Risiko* die Möglichkeit des Schadeneintritts durch Verwirklichung einer versicherten Gefahr. Im Falle einer Lebensversicherung wird also das Risiko des menschlichen Lebens durch Tod und ungewisse Lebensdauer erfasst [26].

der ECML'98 [48]. Allerdings waren nicht nur die Versicherungsvertragsdaten, sondern auch die demographischen Daten (die sogenannten *Haushalts-Attribute*) in diesem Auszug auswertbar. Im Rahmen des Workshops wurden drei Data Mining-Aufgaben formuliert, zwei Klassifikationsaufgaben und eine Clusteringaufgabe. Bei den Klassifikationsaufgaben ging es darum, eine Beschreibung derjenigen Partner bzw. Haushalte zu lernen, die zu einer nicht näher erläuterten Klasse gehörten, d. h. es sollte eine unbekannte Funktion approximiert werden. Die Clusteringaufgabe bestand darin, die Haushalte so zu Gruppen zusammenzufassen, dass sich die Haushalte eines Clusters nicht nur bzgl. der Haushalts-Attribute ähneln, sondern auch bzgl. der von den Mitgliedern eines Haushalts abgeschlossenen Versicherungen.

Der ECML'98-Workshop fand zwar nicht statt, jedoch werden in [16, 27] einige Ergebnisse vorgestellt. Die Autoren vergleichen leider nur die Klassifikationsgenauigkeit der mit verschiedenen Data Mining-Verfahren erstellten Modelle und beschreiben nicht die Vorverarbeitungsschritte, die zu diesen Ergebnissen geführt haben. Deshalb sind ihre Ergebnisse für diese Arbeit auch nicht verwertbar.

Im Rahmen der Projektgruppe 402, die im Wintersemester 2001/2002 und im Sommersemester 2002 am Lehrstuhl für Künstliche Intelligenz des Fachbereichs Informatik der Universität Dortmund stattfand, wurden ebenfalls die Swiss Life-Daten analysiert, diesmal jedoch die gleichen Daten, die dieser Diplomarbeit zu Grunde liegen. Es wurde u. a. nach beliebten, d. h. häufig abgeschlossenen Versicherungen gesucht und nach gemeinsam abgeschlossen Vertragsarten. Ein weiterer Schwerpunkt der Datenanalyse bestand in der Vorhersage des Rückkaufs von Versicherungen. Unter *Rückkauf* versteht man die vorzeitige Beendigung einer kapitalbildenden Lebensversicherung wegen Rücktritt oder Kündigung [26]. Dem Versicherungsnehmer ist dabei der aktuelle Zeitwert der Versicherung auszuzahlen. Für die Versicherungsgesellschaft sind zurückgekaufte Versicherungen immer ein Verlustgeschäft, da kurzfristig Kapital beschafft werden muss bzw. entsprechende Finanzmittel nicht längerfristig und damit gewinnbringend angelegt werden können.

Die Teilnehmer der PG 402 haben nun nach gemeinsam veränderten Vertragsmerkmalen in zurückgekauften Versicherungsverträgen gesucht. Weiter wurde versucht, Rückkauf auf Grundlage der Vertragsänderungen vorherzusagen, Details stehen in [10]. Bemerkt werden muss, dass es der Projektgruppe gelungen ist, die zeitliche Information, die in den Vertragsänderungen steckt, so aufzubereiten, dass Klassifikationsverfahren, die zeitliche Informationen nicht explizit verwenden, den Rückkauf eines Versicherungsvertrags recht präzise vorher sagen konnten.

3.3 Aufgabenstellung der Diplomarbeit

Alle Data Mining-Verfahren, mit denen die Swiss Life-Daten bisher analysiert wurden, haben die zeitliche Information der Daten nicht explizit verwendet. Durch entsprechende Vorverarbeitungsschritte wurde dafür gesorgt, dass diese Information dennoch zum Teil erhalten blieb. Das Ziel dieser Diplomarbeit ist es nun, die Versicherungsdaten mit einem zeitlichen Data Mining-Verfahren nach häufigen Sequenzen zu durchsuchen. Auf Grund der relationalen Struktur der Daten erlaubt diese recht allgemein formulierte Aufgabenstellung, die Versicherungsvertragsdaten nach verschiedenen Arten von Sequenzen zu durchsuchen.

Beschränkt man sich beim Data Mining auf die Tarifkomponenten, so kann man nach häufigen Sequenzen von Änderungen suchen. Es ist wahrscheinlich, dass dabei viele Sequenzen gefunden werden, die kein neues, interessantes Wissen darstellen, sondern nur die typischen Änderungsmuster von Tarifkomponenten widerspiegeln, die durch bestimmte versicherungstechnische Prozesse hervorgerufen werden. Durch entsprechende Vorverarbeitung oder Filterung der gefundenen Sequenzen kann man aber versuchen, interessantere Sequenzen zu finden. Interessant wäre es auch, die

Änderungsmuster von Tarifkomponenten, die zu zurückgekauften Versicherungsverträgen gehören, mit denen von nicht zurückgekauften Tarifkomponenten zu vergleichen. Vielleicht zeigen sich hier Unterschiede, mit deren Hilfe der Rückkauf vermindert werden kann.

Die Tarifkomponenten stellen zwar das „Fleisch“ der Versicherungsverträge dar und enthalten damit die interessantesten Informationen. Die gleichen Fragestellungen wie bei den Tarifkomponenten kann man aber auch auf der Ebene der Versicherungsverträge analysieren und so das Zusammenspiel bzw. die gemeinsamen Änderungen der Tarifkomponenten eines Versicherungsvertrags untersuchen.

Nimmt man zur Untersuchung noch die Informationen über die Partner und ihre Rollen hinzu, so kann man analysieren, welche Versicherungen häufig gemeinsam abgeschlossen wurden, oder ob es typische Sequenzen von Versicherungen gibt, die von einem Partner abgeschlossen wurden.

Nach der Einleitung in Abschnitt 2.1 sollte klar sein, dass nicht alle Fragestellungen schon im Voraus formuliert werden können. Manche Fragen stellen sich im Nachhinein vielleicht als schlecht gestellt heraus, während die Ergebnisse anderer Analysen eine ganze Reihe weiterer Fragen aufwerfen können. Die oben beschriebenen Lernaufgaben stellen nur eine Orientierungshilfe dar und werden im Laufe des iterativen KDD-Prozesses vielleicht umformuliert oder erweitert.

Trotzdem lassen sich schon einige Anforderungen an verwendbare Data Mining-Algorithmen stellen. Sie müssen häufige Sequenzen aus Intervallsequenzen lernen können, da die Änderungen der Tarifkomponenten oder Versicherungen solche Sequenzen darstellen: Jede Änderung hat eine bestimmte zeitliche Dauer, und Folgen oder Sequenzen solcher Änderungen stellen die Lebensgeschichte einer Tarifkomponente oder eines Versicherungsvertrags dar. Von Vorteil wäre es auch, wenn aus den gefundenen häufigen Sequenzen Regeln berechnet werden könnten. Dann wäre es möglich, auf Grund der bisherigen Entwicklung einer Tarifkomponente oder eines Versicherungsvertrags Vorhersagen über die mögliche weitere Entwicklung zu treffen. Konkret heißt das z. B. im Falle von Untersuchungen bzgl. des Rückkaufs, dass Partner, die ihre Versicherung demnächst wahrscheinlich zurückkaufen werden, gefunden und angeschrieben werden können, sodass der Rückkauf möglicherweise verhindert wird.

Aufgrund des großen Datenumfangs ist es weiterhin wünschenswert, dass der Algorithmus gut skaliert und ohne großen Aufwand an eine Datenbank angebunden oder vielleicht sogar innerhalb einer Datenbank, d. h. auf dem Datenbankserver, laufen kann.

Die Auswahl eines den obigen Kriterien genügenden Algorithmus und seine Umsetzung werden in Kapitel 4 beschrieben.

4 – Data Mining in Intervallsequenzen

Nachdem in Kapitel 3 einige Anforderungen an einen für die gegebene Aufgabenstellung geeigneten Data Mining-Algorithmus formuliert wurden, sollen in diesem Kapitel verschiedene, in der Literatur beschriebene Verfahren auf ihre Anwendbarkeit hin untersucht werden. Dabei wird nach einem Verfahren gesucht, das folgende Lernaufgabe löst.

Definition 2 (Finden häufiger Intervallmuster) *Sei $\mathcal{M}$ eine Menge von Merkmalen (oder Intervallmarkierungen) und $\mathcal{D}$ eine Menge von Intervallsequenzen, wobei jede Sequenz $S \in \mathcal{D}$ aus einer Folge von markierten Intervallen (b_i, f_i, m_i), $m_i \in \mathcal{M}$, mit Anfangszeitpunkt b_i und Endzeitpunkt f_i besteht. Sei weiterhin $supp_{\min} \in [0, 1]$ eine benutzerdefinierte Minimalhäufigkeit.*

Gesucht sind alle Intervallmuster P in einer Hypothesensprache $\mathcal{L}_H$, für die gilt:

$$\mathrm{supp}(P) := \frac{|\{S \in \mathcal{D} \,|\, S \text{ enthält } P\}|}{|\mathcal{D}|} \geq supp_{\min}.$$

Diese Lernaufgabe ähnelt den Aufgaben des Findens von Assoziationsregeln [5, 3] oder des Findens aller häufigen Ereignissequenzen [68, 53]. Wie dort ist auch hier das Ziel, lokale Zusammenhänge in den Daten zu finden. Jede Intervallsequenz beschreibt die zeitliche Entwicklung eines beobachteten Objekts, wobei jedes Intervall eine Eigenschaft dieses Objekts repräsentiert, die eine bestimmte Zeit lang gilt. Die beobachteten Objekte gehören zu einer gemeinsamen Klasse – im hier betrachteten Anwendungsfall handelt es sich um die Klasse „Versicherungsverträge“ – und die häufigen Intervallmuster beschreiben dann lokale zeitliche Zusammenhänge, die für eine bestimmte Teilmenge aller Objekte gelten.

Allerdings ist die Aufgabe des Findens häufiger Intervallmuster komplexer als z. B. das Finden von Assoziationsregeln. Dort müssen als Teilaufgabe alle häufigen Teilmengen einer Itemmenge $\mathcal{I}$ in einer Menge $\mathcal{T} = \{t \,|\, t \subseteq \mathcal{I}\}$ von Transaktionen gefunden werden, d. h. zu einer gegebenen Minimalhäufigkeit $supp_{\min}$ werden alle Teilmengen $X \subseteq \mathcal{I}$ gesucht, für die gilt:

$$\mathrm{supp}(X) := \frac{|\{t \in \mathcal{T} \,|\, X \subseteq t\}|}{|\mathcal{T}|} \geq supp_{\min}.$$

Eine ähnliche Aufgabe muss auch beim Finden häufiger Intervallmuster gelöst werden, wobei der Itemmenge $\mathcal{I}$ die Menge $\mathcal{M}$ von Merkmalen entspricht[1], und der

[1] Allerdings mit dem Unterschied, dass in einer Intervallsequenz verschiedene Intervalle durchaus die gleiche Markierung tragen können, wohingegen jedes Item nur einmal in einer Transaktion vorkommen kann.

Menge $\mathcal{T}$ der Transaktionen entspricht die Menge $\mathcal{D}$ von Intervallsequenzen. Allerdings spielen bei der Suche nach häufigen Intervallmustern auch die zeitlichen Relationen der Intervalle untereinander eine Rolle, ähnlich wie beim Finden häufiger Ereignissequenzen. Während zwei Ereignisse A und B aber entweder nur zeitgleich oder nacheinander (A *before* B oder A *after* B) stattfinden können, weisen Intervalle weitaus komplexere Relationen auf. Allen hat in seiner zeitlichen Intervalllogik [7] die dreizehn Relationen zusammengefasst, die zwei Intervalle aufweisen können, von denen aber sechs die Inversen anderer Relationen sind, vgl. Tabelle 4.1. Ein Algorithmus, der häufige Intervallmuster berechnet, müsste Allens oder einen ähnlichen Formalismus benutzen, um die zeitlichen Relationen der Intervalle einer Sequenz untereinander zu beschreiben.

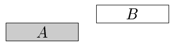
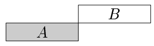
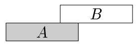
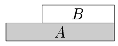
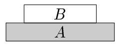
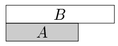
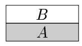
Relation	*inverse Relation*	*graphische Darstellung*
A *before / b* B	B *after / a* A	
A *meets / m* B	B *is-met-by / im* A	
A *overlaps / o* B	B *is-overlapped-by / io* A	
A *is-finished-by / if* B	B *finishes / f* A	
A *contains / c* B	B *during / d* A	
A *starts / s* B	B *is-started-by / is* A	
A *equals / eq* B	B *equals / eq* A	

Tabelle 4.1: Intervallrelationen $\mathcal{R}$ nach [7]

4.1 Kams Algorithmus zum Lernen zeitlicher Intervallmuster

Kam stellt in [43] ein Verfahren zum Lernen häufiger Intervallmuster aus Krankengeschichten vor. Sein Algorithmus ist aber durchaus allgemeiner anwendbar.

Die Eingabe für Kams Algorithmus besteht aus einer Menge $\mathcal{D}$ von Intervallsequenzen, ähnlich wie in Definition 2. Jede Intervallsequenz $S \in \mathcal{D}$ ist über eine ID eindeutig identifizierbar und besteht aus einer Folge von markierten Intervallen, die bezüglich ihrer Endzeitpunkte geordnet sind:

$$S = (b_1, f_1, m_1), (b_2, f_2, m_2), \ldots, (b_n, f_n, m_n), \quad f_i \leq f_{i+1} \text{ für } 1 \leq i \leq n-1.$$

Als Hypothesensprache verwendet Kam rekursiv definierte Intervallmuster. Um die Relationen zwischen den einzelnen Intervallen eines Musters darzustellen, benutzt er sieben von Allens Intervallrelationen:

$$\mathcal{R}_{\text{KAM}} := \{\textit{before}, \textit{meets}, \textit{overlaps}, \textit{is-finished-by}, \textit{contains}, \textit{starts}, \textit{equals}\}.$$

Definition 3 *Eine Intervallmarkierung $M \in \mathcal{M}$ ist ein Intervallmuster, auch* atomares Muster *genannt.*

Sind X und Y Intervallmuster und $r \in \mathcal{R}_{\text{KAM}}$ eine Intervallrelation, dann ist auch $(X\ r\ Y)$ ein Intervallmuster, auch zusammengesetztes Muster *genannt.*

Um überprüfen zu können, ob eine Sequenz S ein Muster X enthält, definiert Kam eine Abbildungsfunktion.

Definition 4 *Ein atomares Muster* X *hat eine Abbildung in eine Sequenz* S *genau dann, wenn diese Sequenz ein Intervall* $I_X = (b, f, X)$ *enthält. Diese Abbildung wird mit* $\mathrm{map}(X, S) := \{I_X\}$ *bezeichnet, und sie gilt für die Zeit* $\mathrm{map}(X, S).b := I_X.b$, $\mathrm{map}(X, S).f := I_X.f$.

Ein zusammengesetztes Muster $(X\ r\ Y)$, $r \in \mathcal{R}_{\mathrm{KAM}}$, *wobei* Y *ein atomares Muster ist, hat eine Abbildung* $\mathrm{map}((X\ r\ Y), S)$ *in eine Sequenz* S *genau dann, wenn* X *eine Abbildung* $\mathrm{map}(X, S)$ *in* S *besitzt und ein mit* Y *markiertes Intervall* I_Y *in* S *existiert, das nicht Teil der Abbildung* $\mathrm{map}(X, S)$ *ist und für das* $Z\ r\ I_Y$ *gilt, wenn* Z *ein imaginäres Intervall mit dem Anfangszeitpunkt* $\mathrm{map}(X, S).b$ *und dem Endzeitpunkt* $\mathrm{map}(X, S).f$ *ist.*

Falls sie existiert, so wird diese Abbildung als $\mathrm{map}((X\ r\ Y), S) := \mathrm{map}(X, S) \cup \{I_Y\}$ *definiert, und sie gilt für den Zeitraum*

$$\begin{aligned}&\mathrm{map}((X\ r\ Y), S).b := \min\{\mathrm{map}(X, S).b, I_Y.b\},\\ &\mathrm{map}((X\ r\ Y), S).f := I_Y.f\end{aligned}$$

Definition 4 beschränkt die von KAMs Algorithmus verwendete Hypothesensprache auf Intervallmuster der Form $((\cdots(A_1\ r_1\ A_2)\ r_2\ A_3)\cdots)\ r_{k-1}\ A_k)$. Ein Grund für die gewählte Hypothesensprache ist, dass KAM vermutet, dass Intervallmuster dieser Form die kausalen Beziehungen, die in den analysierten Daten stecken könnten, gut darstellen können. Weiterere Gründe für diese Einschränkung sind Rechenzeit- und Speicherbedarf, die bei Erweiterung der Hypothesensprache sehr rasch ansteigen würden.

Eine weitere Bedingung die häufige Intervallmuster nach KAM erfüllen müssen ist, dass sie innerhalb einer bestimmten Zeitspanne auftreten müssen. Somit lassen sich häufige Intervallmuster nach KAM wie folgt definieren:

Definition 5 *Sei* $\mathcal{D}$ *eine Menge von Intervallsequenzen,* w *die Größe eines Zeitfensters und* P *ein Intervallmuster.*

Die Häufigkeit *von* P *ist definiert als*

$$\mathrm{supp}(P) := \frac{|\{S \in \mathcal{D} \mid \exists\, \mathrm{map}(P, S)\ \mathit{mit}\ (\mathrm{map}(P, S).f - \mathrm{map}(P, S).b) \leq w\}|}{|\mathcal{D}|}.$$

Ist weiter eine Minimalhäufigkeit $\mathit{supp}_{\min} \in [0, 1]$ *gegeben, dann wird ein Muster* P *genau dann* häufig *genannt, wenn* $\mathrm{supp}(P) \geq \mathit{supp}_{\min}$ *gilt.*

KAMs Algorithmus erfüllt alle Anforderungen, die in der Definition der Lernaufgabe aufgestellt wurden. Allerdings hat die verwendete Hypothesensprache den Nachteil, dass sie einige Intervallmuster nicht unterscheiden kann. Abb. 4.1 zeigt drei Beispielsequenzen, die alle das Muster $(((A\ \mathit{overlaps}\ B)\ \mathit{before}\ C)\ \mathit{overlaps}\ D)$ enthalten, wobei das untersuchte Zeitfenster durch das gestrichelte Rechteck angedeutet ist. Für KAMs Algorithmus enthalten alle drei Beispielsequenzen das gleiche Muster, obwohl die Relation zwischen den Intervallen C und D in allen drei Sequenzen unterschiedlich ist.

Das Problem liegt im Aufbau der Hypothesensprache, der so angelegt ist, dass immer ein weiteres Intervall hinten an ein bestehendes Muster angefügt wird, wobei das vorherige Muster aber nur als ein großes Intervall angesehen wird. Das neue Intervall wird somit nur zu diesem imaginären Intervall in Relation gesetzt, was die Mehrdeutigkeit der Muster verursacht. Je nach Aufgabenstellung kann diese Mehrdeutigkeit durchaus sinnvoll sein, muss aber bei der Interpretation der Lernergebnisse berücksichtigt werden.

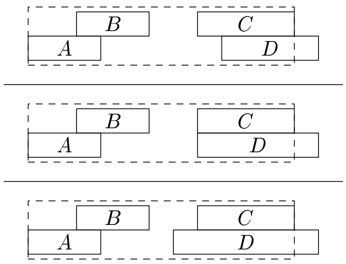


Abbildung 4.1: Intervallmuster, die die Hypothesensprache aus [43] nicht unterscheiden kann

4.2 Villafanes Einschlussgraphen

Villafane et al. stellen in [71] einen Algorithmus zur Suche nach *contains*-Beziehungen in einer Intervallsequenz vor. Die mit diesem Algorithmus lösbare Lernaufgabe entspricht also nicht der in Definition 2, dennoch soll der Algorithmus der Vollständigkeit halber kurz vorgestellt werden.

Die Eingabe des Algorithmus besteht aus einer Folge von markierten Intervallen:

$$(b_1, f_1, m_1), (b_2, f_2, m_2), \ldots, (b_n, f_n, m_n).$$

In dieser Intervallsequenz sucht der Algorithmus nach *contains*-Beziehungen. Diese können instanzenspezifisch sein, d. h. für zwei Intervalle $I_1 = (b_1, f_1, A)$ und $I_2 = (b_2, f_2, B)$ gilt A *contains* B, oder aber allgemein für alle mit A und B markierten Intervalle der Eingabesequenz gelten. Da die *contains*-Relation transitiv ist, entstehen sogenannte *Einschlussgraphen* (*containment graphs*). Abbildung 4.2 zeigt ein Beispiel. Auf der linken Seite ist die Eingabesequenz dargestellt und auf der rechten Seite der entsprechende Einschlussgraph, der Übersichtlichkeit wegen aber ohne die transitiven *contains*-Relationen.

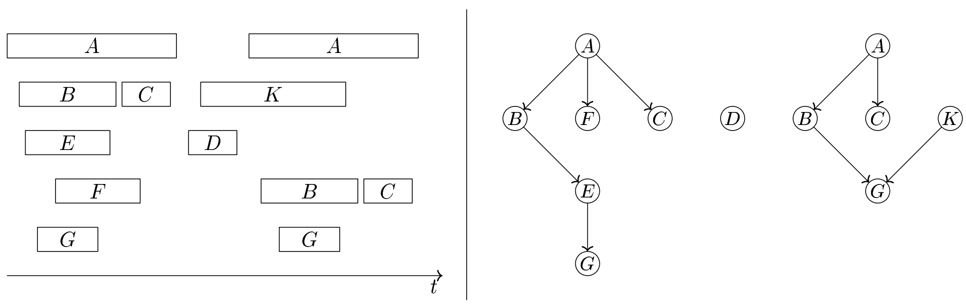


Abbildung 4.2: Ein Einschlussgraph nach [71]

Die Autoren schlagen in [71] verschiedene Prädikate vor, mit denen die Einschlussgraphen bewertet werden können, um die interessanten Graphen einer Intervallsequenz zu finden. Auf Details wird hier aber nicht weiter eingegangen, weil der Algorithmus, wie oben schon erwähnt, nicht dazu geeignet ist, die hier gestellte Lernaufgabe zu lösen.

4.3 Höppners Algorithmus zum Lernen zeitlicher Intervallmuster

Höppner stellt in [37, 40] ein Verfahren zum Lernen häufiger zeitlicher Intervallmuster aus einer Intervallsequenz vor. Die Eingabe für den Algorithmus besteht aus einer lexikographisch sortierten Folge markierter Intervalle:

$$(b_1, f_1, m_1), (b_2, f_2, m_2), \ldots, (b_L, f_L, m_L) \quad \text{mit } b_i \leq b_{i+1}, b_i < f_i \text{ für } 1 \leq i < L.$$

Eine weitere Anforderung an die Eingabesequenz ist die *Maximalitätsbedingung*, die besagt, dass sich keine zwei Intervalle mit derselben Markierung überlappen dürfen:

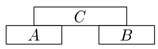
$$\forall (b_i, f_i, m_i), (b_j, f_j, m_j), i < j : f_i \geq b_j \Rightarrow m_i \neq m_j.$$

Wird diese Bedingung durch zwei Intervalle verletzt, dann können diese einfach zu einem Intervall $(\min\{b_i, b_j\}, \max\{f_i, f_j\}, m)$ zusammengefasst werden.

Der Grund für die Maximalitätsbedingung ist, dass Höppner davon ausgeht, dass die Intervallsequenz eine qualitative Beschreibung einer Zeitreihe darstellt. Die Intervalle stellen also Trends wie „steigend“ oder „stark fallend“ oder sonstige relevante Merkmale der Zeitreihe dar. Bei einer solchen qualitativen Beschreibung einer Zeitreihe macht es keinen Sinn, dass sich zwei Intervalle, die dieselbe Eigenschaft beschreiben, überlappen.

Zur Darstellung der Intervallmuster verwendet Höppner Allens zeitliche Intervalllogik, vgl. Tabelle 4.1. Ein aus n Intervallen (b_i, f_i, m_i), $1 \leq i \leq n$, bestehendes Muster wird mit Hilfe einer $n \times n$-Matrix R beschrieben, deren Elemente R_{ij} die Relation zwischen den Intervallen i und j darstellen. Abbildung 4.3 zeigt zwei Beispielmuster, zum Vergleich einmal in einer graphischen Darstellung und ein weiteres Mal in Matrixform.

	A	*B*
A	*eq*	*b*
B	*a*	*eq*



	A	*B*	*C*
A	*eq*	*b*	*o*
B	*a*	*eq*	*io*
C	*io*	*o*	*eq*

Abbildung 4.3: Zwei Intervallmuster nach [37]

Definition 6 *Ein Paar* $P := (m, R)$ *mit* $m : \{1, \ldots, n\} \to \mathcal{M}$ *und* $R \in \mathcal{R}^{n \times n}$, $n \in \mathbb{N}$, *ist ein* zeitliches Intervallmuster der Größe n *genau dann, wenn es eine Intervallsequenz* (b_i, f_i, m_i), $1 \leq i \leq n$, *gibt, sodass* $m(i) = m_i$ *ist und* R_{ij} *die Relation zwischen den Intervallen* $[b_i, f_i]$ *und* $[b_j, f_j]$ *darstellt.*

$\dim(P)$ *bezeichnet die Größe eines Musters* P*, d. h. die Anzahl seiner Intervalle, und bei einem Muster der Größe* n *spricht man auch von einem* n-Muster.

Die Menge aller zeitlichen Intervallmuster mit Markierungen aus $\mathcal{M}$ *wird als* $TP(\mathcal{M})$ *bezeichnet.*

Bei einem zeitlichen Intervallmuster nach Höppner kommt es also nur auf die Intervallmarkierungen und die Intervallrelationen an, ähnlich wie bei den Intervallmustern nach Kam, vgl. Definition 3. Im Gegensatz zu der von Kam verwendeten Hypothesensprache erfasst Höppner in seinen Intervallmustern aber die Relationen *aller* beteiligten Intervalle. Dadurch werden Mehrdeutigkeiten wie bei Kam vermieden. Aber auch HÖPPNERs Hypothesensprache hat einen Nachteil: Durch Umnummerieren der Intervallmarkierungen kann man ein und dasselbe Muster in unterschiedlicher Form darstellen. Allgemein gibt es zu jedem n-Muster $n!$ mögliche Darstellungen. Um jedes n-Muster eindeutig darstellen zu können, wird deshalb eine partielle Ordnung auf $TP(\mathcal{M})$ definiert, mit der dann eine Normalform für die zeitlichen Intervallmuster aufgestellt werden kann.

Definition 7 *Ein Intervallmuster* (m_X, R_X) *ist* Teilmuster *eines Intervallmusters* (m_Y, R_Y), $(m_X, R_X) \sqsubseteq (m_Y, R_Y)$, *genau dann, wenn* $\dim(m_X, R_X) \leq \dim(m_Y, R_Y)$ *gilt und eine injektive Abbildung* $\pi : \{1, \ldots, \dim(m_X, R_X)\} \to \{1, \ldots, \dim(m_Y, R_Y)\}$ *existiert, sodass gilt:*

$$\forall i, j \in \{1, \ldots, \dim(m_X, R_X)\} : m_X(i) = m_Y(\pi(i)) \wedge R_{X,ij} = R_{Y,\pi(i),\pi(j)}.$$

Für Intervallmuster (m_X, R_X) und (m_Y, R_Y), die Permutationen desselben Intervallmusters darstellen, gilt nun $(m_X, R_X) \sqsubseteq (m_Y, R_Y)$ und $(m_Y, R_Y) \sqsubseteq (m_X, R_X)$, ohne dass $m_X = m_Y$ und $R_X = R_Y$ gilt. Deshalb definiert HÖPPNER eine Äquivalenzrelation auf $TP(\mathcal{M})$

$$(m_X, R_X) \equiv (m_Y, R_Y) \;:\Leftrightarrow\; (m_X, R_X) \sqsubseteq (m_Y, R_Y) \wedge (m_Y, R_Y) \sqsubseteq (m_X, R_X)$$

und betrachtet die Faktorisierung $({}^{TP(\mathcal{M})}/_{\equiv}, {}^{\sqsubseteq}/_{\equiv})$ von $TP(\mathcal{M})$ nach $\equiv$. Als *normalisierte Hypothesensprache* $NTP(\mathcal{M})$ wird eine Teilmenge von $TP(\mathcal{M})$ verwendet, die nur noch ein Intervallmuster jeder Äquivalenzklasse enthält.

Definition 8 *Ein zeitliches Intervallmuster* (m, R) *der Größe* n *ist in* Normalform *genau dann, wenn für alle* $1 \leq i, j \leq n$ *gilt:*

$$\begin{aligned} i < j \Leftrightarrow \;& (m(i) = m(j) \wedge R_{ij} = \textit{before}) \\ \vee\; & (m(i) < m(j) \wedge R_{ij} = \textit{equals}) \\ \vee\; & \left(m(i) \neq m(j) \wedge R_{ij} \in \begin{array}{c} \{\textit{before, meets, overlaps,} \\ \textit{is-finished-by, contains, starts}\} \end{array} \right) \end{aligned}$$

Man kann zeigen, dass jedes Intervallmuster eine eindeutige Normalform hat, die man erhält, indem man die Intervalle des Musters lexikographisch sortiert.

Unklar ist noch, wie HÖPPNER die Häufigkeit eines normalisierten Intervallmusters bestimmt. Bei KAM war die Häufigkeit eines Musters als Anteil der Intervallsequenzen, die das Muster als Teilsequenz enthalten, an allen Eingabesequenzen definiert. Da aber HÖPPNERs Algorithmus nur eine lange Intervallsequenz als Eingabe nimmt, muss die Häufigkeit eines Intervallmusters anders definiert werden.

Für HÖPPNER ist ein Intervallmuster nur dann interessant, wenn es in der Eingabesequenz innerhalb eines Zeitfensters einer festen Länge w beobachtet werden kann. Um nun die Häufigkeit eines Intervallmusters zu bestimmen, wird das Zeitfenster durch die Eingabesequenz geschoben und die Gesamtlänge der Zeitfenster berechnet, in denen das Muster beobachtet werden kann. Dabei ist es egal, wie oft das Muster in einem Zeitfenster beobachtet werden kann. Abbildung 4.4 illustriert die Häufigkeitsberechnung für das Muster A *overlaps* B an Hand mehrer Eingabesequenzen, wobei die Zeiträume, in denen das Muster beobachtet werden kann, durch die Doppelpfeile angedeutet sind.

Definition 9 *Sei* t_{akt} *mit* $\min\{b_i \,|\, 1 \leq i \leq L\} \leq t_{\text{akt}} \leq (\max\{f_i \,|\, 1 \leq i \leq L\} + w)$ *die rechte Grenze eines Zeitfensters der Länge* w, *das durch die Eingabesequenz geschoben wird. Dann ist die* Häufigkeit eines Intervallmusters P *definiert als*

$$\text{supp}(P) := \frac{\text{len}\{t \,|\, P \textit{ ist im gleitenden Zeitfenster an Position } t_{\text{akt}} = t \textit{ sichtbar}\}}{T},$$

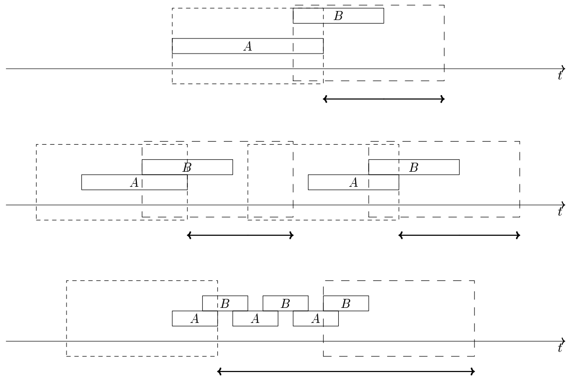


Abbildung 4.4: Supportberechnung nach [37]

wobei len *die Länge der Beobachtungszeiträume aufsummiert und* T *die Gesamtlänge der Eingabesequenz bezeichnet:*

$$T := w + \max\{f_i \,|\, 1 \leq i \leq L\} - \min\{b_i \,|\, 1 \leq i \leq L\}.$$

Ist weiter eine Minimalhäufigkeit $supp_{\min} \in [0,1]$ *gegeben, dann wird ein Muster* P *genau dann* häufig *genannt, wenn* $\mathrm{supp}(P) \geq supp_{\min}$ *gilt.*

Im Gegensatz zu KAMs Hypothesensprache sind HÖPPNERs Intervallmuster besser geeignet, zeitliche Muster in Intervallsequenzen zu finden. Der Nachteil bei HÖPPNERs Ansatz ist aber, dass sein Algorithmus nach häufigen Mustern in *einer* Intervallsequenz sucht, während die Lernaufgabe erfordert, häufige Muster in *vielen* Intervallsequenzen zu finden.

4.4 Der gewählte Ansatz zum Lernen häufiger zeitlicher Intervallmuster

Die Vorstellung der verschiedenen Ansätze in den vorherigen Abschnitten zeigt, dass in der Literatur kein Verfahren existiert, dass die in Definition 2 formulierte Lernaufgabe vollständig löst.

Das Lernverfahren aus [71] hat eine zu stark eingeschränkte Hypothesensprache, die von KAMs Algorithmus ist ebenfalls ungeeignet, und HÖPPNERs Verfahren kann nur häufige Intervallmuster aus einer Intervallsequenz lernen und weist mit der Maximalitätsbedingung zusätzlich eine kleinere Einschränkung der Hypothesensprache auf. Andere, logikbasierte Ansätze wie [16] oder [58] sind ebenfalls nicht anwendbar, da sie sehr rechenzeitaufwändig sind und eine Vorabdefinition der möglichen Intervallmuster benötigen. Das ist für manche Anwendungsfälle sicherlich möglich und notwendig, aber für die vorliegende und viele andere Aufgabenstellungen nicht sinnvoll, da hierfür sehr viel bereichsspezifisches Wissen benötigt wird und nur Regeln gefunden werden, die sowieso in den Daten vermutet werden.

Der im Folgenden gewählte Ansatz kombiniert HÖPPNERs Hypothesensprache mit der Häufigkeitsberechnung von KAMs Algorithmus. Das resultierende Verfahren löst die Lernaufgabe des Findens häufiger Intervallmuster.

4.4.1 Intervallsequenzen und Intervallmuster

Die Eingabe $\mathcal{D}$ des Algorithmus besteht aus einer lexikographisch sortierten Folge von Intervallen

$$\mathcal{D} = (id_1, b_1, f_1, m_1), (id_2, b_2, f_2, m_2), \ldots, (id_L, b_L, f_L, m_L)$$

mit $id_i, b_i, f_i \in \mathbb{N}$ und $m_i \in \mathcal{M}$[2].

Konzeptuell entspricht $\mathcal{D}$ einer Menge von lexikographisch sortierten Sequenzen markierter Intervalle $S_{id} = (b_1, f_1, m_1), (b_2, f_2, m_2), \ldots$, wobei sich jede Sequenz aus aufeinanderfolgenden Intervallen mit derselben Identifikationsnummer id zusammensetzt.

Der Ansatz mit den Eingabedaten $\mathcal{D} = \{(id_i, b_i, f_i, m_i) \mid 1 \leq i \leq L\}$ wurde gewählt, da der Algorithmus so implementiert ist, dass er mit verschiedenen Datenquellen wie Dateien oder Datenbanktabellen arbeiten kann. Für den Zugriff auf eine Datenquelle muss nur eine entsprechende Schnittstelle implementiert werden, die alle Intervalle (id_i, b_i, f_i, m_i) lexikographisch sortiert ausgibt. Die Sequenzen $S_{id} = (b_1, f_1, m_1), (b_2, f_2, m_2), \ldots$ werden dann innerhalb des Algorithmus zusammengesetzt. Im Weiteren bezeichnet $\mathcal{D}_S$ die Sequenzen, die so aus $\mathcal{D}$ zusammengesetzt werden.

Sowohl zur Darstellung der Eingabesequenzen als auch zur Repräsentation der Muster werden normalisierte Intervallmuster verwendet. Allerdings ist die Normalform aus Definition 8 ungeeignet, da sie die Maximalitätsbedingung berücksichtigt. Anfangs wurde fälschlicherweise Höppners Normalform zur Darstellung und Erzeugung der Intervallmuster verwendet (vgl. Abschnitt 5.3.1), was aber später durch folgende Normalform korrigiert wurde[3].

Definition 10 *Ein zeitliches Intervallmuster (m, R) der Größe n ist in* Normalform *genau dann, wenn für alle $1 \leq i, j \leq n$ gilt:*

$$\begin{aligned} i < j \Leftrightarrow \quad & (m(i) \leq m(j) \ \wedge \ R_{ij} = \textit{equals}) \\ \vee & \left(m(i) \neq m(j) \ \wedge \ R_{ij} \in \begin{array}{c} \{\textit{before, meets, overlaps} \\ \textit{is-finished-by, contains, starts}\} \end{array} \right) \end{aligned}$$

Definition 11 *Sei $\mathcal{D}_S$ eine Menge von Intervallsequenzen und sei $P = (m, R)$ ein Intervallmuster. Die* Häufigkeit *von P ist definiert als*

$$\operatorname{supp}(P) := \frac{|\{S \in \mathcal{D}_S \mid P \sqsubseteq S\}|}{|\mathcal{D}_S|}.$$

Ist weiter eine Minimalhäufigkeit $supp_{\min} \in [0, 1]$ gegeben, dann wird ein Muster P genau dann häufig *genannt, wenn $\operatorname{supp}(P) \geq supp_{\min}$ gilt.*

Die Häufigkeitsdefinition ähnelt Definition 5, allerdings ist die zeitliche Ausdehnung, die ein Muster P in einer Sequenz höchstens haben darf, nicht beschränkt.

4.4.2 Umsetzung des Algorithmus

Abbildung 4.5 zeigt den Ablauf des Algorithmus. Dieser ähnelt Apriori [5, 3], was aber nicht verwunderlich ist, da, wie anfangs erwähnt, die Aufgabe des Findens häufiger Intervallmuster eine Erweiterung der Aufgabe des Findens häufiger Itemmengen um die zeitliche Dimension ist.

[2] Damit die Intervallsequenz lexikographisch sortiert werden kann, muss auch auf $\mathcal{M}$ eine Ordnung definiert sein. O. B. d. A. wird im Folgenden $\mathcal{M} = \mathbb{N}$ angenommen.

[3] Auf die Darstellung der Eingabesequenzen hatte die falsche Normalform keinen Einfluss, da deren Repräsentation direkt aus ihrer Intervalldarstellung gewonnen wird.

$\mathcal{L}_1 \leftarrow$ {häufige Intervalle}
$k \leftarrow 2$
while $\mathcal{L}_{k-1} \neq \emptyset$ **do**
  $\mathcal{C}_k \leftarrow$ candidate-generation($\mathcal{L}_{k-1}$)
  support-estimation($\mathcal{C}_k$)
  $\mathcal{L}_k \leftarrow \{P \in \mathcal{C}_k \,|\, \mathrm{supp}(P) \geq supp_{\min}\}$
  $k \leftarrow k + 1$
end while
return $\bigcup_k \ \mathcal{L}_k$

Abbildung 4.5: Ablauf des Algorithmus

Zur Berechnung der häufigen Intervallmuster geht der Algorithmus wie folgt vor: Zuerst werden alle häufigen 1-Muster berechnet. Das sind alle Intervallmarkierungen, die die geforderte Minimalhäufigkeit $supp_{\min}$ erreichen. Aus den $\mathcal{L}_1$ werden dann die möglichen häufigen 2-Muster $\mathcal{C}_2$ bestimmt, deren Häufigkeit im nächsten Schritt berechnet wird. Aus den häufigen 2-Mustern $\mathcal{L}_2$ werden die möglichen häufigen 3-Muster $\mathcal{C}_3$ berechnet usw.

Die Hauptschleife des Algorithmus besteht somit aus zwei Prozeduren, wie alle Apriori-ähnlichen Verfahren. In der k-ten Iteration werden aus häufigen $(k-1)$-Mustern $\mathcal{L}_{k-1}$ die möglichen häufigen Intervallmuster (*Kandidatenmuster*) $\mathcal{C}_k$ erzeugt, deren Häufigkeit im nächsten Schritt berechnet wird. Der Algorithmus terminiert, wenn es keine häufigen Muster mehr gibt. In den nächsten Abschnitten werden die beiden Teilprozeduren „candidate-generation“ und „support-estimation“ detailliert erläutert.

Erzeugung möglicher häufiger Intervallmuster

Die Anzahl möglicher Intervallmuster wächst exponentiell mit der Länge k der Muster. Um möglichst wenige Kandidatenmuster zu erzeugen wird ähnlich wie bei Apriori die Tatsache ausgenutzt, dass jedes Teilmuster eines k-Musters häufig sein muss, insbesondere die $(k-1)$-Teilmuster (vgl. [5, 3]). Um die k-Kandidatenmuster zu erzeugen, werden alle $(k-1)$-Muster $P, Q \in \mathcal{L}_{k-1}$ betrachtet, die ein gemeinsames $(k-2)$-Muster als Präfix haben. Aus P und Q wird ein mögliches häufiges k-Muster X erzeugt, das bis auf die Relation der letzten beiden Intervalle schon festgelegt ist, vgl. Abbildung 4.6.

Die ersten $k-2$ Intervalle und ihre Relationen werden durch den gemeinsamen Präfix von P und Q bestimmt, was in Abbildung 4.6 durch die Teilmatrix A illustriert ist. Bezeichnen p und q die beiden Intervalle von P und Q, die nicht im gemeinsamen Präfix liegen, so sind in X auch die Relationen zwischen p und q und den $k-2$ Intervallen des gemeinsamen Präfix schon durch P und Q festgelegt. Einzig die Relation r zwischen p und q kann in Grenzen frei gewählt werden, womit die inverse Relation $\overline{r}$ zwischen q und p auch festgelegt wird.

Die Einschränkungen für r sind zum einen durch die Normalform gegeben, die r auf eine der Relationen *before*, *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* und *equals* beschränkt.

Weitere Einschränkungen in der Wahl von r ergeben sich durch Allens Transitivitätstabelle für Intervallrelationen, vgl. [7]. Sollen z. B. aus den beiden 2-Mustern A *meets* B und A *meets* C 3-Kandidatenmuster erzeugt werden, so muss die Relation zwischen den Intervallen B und C festgelegt werden. Auf Grund der Transitivitätstabelle muss r aus *starts*, *equals* und *is-started-by* gewählt werden, und in der Normalform bleiben nur die Relationen *starts* und *equals* übrig.

$$
\begin{array}{c|c|c}
R_P & 0\ldots k-2 & p \\
\hline
0 & & \\
\vdots & A & B \\
k-2 & & \\
\hline
p & C & \mathrm{eq}
\end{array}
\qquad\qquad
\begin{array}{c|c|c}
R_Q & 0\ldots k-2 & q \\
\hline
0 & & \\
\vdots & A & D \\
k-2 & & \\
\hline
q & E & \mathrm{eq}
\end{array}
$$

Muster P Muster Q

$$
\begin{array}{c|c|c|c}
R_X & 0\ldots k-2 & p & q \\
\hline
0 & & & \\
\vdots & A & B & D \\
k-2 & & & \\
\hline
p & C & \mathrm{eq} & r \\
\hline
q & E & \overline{r} & \mathrm{eq}
\end{array}
$$

neues Muster X

Abbildung 4.6: Illustration der Kandidatenmustererzeugung

Sei trans : $\mathcal{R} \times \mathcal{R} \to \mathcal{P}(\mathcal{R})$ die Transitivitätsrelation aus [7], d. h. für drei Intervalle $I_1\ r_1\ I_2$ und $I_2\ r_2\ I_3$ gilt $I_1\ r_3\ I_3$ mit $r_3 \in \mathrm{trans}(r_1, r_2)$. Die Menge der möglichen Relationen zwischen p und q berechnet sich dann als

$$
\begin{aligned}
\mathcal{R}_{\mathrm{cand}}(P,Q) := \quad & \Big(\textstyle\bigcap_{1 \leq i \leq k-2} \mathrm{trans}\big(R_{P,(k-1),i}, R_{Q,i,(k-1)}\big)\Big) \\
& \cap \{\textit{before, meets, overlaps, is-finished-by, contains, starts, equals}\}
\end{aligned}
$$

In einem letzten Schritt wird für ein k-Kandidatenmuster X überprüft, ob jedes seiner $(k-1)$-Teilmuster häufig ist. Da die beiden erzeugenden Muster P und Q in jedem Fall häufig sind, müssen nur noch $k-2$ weitere $(k-1)$-Teilmuster überprüft werden.

Bevor der Algorithmus zur Kandidatenmustererzeugung vorgestellt wird, müssen noch einige Implementationsdetails erläutert werden. Der Einfachheit halber werden Intervallmuster $P = (s, R)$ als Vektoren dargestellt (vgl. [40]):

$$(m(1), m(2), R_{21}, m(3), R_{31}, R_{32}, \ldots)\,.$$

Aus diesen Vektoren lassen sich die Intervallmuster eindeutig rekonstruieren. Die Mengen $\mathcal{L}_{k-1}$ und $\mathcal{C}_k$ werden als lexikographisch sortierte Listen solcher Vektoren repräsentiert. In einer sortierten Liste $\mathcal{L}_{k-1}$ bilden Muster mit identischem $(k-2)$-Präfix Blöcke (s. [53]), wobei B_P für ein Muster P im Folgenden den Listenindex des ersten Musters im selben Block wie P bezeichnet. Bei der Kandidatenmustererzeugung kann man sich somit auf Muster P, Q aus demselben Block beschränken. Abbildung 4.7 zeigt den vollständigen Algorithmus.

Trotz der Normalform und der Beschneidung der $\mathcal{C}_k$ kann die Menge der Kandidatenmuster immer noch exponentiell mit der Musterlänge k ansteigen, was aber natürlich auch von den Eingabedaten abhängt.

Satz 1 *Aus einer Menge $\mathcal{L}_{k-1}$ häufiger Intervallmuster können im schlimmsten Fall*

$$|\mathcal{L}_{k-1}| \cdot \left(6 \cdot (|\mathcal{M}| - 1) + \frac{|\mathcal{M}| + 1}{2}\right) \in O\left(7^{(k-1)} |\mathcal{M}|^k\right)$$

k-Kandidatenmuster erzeugt werden.

```
function candidate-generation(L_{k-1})
C_k ← ∅, n ← 0
for i ← 1 upto |L_{k-1}| do
  thisblock ← n + 1, j ← B_{L_{k-1}[i]}
  while B_{L_{k-1}[i]} = B_{L_{k-1}[j]} do
    Erzeuge X aus B_{L_{k-1}[i]} und B_{L_{k-1}[j]} (vgl. Abbildung 4.6)
    for all r ∈ R_cand(B_{L_{k-1}[i]}, B_{L_{k-1}[j]}) do
      Erzeuge alle (k − 2)-Teilmuster T_l von X
      if (∀ 1 ≤ l ≤ k − 1 : T_l ∈ L_{k-1}) then
        n ← n + 1
        C_k[n] ← X
        B_{C_k[n]} ← thisblock
      end if
    end for
    j ← j + 1
  end while
end for
return C_k
```

Abbildung 4.7: Algorithmus zur Kandidatenmustererzeugung (nach [40])

Beweis (informell). Jedes Muster in $\mathcal{L}_{k-1}$ wird bei Erzeugung der k-Kandidatenmuster um ein Intervall verlängert, wobei die Markierungen dieses Intervalls aus $\mathcal{M}$ gewählt werden. Nach Definition 10 gibt es für $|\mathcal{M}|-1$ der verlängernden Intervalle sechs mögliche Intervallrelationen zum letzten Intervall des verlängerten Musters, und für durchschnittlich $\frac{|\mathcal{M}|+1}{2}$ der verlängernden Intervalle ist *equals* noch eine mögliche Relation zum letzten Intervall des verlängerten Musters. □

$O\left(7^{(k-1)}|\mathcal{M}|^k\right)$ ist natürlich eine sehr schwache obere Schranke für die Anzahl der möglichen k-Kandidatenmuster, aber sie zeigt deutlich, dass die Anzahl der Muster exponentiell in der Länge der Muster und in der Anzahl der möglichen Intervallrelationen ist.

Berechnung der Häufigkeit von Intervallmustern

Die Häufigkeit eines Intervallmuster P in einer Menge $\mathcal{D}_S$ von Intervallsequenzen lässt sich durch einfaches Abzählen bestimmen. Für jede Eingabesequenz $S \in \mathcal{D}_S$ und jedes Kandidatenmuster $P \in \mathcal{C}_k$ wird überprüft, ob P ein Teilmuster von S ist. Falls ja, so wird der Häufigkeitszähler von P erhöht. In einem letzten Schritt wird schließlich für jedes Muster aus der absoluten Häufigkeit die relative Häufigkeit berechnet. Abbildung 4.8 zeigt den vollständigen Algorithmus.

Die Berechnung der Häufigkeit eines Musters kann noch optimiert werden. Dazu werden zu jedem k-Muster P in $O_{id}(P)$ die Identifikationsnummern der Eingabesequenzen abgespeichert, die P als Teilmuster enthalten. Soll nun die Häufigkeit eines $(k+1)$-Kandidatenmusters P' bestimmt werden, so muss die Teilmusterrelation $P' \sqsubseteq S$ nur für diejenigen Eingabesequenzen S überprüft werden, deren Identifikationsnummer in der Schnittmenge $E_{id}(P') := \bigcap_{1\leq i\leq k+1} O_{id}(P_i)$ der $O_{id}(P_i)$ aller k-Teilmuster P_i von P' liegt, denn nur wenn eine Eingabesequenz alle k-Teilmuster eines $(k+1)$-Musters enthält, kann es auch das $(k+1)$-Muster enthalten.

Die Verwendung der Mengen $O_{id}(P)$ und $E_{id}(P)$ hat zweifachen Nutzen. Zum einen wird die Berechnung der Häufigkeit eines Kandidatenusters P beschleunigt, da die Teilmusterrelation $P \sqsubseteq S$ nur für solche Sequenzen S berechnet werden muss, deren Identifikationsnummer in $E_{id}(P)$ enthalten ist. Zum anderen muss die

procedure support-estimation($\mathcal{C}_k$)
for all $P \in \mathcal{C}_k$ **do**
  $\mathrm{cnt}(P) \leftarrow 0$
end for

for all $S \in \mathcal{D}_S$ **do**
  for all $P \in \mathcal{C}_k$ **do**
    if $(P \sqsubseteq S)$ **then**
      $\mathrm{cnt}(P) \leftarrow \mathrm{cnt}(P) + 1$
    end if
  end for
end for

for all $P \in \mathcal{C}_k$ **do**
  $\mathrm{supp}(P) \leftarrow \frac{\mathrm{cnt}(P)}{|\mathcal{D}_S|}$
end for

Abbildung 4.8: Algorithmus zur Berechnung der Häufigkeit der Muster in $\mathcal{C}_k$

Häufigkeitsberechnung für weniger Kandidatenmuster P durchgeführt werden. Es gilt nämlich

$$\frac{E_{id}(P)}{|\mathcal{D}_S|} \geq \mathrm{supp}(P),$$

d. h. $\frac{E_{id}(P)}{|\mathcal{D}_S|}$ ist eine obere Schranke für die Häufigkeit eines Musters P. Erreicht diese obere Schranke nicht die erforderliche Minimalhäufigkeit, so kann P die Minimalhäufigkeit mit Sicherheit nicht erreichen und kann somit verworfen werden.

Die Verwaltung der $O_{id}(P)$ und $E_{id}(P)$ benötigt natürlich Speicherplatz und Rechenzeit bei der Berechnung $E_{id}(P)$ während der Erzeugung der Kandidatenmuster. In welchen Fällen mit dem größeren Speicherplatzbedarf wirklich ein Gewinn an Rechenzeit erzielt wird, und wann die optimierte Berechnung der Häufigkeit keine großen Nutzen bringt, wird in Abschnitt 5.3.2 untersucht.

Unklar ist immer noch, wie die einzelnen Sequenzen S aus der Eingabesequenz

$$\mathcal{D} = (id_1, b_1, f_1, m_1), (id_2, b_2, f_2, m_2), \ldots, (id_L, b_L, f_L, m_L)$$

zusammengesetzt werden und wie die Teilmusterrelation $\sqsubseteq$ berechnet werden kann.

Da $\mathcal{D}$ lexikographisch sortiert ist, bilden alle Intervalle mit derselben Identifikationsnummer id eine zusammenhängende Teilsequenz $(id, b_1, f_1, m_1), (id, b_2, f_2, m_2), \ldots, (id, b_l, f_l, m_l)$ in $\mathcal{D}$, und wiederum auf Grund der lexikographischen Sortierung gilt:

$$\forall 1 \leq i, j \leq l : i < j \;\Rightarrow\; [b_i, f_i]\; r\; [b_j, f_j] \text{ mit } r \in \begin{array}{c}\{\textit{before, meets, overlaps,}\\ \textit{is-finished-by, contains,}\\ \textit{starts, equals}\}\end{array}$$

Das Intervallmuster einer Sequenz kann also sukzessive aufgebaut werden: Zuerst wird das erste Intervall hinzugefügt, dann folgt das zweite Intervall, wobei jetzt die Intervallrelation zwischen dem ersten und dem zweiten Intervall bestimmt werden muss. Allgemein müssen beim Hinzufügen des i-ten Intervalls alle Intervallrelationen zwischen dem neuen Intervall und den $i-1$ vorherigen Intervallen bestimmt werden. Somit lassen sich auch die Eingabesequenzen als Vektoren

$$(m(1), m(2), R_{21}, m(3), R_{31}, R_{32}, \ldots)$$

darstellen.

Die Berechnung der Teilmusterrelation $\sqsubseteq$ ist etwas aufwändiger. Sie ähnelt einem naiven Algorithmus zur Suche in Zeichenketten. Dabei wird rekursiv eine Abbildung der Intervalle von P auf Intervalle von Q konstruiert, die die Intervallrelationen erhält, vgl. Definition 7. Abbildung 4.9 zeigt den entsprechenden Algorithmus.

function $P \sqsubseteq Q$
if $\dim(P) > \dim(Q)$ **then**
  return *false*
else
  $i \leftarrow 1$, $j \leftarrow 1$, $\pi(\cdot) \leftarrow 0$
  return $\mathrm{perm}(i, j, P, Q, \pi)$
end if

function $\mathrm{perm}(i, j, P, Q, \pi)$
found $\leftarrow$ *false*
repeat
  if $(m_P(i) = m_Q(j))$ **then**
    $\pi(i) \leftarrow j$
    ok $\leftarrow$ *true*
    for $k \leftarrow 1$ **upto** i **do**
      $ok \leftarrow ok \wedge \left(R_{P,i,k} = R_{Q,j,\pi(k)}\right)$
    end for
    if (ok) **then**
      if $(i < \dim(P))$ **then**
        found $\leftarrow \mathrm{perm}(i+1, j+1, P, Q, \pi)$
      else
        found $\leftarrow$ *true*
      end if
    end if
  end if
  $j \leftarrow j + 1$
until $((j > \dim(Q)) \vee found)$
return *found*

Abbildung 4.9: Berechnung der Teilmusterrelation (nach [40])

4.4.3 Von Mustern zu Regeln

Die häufigen Intervallmuster $\mathcal{L}_k$ liefern zwar Informationen über die Regelmäßigkeiten in den Eingabedaten, ermöglichen aber keine Voraussage über die wahrscheinliche weitere Entwicklung einer Intervallsequenz. Wenn die Intervallsequenzen zum Beispiel die Änderungen von Versicherungsverträgen enthalten, dann wäre es interessant zu wissen, wie sich der Versicherungsvertrag wahrscheinlich weiterentwickelt.

Um aus häufigen Intervallmustern Regeln zu berechnen, übernimmt Höppner in [37, 39] Ideen aus der Berechnung von Assoziationsregeln. Dort werden aus häufigen Itemmengen $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$ mit $X \cap Y = \emptyset$ Assoziationsregeln $X \rightarrow Y$ berechnet. Eine Regel $X \rightarrow Y$ wird genau dann als interessant angesehen, wenn sie eine bestimmte *Minimalkonfidenz* $conf_{\min} \in [0, 1]$ aufweist. Die Konfidenz einer Regel ist dabei definiert als

$$\mathrm{conf}(X \rightarrow Y) := \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)},$$

d. h. sie entspricht der bedingten Wahrscheinlichkeit $\Pr(Y|X)$ [5, 3].

HÖPPNER berechnet Regeln $P \mapsto Q$ aus jedem Paar (P, Q) häufiger Intervallmuster mit $P \sqsubseteq Q$, wobei sich die Konfidenz dieser Regel als

$$\mathrm{conf}(P \mapsto Q) := \frac{\mathrm{supp}(Q)}{\mathrm{supp}(P)}$$

berechnet. Dabei beschränkt er sich auf vorwärtsgerichtete Regeln, d. h. Regeln, die Vorhersagen über die Zukunft machen. Für solche vorwärtsgerichteten Regeln ist P somit ein Präfix von Q.

Die Regelsemantik $P \mapsto Q$ mit $P \sqsubseteq Q$ erscheint auf den ersten Blick ungewöhnlich, da die Prämisse P auch Teilmuster der Konklusion Q ist, wohingegen für die statischen Assoziationsregeln $X \to Y$ $X \cap Y = \emptyset$ gilt. Für zeitliche Intervallregeln hingegen müssen aber noch die Intervallrelationen zwischen den Intervallen der Prämisse und der Konklusion spezifiziert werden, weshalb das Konklusionsmuster Q das Prämissenmuster P als Präfix enthalten muss, s. auch [38].

Wären Prämisse P und Konklusion Q einer Regel $P \mapsto Q$ disjunkt, so könnte es passieren, dass eine Intervallsequenz die Regel $P \mapsto Q$ enthält, Q in dieser Sequenz aber *vor* P auftritt. Die zeitliche Reihenfolge von Prämisse und Konklusion wäre vertauscht. Diese Vertauschung könnte verhindert werden, wenn explizit gefordert würde, dass bei der Berechnung der Konfidenz einer Regel die Prämisse *vor* der Konklusion auftritt. Dies würde aber einen weiteren Durchlauf durch die Eingabedaten notwendig machen, da die Definition der Konfidenz geändert werden müsste:

$$\mathrm{conf}(P \mapsto Q) := \frac{|\{S \in \mathcal{D}_S \mid (P \sqsubseteq S) \wedge (Q \sqsubseteq S) \wedge (P \textit{ before } Q)\}|}{\mathrm{supp}(P) \cdot |\mathcal{D}_S|}.$$

Ein weiterer Nachteil dieser Regelsemantik ist, dass der zeitliche Abstand zwischen Konklusion und Prämisse beliebig groß sein kann. Dies könnte verhindert werden indem gefordert wird, dass Prämisse und Konklusion innerhalb eines Zeitfensters der Länge w auftreten müssen.

Auf Grund der oben geschilderten Probleme mit einer geänderten Regelsemantik wird im Folgenden HÖPPNERs Semantik verwendet, d. h. es werden Regeln $P \mapsto Q$ mit $P \sqsubseteq Q$ erzeugt. Dazu wird für alle Paare (P, Q) häufiger Intervallsequenzen mit $P \sqsubseteq Q$ überprüft, ob die Regel $P \mapsto Q$ die geforderte Minimalkonfidenz $\mathit{conf}_{\min} \in [0, 1]$ erreicht. Dies wird für jedes Q mit immer kürzeren Präfixmustern durchgeführt, denn wenn das Muster $P \mapsto Q$ die geforderte Minimalkonfidenz nicht erreicht, dann kann das Muster $P' \mapsto Q$ mit $P' \sqsubset P$[4] die Minimalkonfidenz auch nicht erreichen. Das folgt aus der Definition der Konfidenz und aus

$$P' \sqsubset P \;\Rightarrow\; \mathrm{supp}(P') \geq \mathrm{supp}(P).$$

Abbildung 4.10 zeigt den Algorithmus zur Erzeugung der Regeln. Dieser entspricht dem einfachen Algorithmus aus [5].

Bewertung der Regeln

Die Menge aller zeitlichen Intervallregeln ist noch größer als die Menge der häufigen Muster, da aus einem k-Muster im schlimmsten Fall $k - 1$ Regeln erzeugt werden. Es ist somit im Allgemeinen nicht möglich, alle erzeugten Regeln detailliert zu analysieren, sondern man wird sich auf die interessantesten Regeln beschränken. Bei der Suche nach den interessantesten Regeln und somit der Bewertung der Lernergebnisse kann man nach verschiedenen, subjektiven oder objektiven Kriterien vorgehen. Subjektive Kriterien werden durch den Benutzer vorgegeben und können die

[4] P' ist ein echtes Teilmuster von P.

for all $Q \in \mathcal{L}_k,\ k \geq 2$
  generate-rules(Q, Q)
end for

procedure generate-rules(Q: k-Muster, U: l-Muster)
$\mathcal{P} \leftarrow \{(l-1)\text{-Muster } P \mid P \sqsubset U\}$
for all $P \in \mathcal{P}$ **do**
  $\mathrm{conf}(P \mapsto Q) \leftarrow \frac{\mathrm{supp}(Q)}{\mathrm{supp}(P)}$
  if $(\mathrm{conf}(P \mapsto Q) \geq \mathit{conf}_{\min})$ **then**
    Gib Regel $P \mapsto Q$ aus
    if $((l-1) > 1)$ **then**
      generate-rules(Q, P)
    end if
  end if
end for

Abbildung 4.10: Der Algorithmus zur Regelerzeugung (nach [5])

Regeln zum Beispiel bzgl. des Auftretens bestimmer Intervallmarkierungen bewerten. Da subjektive Kriterien immer vom Benutzer und der zu lösenden Aufgabe abhängen, werden sie im Folgenden nicht weiter berücksichtigt. Objektive Kriterien hingegen bewerten die Regeln danach, wie gut sie die Regelmäßigkeiten in den Daten widerspiegeln. In der Literatur werden viele unterschiedliche objektive Bewertungskriterien vorgestellt – [36] gibt einen Überblick –, von denen im Folgenden drei zur Bewertung der Intervallregeln verwendet werden: Das *J-measure* [66], der *interest* [18, 67] und die *certainty factors* [15].

Smyth und Goodman stellen in [66] das *J-measure* zur Bewertung von Regeln $X \to Y$ vor. Prämisse X und Konklusion Y werden dabei als diskrete Wahrscheinlichkeitsvariablen interpretiert, und das *J-measure* vergleicht die a priori Verteilung von Y mit der a posteriori Verteilung von Y gegeben X. Dabei werden aber nur zwei Fälle unterschieden: Gegeben den Fall, dass die Prämisse X der Regel erfüllt ist $(X = x)$, wie hoch ist die Wahrscheinlichkeit, dass die Regel erfüllt $(Y = y)$ bzw. nicht erfüllt ist $(Y = \overline{y})$? Dieser Informationsgewinn, den die Kenntnis der Prämisse für Aussagen über die Konklusion liefert, ist ein spezieller Fall der relativen Entropie zweier Wahrscheinlichkeitsverteilungen und berechnet sich als

$$\begin{aligned}
\mathrm{j}(Y|X=x) :=\ & \Pr(Y=y|X=x)\log_2\left(\tfrac{\Pr(Y=y|X=x)}{\Pr(Y=y)}\right) \\
& + \Pr(Y=\overline{y}|X=x)\log_2\left(\tfrac{\Pr(Y=\overline{y}|X=x)}{\Pr(Y=y)}\right) \\
=\ & \Pr(Y=y|X=x)\log_2\left(\tfrac{\Pr(Y=y|X=x)}{\Pr(Y=y)}\right) \\
& + \big(1-\Pr(Y=y|X=x)\big)\log_2\left(\tfrac{\left(1-\Pr(Y=y|X=x)\right)}{\Pr(Y=y)}\right).
\end{aligned}$$

Das *J-measure* misst den durchschnittlichen Informationsgewinn einer Regel:

$$\mathrm{J}(Y|X=x) := \Pr(X=x)\cdot \mathrm{j}(Y|X=x).$$

Das *J-measure* wird als ein gutes Bewertungskriterium angesehen, da es eine Balance zwischen der Gültigkeit einer Regel – ihrer Konfidenz – und ihrer Häufigkeit darstellt. Es lässt sich ohne weiteres zur Bewertung von Intervallregeln $P \mapsto Q$

verwenden [38]:

$$\begin{aligned}
\mathrm{j}(P \mapsto Q) :=& \quad \mathrm{conf}(P \mapsto Q) \log_2 \left(\tfrac{\mathrm{conf}(P \mapsto Q)}{\mathrm{supp}(P)} \right) \\
&+ \big(1 - \mathrm{conf}(P \mapsto Q)\big) \log_2 \left(\tfrac{\left(1 - \mathrm{conf}(P \mapsto Q)\right)}{\mathrm{supp}(P)} \right) \\
\mathrm{J}(P \mapsto Q) :=& \ \mathrm{supp}(P) \cdot \mathrm{j}(P \mapsto Q).
\end{aligned}$$

Die Übertragung der Bewertungskriterien *interest* und *certainty factors* ist nicht so problemlos möglich. Sie wurden zur Bewertung statischer Regeln $X \to Y$ mit *disjunkter* Prämisse und Konklusion – also zum Beispiel Assoziationsregeln – entworfen und sind definiert als

$$\mathrm{interest}(X \to Y) := \frac{\mathrm{supp}(X \wedge Y)}{\mathrm{supp}(X)\mathrm{supp}(Y)}$$

$$\mathrm{cf}(X \to Y) := \begin{cases} \frac{\mathrm{conf}(X \to Y) - \mathrm{supp}(Y)}{1 - \mathrm{supp}(Y)} & \text{falls } \mathrm{conf}(X \to Y) > \mathrm{supp}(Y) \\ \frac{\mathrm{conf}(X \to Y) - \mathrm{supp}(Y)}{\mathrm{supp}(Y)} & \text{falls } \mathrm{conf}(X \to Y) < \mathrm{supp}(Y) \\ 0 & \text{sonst} \end{cases}$$

Bei zeitlichen Intervallregeln $P \mapsto Q$ enthält die Konklusion die Prämisse aber als Teilmuster, wie oben erläutert. Um *interest* und *certainty factors* dennoch zur Bewertung zeitlicher Intervallregeln verwenden zu können bezeichnet $Q \setminus P$ im Folgenden den Suffix von Q, der entsteht, wenn der Präfix P aus Q entfernt wird. *interest* und *certainty factors* werden dann wie folgt berechnet:

$$\mathrm{interest}(P \mapsto Q) := \frac{\mathrm{supp}(Q)}{\mathrm{supp}(P)\mathrm{supp}(Q \setminus P)}$$

$$\mathrm{cf}(P \mapsto Q) := \begin{cases} \frac{\mathrm{conf}(P \mapsto Q) - \mathrm{supp}(Q \setminus P)}{1 - \mathrm{supp}(Q \setminus P)} & \text{falls } \mathrm{conf}(P \mapsto Q) > \mathrm{supp}(Q \setminus P) \\ \frac{\mathrm{conf}(P \mapsto Q) - \mathrm{supp}(Q \setminus P)}{\mathrm{supp}(Q \setminus P)} & \text{falls } \mathrm{conf}(P \mapsto Q) > \mathrm{supp}(Q \setminus P) \\ 0 & \text{sonst} \end{cases}$$

Trotz dieser angepassten Definition ist es fraglich, ob *interest* und *certainty factors* genauso gut zu Bewertung von Intervallregeln geeignet sind wie zur Bewertung von statischen Regeln. Dass Problem liegt hier in der unterschiedlichen Semantik, die der Bedeutung von Regeln $P \mapsto Q$ und ihrer Bewertung durch *interest* und *certainty factors* zu Grunde liegt, s. o. Dennoch werden diese beiden Kriterien in Abschnitt 5.3.5 zur Bewertung von Intervallregeln verwendet, da sie – zumindest bei statischen Regeln – sehr gut die Korrelation zwischen der Prämisse und der Konklusion einer Regel erfassen können. Inwiefern dies auch bei Intervallregeln gelingt, wird dort erläutert.

4.4.4 Erweiterung der Hypothesensprache

Die vom Algorithmus verwendete Hypothesensprache ist zwar gut geeignet, exakte zeitliche Intervallmuster zu beschreiben, für manche Anwendungen ist sie aber zu einschränkend. So kann es zum Beispiel sinnvoll sein, nicht zwischen den Intervallrelationen A *before* B, A *meets* B und A *overlaps* B zu unterscheiden, wenn die wirkliche Relation der Intervalle A und B darin besteht, dass Intervall B nach Intervall A endet, es aber egal ist, wann genau A angefangen hat. Die wirkliche Intervallrelation zwischen A und B ist somit als Disjunktion mehrerer Grundrelationen darstellbar:

$$A\ (\textit{before} \vee \textit{meets} \vee \textit{overlaps})\ B.$$

Höppner greift diesen Punkt in [37] auf und schlägt zwei mögliche Lösungsansätze vor. Beide Ansätze beruhen auf der Verwendung einer erweiterten Hypothesensprache, unterscheiden sich aber darin, in welchem Teil des Algorithmus die erweiterte Hypothesensprache verwendet wird. Der erste Lösungsansatz besteht darin, die erweiterte Hypothesensprache schon bei der Suche nach häufigen Intervallmustern zu verwenden. Dazu müssen aber mehr häufige Intervallmuster erzeugt und überprüft werden. Diesen Nachteil vermeidet der alternative Ansatz, der die erweiterte Hypothesensprache erst *nach* der Suche nach häufigen Intervallmustern verwendet. Dabei wird in den häufigen Intervallmustern nach solchen Mustern gesucht, die sich zu einem Muster in der erweiterten Hypothesensprache kombinieren lassen. Der Nachteil dieses Ansatzes ist, dass die Mengen der Intervallsequenzen, in denen die einzelnen Muster auftreten, nicht notwendigerweise disjunkt sind. Um die Häufigkeit eines Musters in der erweiterten Hypothesensprache zu bestimmen, muss dann für jedes Muster eine Menge mit den Identifikationsnummern der Sequenzen, die dieses Muster enthalten, mit abgespeichert werden. Die Häufigkeit des Musters in der erweiterten Hypothesensprache ist dann die Kardinalität der vereinigten Identifikationsnummernmengen. Ein weiterer Nachteil des zweiten Ansatzes ist, dass es passieren kann, dass die *einzelnen* Muster die geforderte Minimalhäufigkeit nicht erreichen, das *kombinierte* Muster aber doch häufig wäre, aber auf Grund der zu geringen Häufigkeit der einzelnen Muster nicht gefunden wird.

Beide möglichen Ansätze haben somit Vor- und Nachteile. Da aber immer mehr Rechenzeit und Speicherplatz zur Verfügung stehen, wird der erste Ansatz gewählt, um einen Algorithmus mit einer erweiterten Hypothesensprache zu implementieren. Die eigentliche Vorgehensweise des Algorithmus bleibt dabei unverändert, es wird nur eine andere Darstellungsform der Intervallmuster verwendet.

Bisher wurde die Relation $A\ r\ B,\ r \in \mathcal{R}$, zwischen zwei Intervallen durch eine von Allens Intervallrelationen (im Folgenden auch als *atomare* Intervallrelationen bezeichnet) definiert. In der erweiterten Hypothesensprache werden nun Disjunktionen von Intervallrelationen verwendet:

$$A\ (r_1 \vee r_2 \vee \ldots \vee r_n)\ B,\ \ r_i \in \mathcal{R}.$$

Die atomaren Intervallrelationen bleiben somit darstellbar, und weiterhin gibt es die Möglichkeit, neue Intervallrelationen durch Kombination atomarer Relationen darzustellen. Eine wahllose Kombination atomarer Relationen ist aber nicht sinnvoll, da es exponentiell viele solcher kombinierten Intervallrelationen gibt. Deshalb werden Ideen und Konzepte von Freksa aufgegriffen, um sinnvolle Kombinationen von Intervallrelationen zu verwenden. Er definiert in [25] sogenannte *Nachbarschaften*, um die Relationen zwischen Halbintervallen darzustellen, d. h. Zeitintervallen, deren Anfangs- oder Endzeitpunkt unbekannt ist. Insgesamt werden 17 Nachbarschaftsrelationen mit Hilfe von Disjunktionen[5] atomarer Intervallrelationen definiert, wobei aber einige von anderen Relationen subsumiert werden oder Inverse anderer Relationen sind. Die Verwendung aller 17 Nachbarschaftsrelationen ist somit nicht sinnvoll, auch aus praktischen Gründen, da die Anzahl möglicher Intervallmuster exponentiell in der Anzahl der verwendeten Intervallrelationen ist (vgl. Satz 1). Deshalb wurden die fünf interessantesten der 17 Nachbarschaftsrelationen ausgewählt, um zusätzlich zu Allens atomaren Intervallrelationen erweiterte Relationen darstellen zu können. Abbildung 4.2 zeigt die ausgewählten Nachbarschaftsrelationen und ihre Definition als Disjunktion atomarer Intervallrelationen.

Um die erweiterte Hypothesensprache verwenden zu können müssen einige Teile des Algorithmus verändert werden. Die Normalform zur Darstellung der Intervallmuster muss angepasst werden, ebenso müssen die Verfahren zur Kandidatenmu-

[5] Teilweise werden im Folgenden die logische und die Mengenschreibweise für die erweiterten Intervallrelationen synonym verwendet.

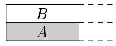
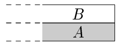
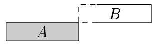
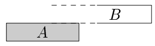
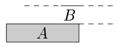
Relation	*Disjunktion*	*graphische Darstellung*
A *head-to-head / hh* B	$(s \vee eq \vee is)$	
A *tail-to-tail / tt* B	$(if \vee eq \vee f)$	
A *precedes / pr* B	$(b \vee m)$	
A *older-&-survived-by / ob* B	$(b \vee m \vee o)$	
A *older-contemporary-of / oc* B	$(o \vee if \vee c)$	

Tabelle 4.2: Erweiterte Intervallrelationen $\mathcal{R}_{\text{ext}}$ nach [25]

stererzeugung und zur Berechnung der Teilmusterrelation geändert werden. Alle restlichen Teil des Algorithmus können aber problemlos übernommen werden.

Definition 12 *Ein zeitliches Intervallmuster* (m, R) *der Größe* n *ist in* erweiterter Normalform *genau dann, wenn für alle* $1 \leq i, j \leq n$ *gilt:*

$$i < j \Leftrightarrow \quad (m(i) < m(j) \wedge R_{ij} \in \{\textit{equals, head-to-head, tail-to-tail}\}) \\ \vee \left(m(i) \neq m(j) \wedge R_{ij} \in \begin{array}{c} \{\textit{before, meets, precedes, overlaps,} \\ \textit{older-\&-survived-by, is-finished-by,} \\ \textit{contains, older-contemporary-of, starts}\} \end{array} \right)$$

Wie bisher werden sowohl die Intervallmuster als auch die Eingabesequenzen in Normalform dargestellt. Allerdings werden zur Darstellung der Eingabesequenzen weiterhin nur die atomaren Intervallrelationen verwendet.

Die Kandidatenmustererzeugung für die erweiterte Hypothesensprache läuft im Wesentlichen genauso ab wie bisher. Einzig die Verwendung von Allens Transitivitätstabelle zur Reduktion der Kandidatenmustermenge gestaltet sich etwas aufwändiger, da jede Intervallrelation jetzt aus mehreren atomaren Intervallrelationen besteht. Die möglichen Intervallrelationen zwischen den letzten beiden Intervallen p und q zweier $(k-1)$-Muster P, Q berechnet sich in der erweiterten Hypothesensprache als

$$\mathcal{R}_{\text{cand}}(P, Q) := \bigcap_{1 \leq i \leq k-2} \text{trans}\big(R_{P,(k-1),i}, R_{Q,i,(k-1)}\big),$$

wobei $\text{trans}\big(R_{P,(k-1),i}, R_{Q,i,(k-1)}\big)$ aber jetzt für die erweiterten Intervallrelationen $R_{P,(k-1),i}$, $R_{Q,i,(k-1)}$ berechnet werden muss, d. h. für jede mögliche Kombination atomarer Intervallrelationen in $R_{P,(k-1),i}$ und $R_{Q,i,(k-1)}$ wird die Menge möglicher Intervallrelationen mit Allens Transitivitätstabelle berechnet, und die Vereinigung dieser Mengen ist $\text{trans}\big(R_{P,(k-1),i}, R_{Q,i,(k-1)}\big)$.

$\mathcal{R}_{\text{cand}}(P, Q)$ bildet somit eine Maske, die jede mögliche (atomare) Intervallrelation zwischen p und q enthält, und für jede Intervallrelation wird überprüft, ob diese Maske alle Disjunktionsglieder der Relation enthält. Falls ja, so wird ein entsprechendes Kandidatenmuster erzeugt.

Der letzte Punkt in dem sich der Algorithmus mit der erweiterten Hypothesensprache vom bisherigen Algorithmus unterscheidet, ist die Berechnung der Teilmusterrelation $\sqsubseteq$. Hier wird aber nur eine Zeile des Algorithmus verändert (vgl. Abbildung 4.9):

$ok \leftarrow ok \wedge \big(R_{P,i,k} = R_{Q,j,\pi(k)}\big)$

wird zu

$ok \leftarrow ok \wedge \big((R_{P,i,k} \cap R_{Q,j,\pi(k)}) \neq \emptyset\big)$

Ob die erweiterte Hypothesensprache besser geeignet ist, die Regelmäßigkeiten in den vorliegenden Versicherungsvertragsdaten zu erfassen, wird in Abschnitt 5.3.4 untersucht.

5 – Wissensentdeckung in Versicherungsvertragsdaten

In diesem Kapitel wird die Durchführung der in Abschnitt 3.3 skizzierten Data Mining-Aufgaben erläutert und die erzielten Ergebnisse werden vorgestellt. Gleichzeitig wird eine Anleitung zur Verwendung des Data Mining-Algorithmus gegeben. Diese Anleitung soll zum einen den allgemeinen Anwendungsfall des Algorithmus an einem konkreten Beispiel erläutern, zum anderen sollen aber auch Alternativen zu den hier getroffenen Entscheidungen aufgezeigt werden, die in diesem Wissensentdeckungsprozess aus verschiedensten Gründen nicht gewählt wurden, bei der Analyse anderer Daten aber durchaus sinnvoll wären.

Der verwendete Data Mining-Algorithmus wurde in der Programmiersprache Java implementiert. Obwohl Java-Programme immer noch um den Faktor 5–10 langsamer sind als vergleichbare C- oder C++-Programme hat Java den Vorteil der automatischen Speicherverwaltung, was die oftmals mühselige Suche nach Fehlern in der Speicherverwaltung weitgehend überflüssig macht und die Entwicklung großer Programme erheblich erleichtert. Zudem besitzt Java mit JDBC eine standardisierte Zugriffsmöglichkeit auf Datenbanksysteme, was eine unabdingbare Voraussetzung für die vorliegende Arbeit ist.

5.1 Anwendung des Algorithmus

Wie Abbildung 2.1 verdeutlicht, sind vor der eigentlichen Analyse von Daten mit Hilfe eines Data Mining-Algorithmus einige Vorverarbeitungsschritte notwendig. Für die hier zu lösende Lernaufgabe des Findens häufiger Intervallmuster bedeutet dies, dass man sich überlegen muss, wie die Eingabesequenzen für den Algorithmus definiert werden, d. h. welche Objekte durch die Intervallsequenzen dargestellt werden, durch welche Variablen die Intervallgrenzen und die Markierung der Intervalle definiert werden. In den im Folgenden beschriebenen Untersuchungen definieren zum Beispiel einzelne Tarifkomponenten oder ganze Versicherungsverträge, die aus mehreren Tarifkomponenten bestehen, die Objekte, die nach häufigen Intervallmustern durchsucht werden. Wie die Vorverarbeitung zur Erzeugung der Eingabesequenzen in den einzelnen Fällen genau gelöst wird, erläutern die entsprechenden Abschnitten detailliert.

Ein weiterer Aspekt, der bei der Vorverarbeitung berücksichtigt werden muss, ist, dass der Algorithmus in den Eingabesequenzen nach Intervallmustern sucht, die die zeitliche Entwicklung der Sequenzen möglichst gut widerspiegeln. Die Eingabesequenzen sollten also zu einer möglichst homogenen Gruppe gehören, damit aussagekräftige Ergebnisse erzielt werden. Gibt es in den Daten vorab bekannte

Untergruppen, so sollten die Daten gemäß dieser Untergruppenstruktur unterteilt und die einzelnen Partitionen der Daten getrennt analysiert werden.

Im vorliegenden Anwendungsfall wird zum Beispiel bei der Analyse der Tarifkomponenten und der Versicherungsverträge die Wissensentdeckung für jede Versicherungsart getrennt durchgeführt, d. h. die Daten in den Tabellen werden bzgl. der fünf Versicherungsarten (vgl. Abschnitt 3.1) aufgeteilt und in neue Tabellen kopiert. Der Grund für diese Trennung ist, dass die Anzahl und Art der Änderungen zwischen den verschiedenen Versicherungsarten variiert, da jede Versicherungsart andere typische Änderungsprofile aufweist. Tabelle 5.1 zeigt als Beispiel die durchschnittliche Anzahl an Änderungen pro Versicherungsvertrag bzw. Tarifkomponente, zum einen aufgeteilt nach Versicherungsart[1] und zum anderen ingesamt. Es fällt sofort auf,

Tabelle	*KAV*	*REV*	*KRV*	*EUV*	*FGV*	*gesamt*
MO_VVERT	6,02	4,11	2,03	6,29	29,90	6,76
MO_TFKOMP	3,91	2,94	1,85	3,95	14,45	4,12

Tabelle 5.1: Durchschnittliche Anzahl an Änderungen der Versicherungen und Tarifkomponenten

dass die durchschnittliche Anzahl an Änderungen zwischen den einzelnen Versicherungsarten deutlich variiert. Die Krankenversicherungen weisen durchschnittlich am wenigsten Änderungen auf, während die Fondsgebundenen Lebensversicherungen die mit Abstand meisten Änderungen aufweisen. Weiterhin sind die Änderungsgründe von Versicherungsart zu Versicherungsart teilweise unterschiedlich, sodass das Data Mining für jede Versicherungsart einzeln durchgeführt werden muss, damit aussagekräftige Ergebnisse erzielt werden.

Nachdem die Daten für den Data Mining-Algorithmus vorbereitet wurden, werden die ersten Analysen durchgeführt, vgl. Abschnitt 5.2.1 und 5.3.1. Da das hier verwendete Verfahren eine Weiterentwicklung anderer Algorithmen darstellt, werden zuerst einige Versuche mit unterschiedlichen Parametereinstellungen durchgeführt, um den Einfluss der Parameter auf die Anzahl der gefundenen Regeln herauszufinden. Weiterhin werden die optimierte und die nichtoptimierte Variante des Algorithmus bzgl. ihrer Laufzeit und der Gesamtzahl an erzeugten Kandidatenmustern verglichen.

Im Allgemeinen wird man bei der Datenanalyse aber nicht mit einer bestimmten Anzahl an Regeln zufrieden sein, sondern man ist an der Bedeutung der Regeln interessiert, s. Abschnitt 5.2.2 und 5.3.3. Die große Menge an gefundenen Regeln macht es dabei meistens unmöglich, alle Regeln detailliert zu analysieren. Stattdessen wird man sich auf die interessantesten Regeln konzentrieren wollen. Was eine Regel interessant macht, hängt natürlich immer vom Benutzer ab, d. h. in manchen Fällen wird man die Regeln auf bestimmt Strukturen hin untersuchen. Hat man aber keine Vorstellung davon wie interessante Regeln aussehen könnten, so wird man objektive Regelbewertungskriterien zur Sortierung der Regeln verwenden. Da es in der Literatur viele verschiedene Bewertungskriterien gibt, werden in Abschnitt 5.3.5 einige Bewertungskriterien auf ihre Anwendbarkeit hin miteinander verglichen.

Obwohl die normale Hypothesensprache des Algorithmus gut geeignet ist, die Regelmäßigkeiten in den analysierten Daten zu erfassen, ist sie für manche Anwendungen zu strikt, vgl. Abschnitt 4.4.4. Deshalb wird in Abschnitt 5.3.4 untersucht,

[1] Die unterschiedlichen Versicherungsarten werden im Folgenden als KAV (Kapitallebensversicherungen), REV (Rentenversicherungen), KRV (Krankenversicherungen), EUV (Erwerbsunfähigkeitsversicherungen) und FGV (Fondsgebundene Lebensversicherungen) abgekürzt.

ob eine Erweiterung der Hypothesensprache vielleicht noch besser geeignet ist, die in den Daten vorhandenen Muster zu erfassen.

Die Regeln, die der hier angewandte Data Mining-Algorithmus aus den Eingabedaten lernt, beschreiben die zeitlichen Muster oder Änderungen der Eingabesequenzen. Nun können sich die Muster selber aber mit der Zeit verändern, d. h. die am Anfang der Eingabesequenzen vorkommenden Muster können anders sein als die am Ende auftretenden Muster. Da die Hypothesensprache des Algorithmus diesen Aspekt nicht erfasst, muss man versuchen, durch entsprechende Vorverarbeitung die Daten so zu filtern, dass sie auf Veränderungen der Muster mit der Zeit untersucht werden können. Die entsprechenden Analysen werden in Abschnitt 5.3.6 beschrieben.

5.2 Erste Versuche – Lernen aus den Versicherungsvertragsdaten

Mit den ersten durchgeführten Versuchen soll überprüft werden, ob es in den Versicherungsvertragsdaten interessante Sequenzen oder Muster gibt (vgl. Abschnitt 3.1). Es werden somit nur Daten aus der Tabelle MO_VVERT für die Wissensentdeckung verwendet, da diese Tabelle die gesamte Lebensgeschichte eines Versicherungsvertrags enthält. Die Daten über die Kunden und die Rollen, die diese während der Laufzeit eines Versicherungsvertrages annehmen, werden nicht verwendet.

Jeder Versicherungsvertrag bzw. jede Tarifkomponente stellt eine Eingabesequenz dar, und jede Änderung eines Vertrags bzw. einer Tarifkomponente stellt ein Intervall der Eingabesequenzen dar. Die Intervallgrenzen werden den Datumsangaben VVBEG/VVEND entnommen, und der Änderungsgrund (Spalte VVAENDART) definiert die Intervallmarkierung.

5.2.1 Einfluss der Parameter auf die Anzahl der gefundenen Regeln

Da der Data Mining-Algorithmus in der hier verwendeten Form noch nicht implementiert und angewandt wurde, soll zuerst überprüft werden, wieviele Regeln bei unterschiedlichen Einstellungen der beiden Parameter Minimalhäufigkeit ($supp_{\min}$) und Minimalkonfidenz ($conf_{\min}$) gefunden werden. Diese Untersuchung hat für die Ergebnisse der weiteren Datenanalyse nur insofern Relevanz, als sie Hinweise für „günstige“ Parametereinstellungen liefern kann, d. h. Grenzwerte für die Parameter, jenseits derer zuwenig oder zuviele Regeln gefunden werden. Werden sehr wenige Regeln in den Daten gefunden, so enthalten diese möglicherweise keine interessanten Informationen. Werden hingegen sehr viele Regeln gefunden, so wird man nicht alle Regeln detailliert analysieren können. Es ist deshalb wichtig, die Parametereinstellungen so zu wählen, dass zwar die interessanten Informationen wahrscheinlich gefunden werden, diese aber nicht durch viele uninteressante Regeln „verdeckt“ werden.

Abbildung 5.1 zeigt die Anzahl der in den jeweiligen Versicherungsdaten gefundenen Regeln in Abhängigkeit der gewählten Parameter auf einer logarithmischen Skala. Bei näherer Betrachtung von Abbildung 5.1 fällt auf, dass in den meisten Daten erst ab einer Minimalhäufigkeit von 10–20% Regeln gefunden werden. Mit niedriger gewählter Minimalhäufigkeit nimmt die Anzahl der gefundenen Regeln dann exponentiell zu, wobei bei gleichbleibender Minimalhäufigkeit und sinkender Minimalkonfidenz die Anzahl der gefundenen Regeln nicht ganz so schnell wächst. Sehr deutlich sieht man auch, dass bei steigender Komplexität der Eingabesequenzen viel mehr Regeln gefunden werden. In den FGV-Daten mit durchschnittlich 29,90

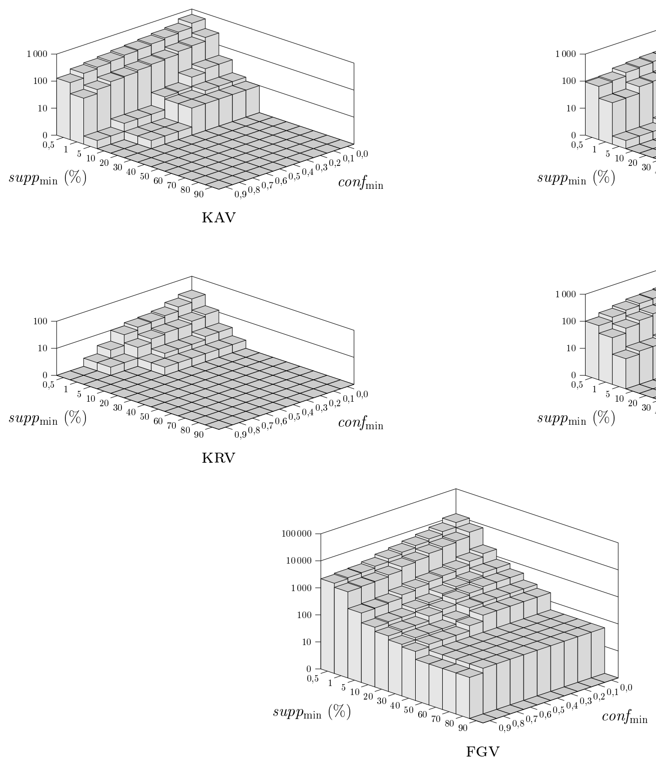


Abbildung 5.1: Anzahl der in den Versicherungsvertragsdaten gefundenen Regeln, in Abhängigkeit der Parametereinstellungen

Änderungen pro Versicherungsvertrag werden viel mehr Regeln gefunden als in den KAV-Daten mit durchschnittlich 6,02 Änderungen.

Die Menge der gefundenen Regeln hängt somit außer von den beiden direkt beeinflussbaren Parametern Minimalhäufigkeit und Minimalkonfidenz auch sehr stark von den Daten ab. Die Anzahl der möglichen Muster ist exponentiell in der Anzahl der unterschiedlichen Intervallmarkierungen und der Länge der Eingabesequenzen. Die Anzahl an unterschiedlichen Intervallmarkierungen, die vom Algorithmus zur Konstruktion der Muster verwendet werden, hängt wiederum von den Daten und der gewählten Minimalhäufigkeit ab, ebenso wie die maximale Länge der betrachteten Muster.

Die Anzahl an Regeln, die aus der Menge der häufigen Sequenzen erzeugt werden, ist schließlich noch größer als die Anzahl häufiger Sequenzen, da im Extremfall bei einer Minimalkonfidenz von 0,0 aus jeder Sequenz der Länge l $l-1$ Regeln erzeugt werden.

Aufgrund der komplexen Abhängigkeit zwischen den Parametern, den Daten und der Anzahl der gefundenen Regeln wird man nicht erwarten können, eine einfache Formel für die Anzahl an Regeln, in Abhängigkeit von den gewählten Parametern, angeben zu können. Trotzdem kann man aber vielleicht einige Daumenregeln für „günstige“ Parametereinstellungen finden.

Als eine mögliche Daumenregel für die Wahl der Parameter des Algorithmus kann man angeben, dass die Minimalhäufigkeit bei langen oder komplexen Eingabesequenzen erstmal hoch gewählt werden sollte (50–80%), um eine überschaubare Menge an Regeln zu erhalten. Um mehr Regeln zu bekommen kann die Minimalhäufigkeit schrittweise gesenkt werden, wobei man aber beachten sollte, dass anscheinend gerade ab einer Minimalhäufigkeit von 10% das exponentielle Wachstum die Anzahl der gefundenen Regeln stark in die Höhe treibt. Die Minimalkonfidenz dagegen hat keinen so starken Einfluss auf die Menge der gefundenen Regeln. Sie kann sogar auf 0,0 gesetzt werden, um so alle Regeln mit der gewählten Minimalhäufigkeit zu berechnen. In einem Nachbearbeitungsschritt können dann die Regeln mit einer gewissen Minimalkonfidenz oder anderen Eigenschaften aus der Menge aller Regeln herausgefiltert werden.

5.2.2 Analyse der Versicherungsvertragsdaten

Um neue, interessante Informationen über die zeitliche Entwicklung der Versicherungsverträge – oder allgemein aus einer Menge von Intervallsequenzen – zu gewinnen reicht es aber nicht aus, eine bestimmte Menge an Regeln zu lernen. Erst die Interpretation dieser Regeln liefert die interessanten Ergebnisse der Wissensentdeckung. Im Folgenden werden deshalb die Regeln aus den in Abschnitt 5.2.1 geschilderten Data Mining-Versuchen näher analysiert. Dazu werden alle Regeln mit einer Minimalhäufigkeit von 10% und einer Minimalkonfidenz von 0,0 mit Hilfe des *J-measures* bewertet und die jeweils 20 am höchsten bewerteten Regeln werden weiter untersucht.

Diese ausführliche Auswertung der gefundenen Regeln zeigt, dass im ersten Versuch gar keine zeitlichen Muster und Regeln gefunden wurden, sondern fast nur Mengen von in ihrer zeitlichen Ausdehnung identischen Intervallen, sodass die Regeln zu einfachen nichtzeitlichen Assoziationsregeln degeneriert sind. Abbildung 5.2 zeigt das grundlegende Schema der gefundenen Regeln: Der größte Teil der Regeln besteht aus zwei oder mehr zeitgleichen Intervallen, wobei bei einigen Regeln das erste Intervall vor den restlichen endet.

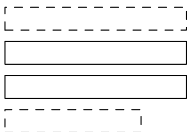


Abbildung 5.2: Schema der gefundenen Regeln aus den ersten Versuchen

Da die zeitlichen Intervallbeziehungen bei der Suche nach häufigen Mustern somit keine Rolle spielen, treten die häufigen Intervalle in vielen möglichen Kombinationen auf, wodurch die Anzahl der möglichen Regeln natürlich stark erhöht wird.

Eine Stichprobe von jeweils 20 Versicherungsverträgen je Versicherungsart macht klar, warum die gefundenen Regeln die in Abbildung 5.2 skizzierte Form haben. Abbildung 5.3 zeigt als verdeutlichende Beispiele die zeitliche Entwicklung dreier Versicherungen, einer Kapitallebensversicherung (a), einer Fondsgebundenen Lebensversicherung[2] (b) und einer Krankenversicherung (c). Jede Zeile zeigt den Gültigkeitszeitraum einer Änderung der Versicherung, wobei die nichtschraffierten Rechtecke den Gültigkeitszeitraum anhand der Datumsangaben VVBEG/VVEND darstellen, während die grau schraffierten Rechtecke den Gültigkeitszeitraum anhand der Datumsangaben VVWIVON/VVWIBIS anzeigen. Eine unbekannte Datumsangabe ist durch eine „Ausfransung“ gekennzeichnet.

Es zeigt sich, dass die Zeitangaben VVBEG und VVEND falsch interpretiert wurden. Sie geben die Gesamtzeitdauer eines Versicherungsvertrags an, nicht den Gültigkeitszeitraum einer einzelnen Änderung. Um typische Profile von Versicherungen zu

[2] In Teilabbildung (b) sind aus Platzgründen in den Zeilen „1004^+“ und „1003^+“ Intervalle mit der gleichen Markierung in einer Zeile dargestellt.

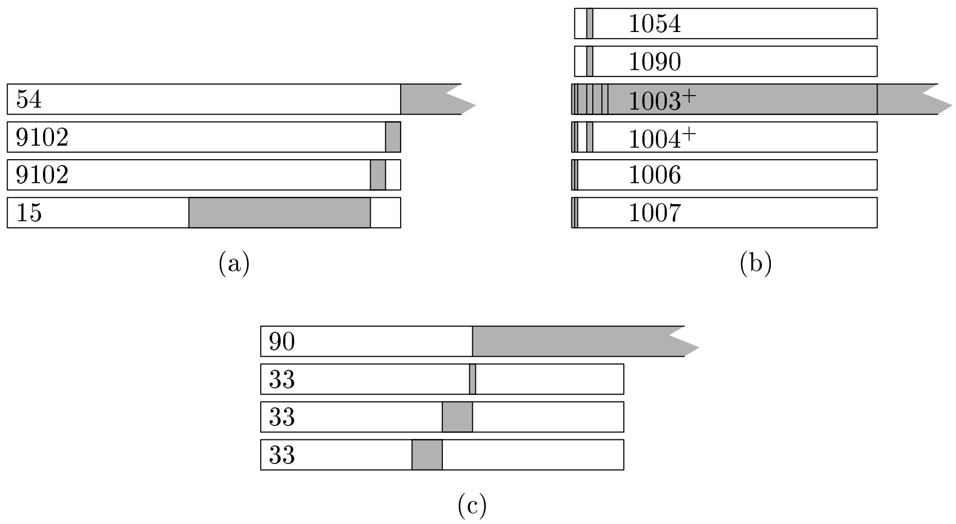


Abbildung 5.3: Die zeitliche Abfolge der Einträge dreier Beispielversicherungen

finden, müssen also die Datumsangaben aus VVWIVON/VVWIBIS als Gültigkeitszeitraum der Vertragsänderungen gewählt werden.

Ein weiteres Ergebnis der Stichprobenanalyse ist, dass die Einträge aus den Tabellen MO_VVERT nicht ohne weitere Vorverarbeitung analysiert werden können. Jede Änderung einer Tarifkomponente verursacht nämlich einen Eintrag in Tabelle MO_TFKOMP *und* in Tabelle MO_VVERT. Eine Änderung, die nur den Versicherungsvertrag als Ganzes betrifft, verursacht hingegen nur einen Eintrag in MO_VVERT. Um die Daten aus Tabelle MO_VVERT analysieren zu können, müssen die Intervalldaten deshalb so vorverarbeitet werden, dass die Änderungen den betroffenen Tarifkomponenten und/oder dem gesamten Vertrag zugeordnet werden können.

Diese ersten Ergebnisse zeigen sehr deutlich den iterativen Charakter des Wissensentdeckungsprozesses: Häufig werden Daten anfangs falsch interpretiert oder nicht richtig vorverarbeitet, und dies fällt erst bei der genaueren Analyse der Ergebnisse auf. Diese Erkenntnisse werden dann im nächsten Durchlauf des KDD-Prozesses genutzt, um die Vorverarbeitungsschritte zu korrigieren. Für den hier vorgestellten Anwendungsfall heißt das, dass sich die nächsten Versuche auf die Analyse der Tarifkomponenten konzentrieren, da diese den interessanten Kern der Versicherungsvertragsdaten darstellen. Erst in späteren Versuchen wird auf der Ebene der Versicherungsverträge nach Regelmäßigkeiten gesucht.

5.3 Wissensentdeckung in den Tarifkomponentendaten

In den nächsten Versuchen wird in den Tarifkomponentendaten nach typischen Änderungsprofilen gesucht. Da die in Abschnitt 5.2.1 vorgestellten Ergebnisse wegen der falsch gelösten Vorverarbeitung nicht verallgemeinbar sind, wird im ersten Versuch nochmals der Einfluss der Parameter auf die Anzahl an gefundenen Regeln überprüft. Weiterhin werden die Laufzeiteigenschaften des optimierten und des nichtoptimierten Algorithmus miteinander verglichen. Anschließend werden die Regeln näher analysiert, wobei gleichzeitig verschiedene Bewertungskriterien miteinander verglichen werden. In weiteren Versuchen werden die Daten gefiltert, um zum Beispiel zu überprüfen, ob die Änderungen einer Tarifkomponente selber zeitlichen Veränderungen unterworfen sind, oder ob der Rückkauf eines Vertrags und damit einer Tarifkomponente aus den Änderungen vorhersagbar ist.

5.3.1 Einfluss der Parameter

Wie in Abschnitt 5.2.1 geschildert wird überprüft, welchen Einfluss die beiden Parameter Minimalhäufigkeit und Minimalkonfidenz auf die Anzahl der gefundenen Regeln haben. Abbildung 5.4 zeigt die Anzahl der in den jeweiligen Tarifkomponentendaten gefundenen Regeln in Abhängigkeit der gewählten Parameter auf einer logarithmischen Skala.

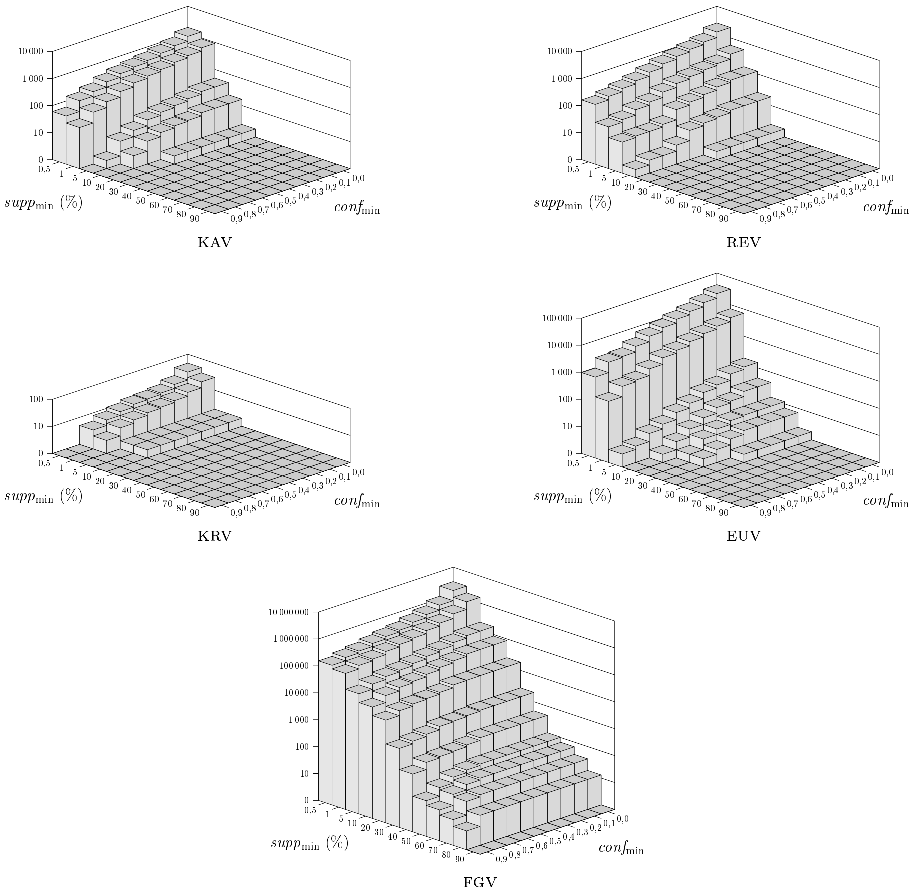


Abbildung 5.4: Anzahl der in den Tarifkomponentendaten gefundenen Regeln, in Abhängigkeit der Parametereinstellungen

Abbildung 5.4 ähnelt sehr stark Abbildung 5.1, wenn auch in fast allen Daten mehr Regeln gefunden wurden als im ersten, falsch durchgeführten Versuch. Das ist zunächst verwunderlich, da die Tarifkomponenten nach Tabelle 5.1 durchschnittlich weniger Änderungen aufweisen als die Versicherungsverträge. Im ersten Versuch wurden die Daten aber noch so vorverarbeitet, dass die Maximalitätsbedingung (vgl. Abschnitt 4.3) erfüllt war. Dadurch wurden einige Intervalle der Eingabesequenzen zu einem einzigen Intervall zusammengefasst. Die Maximalitätsbedingung wurde außerdem in der Hypothesensprache des Data Mining-Algorithmus verwendet, um die Menge an möglichen häufigen Sequenzen zu beschränken (vgl. Abschnitt 4.4.1). Die Analyse der ersten Versuche ergab aber, dass die Maximalitätsbedingung und die daraus resultierende Einschränkung der Hypothesensprache für die mit dem hier verwendeten Algorithmus lösbaren Lernaufgaben nicht sinnvoll ist. Aufgrund der veränderten Hypothesensprache werden somit trotz der kürzeren Eingabesequenzen mehr Regeln gefunden als im ersten Versuch. Welche Auswirkungen die geänderte Hypothesensprache auf die gefundenen Regeln hat, wird im nächsten Abschnitt erläutert.

Trotz der veränderten Vorverarbeitung und der geänderten Hypothesensprache werden die in Abschnitt 5.1 aufgestellten Daumenregeln durch diesen Versuch bestätigt. Man sollte somit bei langen oder komplexen Eingabesequenzen den Algorithmus zuerst mit einer hohen Minimalhäufigkeit laufen lassen, um eine überschaubare Menge an Regeln und einen ersten groben Überblick über die Daten zu bekommen. In weiteren Versuchen kann die Minimalhäufigkeit dann abgesenkt werden, um auch nicht so häufige, aber dafür vielleicht interessantere Regelmäßigkeiten zu finden. Wie eine große Regelmenge nach interessanten Regeln durchsucht werden kann und welche Alternativen es bei der Analyse gibt, wird in den Abschnitten 5.3.3 und 5.3.5 erläutert.

5.3.2 Laufzeiteigenschaften der Algorithmen

Neben dem Einfluss der Parameter des Algorithmus auf die Anzahl der gefundenen Regeln ist auch die Laufzeit und die Anzahl der erzeugten Kandidatenmuster in Abhängigkeit der Parameter für die Anwendung des Algorithmus von Interesse. In Abschnitt 4.4.2 wurde zum Beispiel gesagt, dass die Erzeugung der Kandidatenmuster und die Berechnung der Häufigkeit der Kandidatenmuster durch die Verwendung zusätzlicher Datenstrukturen optimiert werden kann. Die Verwaltung dieser Datenstrukturen benötigt allerdings auch Rechenzeit und Speicherplatz. Im Folgenden soll nun versucht werden, einige Hinweise zu geben, wann die optimierte und wann die nichtoptimierte Version des Algorithmus verwendet werden sollte.

Abbildung 5.5 zeigt die Anzahl der bei verschiedenen Minimalhäufigkeiten von beiden Algorithmen erzeugten Kandidatenmuster. Werden weniger Kandidatenmu-

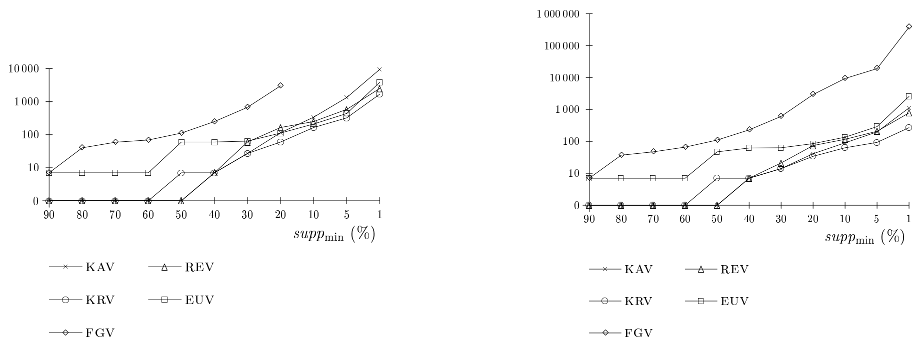


Abbildung 5.5: Anzahl der von der nichtoptimierten (links) und der optimierten Variante (rechts) des Data Mining-Algorithmus erzeugten Kandidatenmuster in Abhängigkeit der Minimalhäufigkeit

ster erzeugt, dann muss auch die Häufigkeit für weniger Muster berechnet werden. Abbildung 5.5 zeigt deutlich, dass der optimierte Algorithmus gerade bei niedrigen Minimalhäufigkeiten weitaus weniger Kandidatenmuster erzeugt als der nichtoptimierte Algorithmus. So musste zum Beispiel die Analyse der FGV-Daten mit dem nichtoptimierten Algorithmus bei einer Minimalhäufigkeit von 10% abgebrochen werden, da so viele Kandidatenmuster erzeugt wurden, dass die Berechnung der

Häufigkeit zuviel Zeit in Anspruch genommen hätte. Bei hohen Minimalhäufigkeiten bis ca 30% unterscheidet sich die Anzahl der von beiden Algorithmen erzeugten Kandidatenmuster aber nicht sehr stark. Werden die Daten also mit einer hohen Minimalhäufigkeit analysiert, so sollte der nichtoptimierte Algorithmus verwendet werden. Bei niedrigen Minimalhäufigkeiten und vielen Eingabesequenzen sollte aber, sofern genug Speicher zur Verfügung steht, der optimierte Algorithmus benutzt werden. Abbildung 5.6 verdeutlicht das nochmals an Hand der Laufzeit der beiden Algorithmen. Zur Bestimmung der Laufzeit wurden beide Varianten des Algorithmus mit dem Sun JDK 1.3.1 auf einem Athlon 2x MP 2100+-System mit 1024 MB Hauptspeicher unter Linux ausgeführt. Von den beiden Prozessoren des Rechners wurde allerdings nur einer genutzt. Die angegebenen Laufzeiten stellen allerdings nur die Ergebnisse eines einzigen Durchlaufs dar und können durch andere Einflüsse wie Nutzung des Rechners durch andere Benutzer beeinträchtigt sein. Dennoch geben sie die grundlegenden Laufzeiteigenschaften der beiden Algorithmusvarianten wieder.

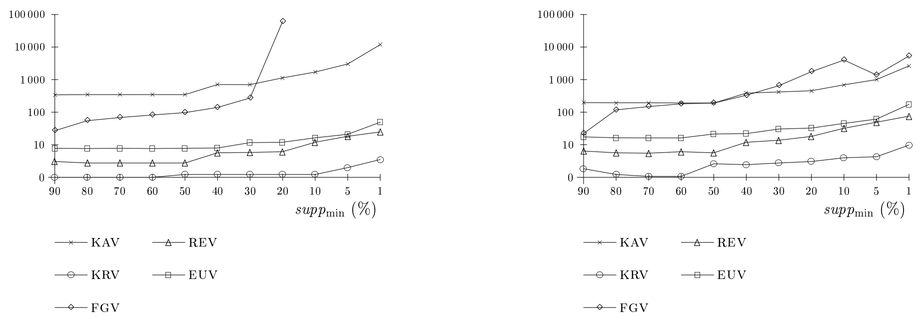


Abbildung 5.6: Laufzeit (in Sekunden) der nichtoptimierten (links) und der optimierten Variante (rechts) des Data Mining-Algorithmus in Abhängigkeit der Minimalhäufigkeit

Abbildung 5.6 zeigt, dass auch die Verwaltung der Datenstrukturen des optimierten Algorithmus einige Rechenzeit benötigt. Deutlich sieht man das an den Laufzeiten der Algorithmen bei der Analyse der REV-, KRV- und EUV-Daten. Hier benötigt der optimierte Algorithmus immer mehr Zeit als die nichtoptimierte Variante, obwohl er gerade bei niedrigen Minimalhäufigkeiten weniger Kandidatenmuster überprüfen muss. Da die REV-, KRV- und EUV-Daten nur 4 000–12 000 Eingabesequenzen umfassen, ist der durch die Optimierung erzielte Laufzeitgewinn nicht groß genug, um die durch die Verwaltung der Datenstrukturen verursachte Laufzeiteinbuße aufzuwiegen.

Anders sieht es bei der Analyse der KAV- und FGV-Daten aus. Bei der Analyse der KAV-Daten ist der optimierte Algorithmus immer schneller als der nichtoptimierte, da diese Daten fast 500 000 Eingabesequenzen umfassen und die Optimierung hier einen entsprechenden Laufzeitgewinn erzielt. Die FGV-Daten schließlich lassen sich erst mit dem optimierten Algorithmus auch bei niedrigen Minimalhäufigkeiten analysieren, da durch die Optimierung die Anzahl der Kandidatenmuster eingeschränkt wird.

Als Daumenregel kann man somit sagen, dass bei hohen Minimalhäufigkeiten (über 30%) der optimierte Algorithmus verwendet werden sollte. Bei niedrigen Minimalhäufigkeiten muss man je nach Art der zu analysierenden Daten abwägen. Bei vielen oder komplexen Eingabesequenzen bringt die Verwendung des optimierten Algorithmus einen Laufzeitgewinn, wenn auch auf Kosten des Speicherverbrauchs. Bei wenigen Eingabesequenzen ist hingegen in der Regel die nichtoptimierte Variante schneller.

5.3.3 Analyse der Tarifkomponentendaten

Wie in Abschnitt 5.2.2 schon gesagt reicht es bei der Suche nach häufigen Sequenzen nicht aus, einfach eine bestimmte Menge an Regeln zu finden, sondern die gefundenen Regeln müssen – am besten von einem Experten – daraufhin überprüft werden, ob sie neues, interessantes Wissen darstellen. Deshalb werden im Folgenden die Regeln aus den in Abschnitt 5.3.1 geschilderten Data Mining-Versuchen näher analysiert. Dazu werden alle Regeln mit einer Minimalhäufigkeit von 10% bzw. 1% und einer Minimalkonfidenz von 0,0 mit Hilfe des *J-measures* bewertet und die jeweils 20 am höchsten bewerteten Regeln werden weiter untersucht. Die 10%-Minimalhäufigkeit wurde gewählt, da erst bei dieser Minimalhäufigkeit auch in den KRV-Daten Regeln gefunden wurden, und weil Regeln mit einer Häufigkeit über 10% einen groben Überblick über die Regelmäßigkeiten in den Daten geben. Bei einer Minimalhäufigkeit von nur 1% werden zwar weitaus mehr Regeln gefunden als bei einer Minimalhäufigkeit von 10%, solche seltenen Regeln könnten aber interessanteres Wissen darstellen als die häufigeren, allgemeineren Regeln, weshalb auch Regeln mit dieser geringen Häufigkeit näher untersucht werden.

Als einen ersten Überblick über die gefundenen Regeln kann man sagen, dass keine wirklich überraschenden oder interessanten Regelmäßigkeiten gefunden wurden, weder in den Regeln mit einer Minimalhäufigkeit von 10% noch in denen mit 1%. Tabelle 5.2 zeigt zur Verdeutlichung exemplarisch fünf Regeln pro Versicherungsart.

Tabelle 5.2: Fünf Beispielregeln pro Versicherungsart aus der Analyse der Tarifkomponentenänderungen

KAV	Neueintritt(I_1), Nachführen_aktuelle_Prämie_EVBS(I_2), I_1 *meets* I_2 $\mapsto$ Nachführen_aktuelle_Prämie_EVBS(I_3), I_1 *before* I_3, I_2 *meets* I_3
	Neueintritt(I_1) $\mapsto$ Nachführen_aktuelle_Prämie_EVBS(I_2), I_1 *meets* I_2
	Nachführen_aktuelle_Prämie_EVBS(I_1) $\mapsto$ Nachführen_aktuelle_Prämie_EVBS(I_2), I_1 *meets* I_2
	Antragsannahme(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *meets* I_2 $\mapsto$ kalkulierte_Prämien-/Leistungsanpassung(I_3), I_1 *before* I_3, I_2 *meets* I_3
	sonstige_planmäßige_Änderung(I_1) $\mapsto$ sonstige_planmäßige_Änderung(I_2), I_1 *meets* I_2
REV	Fortschreibung_Überschuss_vorschüssig(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *meets* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *meets* I_3, I_2 *starts* I_3

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

	Fortschreibung_Überschuss_vorschüssig(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *before* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *starts* I_3
	Antragsannahme(I_1), Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *starts* I_2, kalkulierte_Prämien-/Leistungsanpassung(I_3), I_1 *before* I_3, I_2 *meets* I_3 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_4), I_1 *before* I_4, I_2 *meets* I_4, I_3 *starts* I_4
	Beginn_des_Rentenbezugs(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2
	Fortschreibung_Überschuss_vorschüssig(I_1) $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *meets* I_2
KRV	Neueintritt(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2
	Tarifanpassung(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2
	Einschluss_einer_Person(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2
	Tarifanpassung(I_1), Tarifanpassung(I_2), I_1 *meets* I_2 $\mapsto$ Richtigstellung_von_Daten_in_DB(I_3), I_1 *before* I_3, I_2 *meets* I_3
	Übertritt_zu_SANREMO(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2
EUV	kalkulierte_Prämien-/Leistungsanpassung(I_1) $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *starts* I_2
	Antragsannahme(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *meets* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *meets* I_3, I_2 *starts* I_3
	Dynamik(I_1) $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *starts* I_2

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

	Dynamik(I_1), sonstige_planmäßigeÄnderung(I_2), I_1 *equals* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *starts* I_3, I_2 *starts* I_3
	Antragsannahme(I_1), Dynamik(I_2), I_1 *meets* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *meets* I_3, I_2 *starts* I_3
FGV	Fortschreibung_Reserve(I_1), Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *before* I_2, Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *equals* I_3, Fortschreibung_Reserve(I_4), I_1 *before* I_4, I_2 *equals* I_4, I_3 *equals* I_4 $\mapsto$ Fortschreibung_Reserve(I_5), I_1 *before* I_5, I_2 *equals* I_5, I_3 *equals* I_5, I_4 *equals* I_5, Fortschreibung_Reserve(I_6), I_1 *before* I_6, I_2 *equals* I_6, I_3 *equals* I_6, I_4 *equals* I_6, I_5 *equals* I_6, Fortschreibung_Reserve(I_7), I_1 *before* I_7, I_2 *equals* I_7, I_3 *equals* I_7, I_4 *equals* I_7, I_5 *equals* I_7, I_6 *equals* I_7
	Fortschreibung_Reserve(I_1), Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *before* I_2, Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *equals* I_3, Fortschreibung_Reserve(I_4), I_1 *before* I_4, I_2 *equals* I_4, I_3 *equals* I_4 $\mapsto$ Fortschreibung_Reserve(I_5), I_1 *before* I_5, I_2 *equals* I_5, I_3 *equals* I_5, I_4 *equals* I_5, Fortschreibung_Reserve(I_6), I_1 *before* I_6, I_2 *equals* I_6, I_3 *equals* I_6, I_4 *equals* I_6, I_5 *equals* I_6
	Fortschreibung_Reserve(I_1), Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *before* I_2, Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *equals* I_3, Fortschreibung_Reserve(I_4), I_1 *before* I_4, I_2 *equals* I_4, I_3 *equals* I_4 $\mapsto$ Fortschreibung_Reserve(I_5), I_1 *before* I_5, I_2 *equals* I_5, I_3 *equals* I_5, I_4 *equals* I_5, Fortschreibung_Reserve(I_6), I_1 *before* I_6, I_2 *equals* I_6, I_3 *equals* I_6, I_4 *equals* I_6, I_5 *equals* I_6, Datenübernahme_aus_ALFI(I_7), I_1 *before* I_7, I_2 *equals* I_7, I_3 *equals* I_7, I_4 *equals* I_7, I_5 *equals* I_7, I_6 *equals* I_7

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

Fortschreibung_Reserve(I_1),
Fortschreibung_Überschuss_vorschüssig(I_2),
I_1 *before* I_2,
Fortschreibung_Überschuss_vorschüssig(I_3),
I_1 *before* I_3, I_2 *equals* I_3,
Fortschreibung_Reserve(I_4),
I_1 *before* I_4, I_2 *equals* I_4, I_3 *equals* I_4
$\mapsto$ Datenübernahme_aus_ALFI(I_5),
I_1 *before* I_5, I_2 *equals* I_5, I_3 *equals* I_5, I_4 *equals* I_5

Fortschreibung_Reserve(I_1),
Fortschreibung_Überschuss_vorschüssig(I_2),
I_1 *before* I_2,
Fortschreibung_Überschuss_vorschüssig(I_3),
I_1 *before* I_3, I_2 *equals* I_3,
Fortschreibung_Reserve(I_4),
I_1 *before* I_4, I_2 *equals* I_4, I_3 *equals* I_4
$\mapsto$ Fortschreibung_Reserve(I_5),
I_1 *before* I_5, I_2 *equals* I_5, I_3 *equals* I_5, I_4 *equals* I_5,
Fortschreibung_Reserve(I_6),
I_1 *before* I_6, I_2 *equals* I_6, I_3 *equals* I_6, I_4 *equals* I_6, I_5 *equals* I_6,
Fortschreibung_Reserve(I_7),
I_1 *before* I_7, I_2 *equals* I_7, I_3 *equals* I_7, I_4 *equals* I_7, I_5 *equals* I_7,
I_6 *equals* I_7,
Fortschreibung_Reserve(I_8),
I_1 *before* I_8, I_2 *equals* I_8, I_3 *equals* I_8, I_4 *equals* I_8, I_5 *equals* I_8,
I_6 *equals* I_8, I_7 *equals* I_8

Die gefundenen Regeln bestehen im Wesentlichen aus verschiedenen Kombinationen der häufigen Änderungsarten, wobei die Regeln aus den Versuchen mit einer Minimalhäufigkeit von 10% kürzer und einfacher sind als die aus den Versuchen mit einer Minimalhäufigkeit von 1%. Weiter wurden keine Regeln gefunden, welche die für die Versicherungsgesellschaft interessanten Änderungen „Prämienreduktion" oder „Prämienfreistellung" enthalten oder voraussagen. Beide Änderungen deuten an, dass der Partner finanziell schlecht gestellt ist und die Prämien nicht mehr oder nur in geringerer Höhe zahlen kann. Diese Änderungen sind deshalb so interessant, weil sie auf einen Rückkauf des Versicherungsvertrags hindeuten könnten, ein Ereignis, das für die Versicherungsgesellschaft einen finanziellen Verlust darstellt. Detailiertere Analysen bzgl. der zurückgekauften Versicherungsverträge werden in Abschnitt 5.3.7 vorgestellt.

Die vom *J-measure* am höchsten gewerteten KAV-Regeln besagen zum Beispiel, dass nach einem „Neueintritt" das „Nachführen der aktuellen Prämie aus dem EVBS-System" – einem EDV-System der Swiss Life – erfolgt. Weitere Regeln besagen, dass nach einer „Antragsannahme" eine „kalkulierte Prämien-/Leistungsanpassung" oder eine „sonstige planmäßige Änderung" erfolgt. All diese Regeln haben eine Minimalhäufigkeit von 10% und werden auch bei einer Minimalhäufigkeit von nur 1% sehr hoch bewertet. Natürlich werden bei dieser niedrigen Minimalhäufigkeit auch andere, nicht so häufige Regeln hoch bewertet, die aber nur aus der Änderungsart „Domino: Erhöhung Sparteil, Reduktion Risikoteil" bestehen.

Ein ähnliches Bild wie die KAV-Regeln bieten die REV- und EUV-Regeln. Auch hier bestehen die gefundenen Regeln aus einigen wenigen, häufigen Änderungsgründen wie zum Beispiel „Neueintritt", „Richtigstellung von Daten in der Datenbank" oder „Fortschreibung Überschuss vorschüssig". Diese Änderungen treten in jeweils unterschiedlichen Kombinationen auf, und die am höchsten bewerteten Regeln mit einer Minimalhäufigkeit von 10% unterscheiden sich größtenteils nicht von den am höchsten bewerteten Regeln mit einer Minimalhäufigkeit von 1%.

In den KRV-Daten wurden, im Vergleich zu den restlichen Daten, sehr wenig Regeln gefunden. Bei einer Minimalhäufigkeit von 10% wurden nur zwei Regeln, bei einer Minimalhäufigkeit von 1% 30 Regeln gelernt. Das Problem bei den KRV-Daten ist, dass die Tarifkomponenten einfach zu wenige Änderungen aufweisen, als dass mehr oder interessante Regeln gefunden werden könnten.

Das gegenteilige Problem tritt bei den FGV-Daten auf. Hier werden zuviele Regeln gefunden, bei einer Minimalhäufigkeit von 1% und einer Minimalkonfidenz von 0,0 z. B. über 1,6 Mio. (vgl. Abbildung 5.4). Wie die aus den anderen Daten gelernten Regeln bestehen auch die FGV-Regeln aus den häufigen Änderungsgründen, allerdings stellen die Regeln keine zeitlichen Zusammenhänge dar, weil die Intervalle meist zeitgleich sind. Einzig das erste Intervall endet bei einigen Regeln vor den restlichen Intervallen. Abbildung 5.7 zeigt das Schema der gefundenen Regeln.

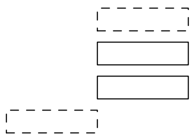


Abbildung 5.7: Schema der in den FGV-Tarifkomponenten gefundenen Regeln

Eine Erklärung für die Form der FGV-Regeln ist die kurze Zeit, die die Fondsgebundenen Lebensversicherungen erst angeboten werden, in Kombination mit der hohen Anzahl an Änderungen. Die Fondsgebundenen Lebensversicherungen laufen erst seit 1998, und bei durchschnittlich 14 Änderungen pro Tarifkomponente, verteilt auf wenige häufige Änderungsgründe, ist es klar, dass keine zeitlichen Zusammenhänge gefunden werden können. Wie bei den Krankenversicherungen ist es deshalb bei den Fondsgebundenen Lebensversicherungen fraglich, ob interessante Regeln gefunden werden können, d. h. ob Data Mining in diesen Daten sinnvoll ist.

In allen analysierten Regelmengen treten zwei Phänomene auf, die Eigenschaften des Algorithmus bzw. der verwendeten Hypothesensprache widerspiegeln und deshalb auch bei der Analyse anderer Daten vorkommen können.

Das erste Phänomen sind sogenannte *Regelhaufen*, vgl. Abbildung 5.8. Dieses

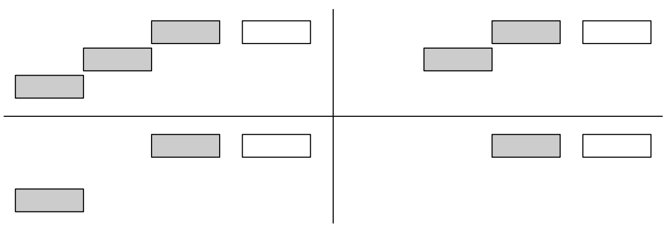


Abbildung 5.8: Skizze eines Regelhaufens

Phänomen ist dadurch zu erklären, dass bei einer niedrig gewählten Minimalhäufigkeit immer längere Muster die geforderte Minimalhäufigkeit erreichen können. Aufgrund des APRIORI-Kriteriums (s. Abschnitt 4.4.2) sind auch alle Teilmuster eines häufigen langen Musters häufig, und bei der Regelerzeugung kann es deshalb passieren, dass aus einem häufigen langen Muster und seinen Teilmustern Regeln generiert werden, die alle sehr hoch bewertet werden. In den hoch gewerteten Regeln treten dann die in Abbildung 5.8 skizzierten Ähnlichkeiten zu Tage.

Aus algorithmischer Sicht gibt es kaum eine Möglichkeit, das Auftreten von Regelhaufen zu verhindern. Eine Lösung wäre, eine höhere Minimalhäufigkeit zu wählen, wobei man aber wieder Gefahr läuft, interessante Regeln zu übersehen, weil sie nicht die geforderte Minimalhäufigkeit erreichen. Eine andere Möglichkeit ist, die Regelmengen nachzubearbeiten. Dabei könnte nach Regelhaufen gesucht werden, und von allen Regeln eines Regelhaufens wird dann nur eine beibehalten, zum Beispiel die längste Regel.

Das andere Phänomen ist, dass sich manche Regeln nur in einer Intervallrelation unterscheiden, wie die Skizze in Abbildung 5.9 verdeutlicht. Dort sind zwei Regeln dargestellt, die sich nur in der Relation der ersten beiden Intervalle unterscheiden. In der ersten Regel ist dies die Relation *meets*, in der zweiten die Relation *before*.

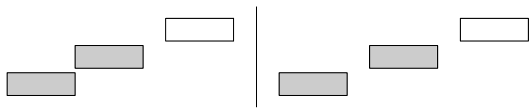


Abbildung 5.9: Ähnliche Regeln, die sich nur in einer Intervallrelation unterscheiden

In den näher analysierten Regeln gibt es einige, die sich nur in einer Intervallrelation unterscheiden. Wenn solche ähnlichen Regeln auftreten könnte das darauf hindeuten, dass die Unterscheidung zwischen diesen beiden Intervallrelationen gar nicht so wichtig ist und gelockert werden könnte, d. h. diese Intervallrelation kann als Disjunktion verschiedener Relationen aufgefasst werden. Die Vermutung bei der disjunktiven Kombination von Intervallrelationen ist, dass Regeln mit gelockerten Intervallrelationen eine größere Häufigkeit haben als solche mit „harten“ Intervallrelationen und somit die wirklichen Regelmäßigkeiten in den Daten besser erfassen können. Höppner greift diesen Punkt in [37] auf und schlägt vor, diese Kombination nach der Berechnung der häufigen Sequenzen und vor der Regelberechnung durchzuführen. Ein Nachteil dieses Ansatzes ist, dass es passieren kann, dass die einzelnen Sequenzen mit den „harten“ Intervallrelationen die geforderte Minimalhäufigkeit nicht erreichen, eine Sequenz, bei der die Intervallrelation aber gelockert ist, schon. Diese würde aber auf Grund der geringen Häufigkeit der Sequenzen mit den „harten“ Intervallrelationen nicht gefunden. Um das zu vermeiden könnte man die Minimalhäufigkeit senken, was aber zur Folge hätte, dass viel mehr Regeln gefunden werden.

Diese Nachteile umgeht der im Folgenden gewählte alternative Ansatz, indem schon bei der Berechnung der häufigen Sequenzen eine erweiterte Hypothesensprache verwendet wird, vgl. Abschnitt 4.4.4.

5.3.4 Erweiterung der Hypothesensprache

Um zu überprüfen, ob mit der erweiterten Hypothesensprache die wirklichen Regelmäßigkeiten in den Daten besser gefunden werden können, wurde für die folgende Analyse die Hypothesensprache um die Intervallrelation *older-&-survived-by* ergänzt, die der Disjunktion (*before* $\vee$ *meets* $\vee$ *overlaps*) entspricht, vgl. Abbildung 5.10. Diese Relation wurde ausgewählt, da gerade die Intervallrelationen *before* und *meets* häufig in ähnlichen Regeln gegeneinander ausgetauscht wurden.

Trotz dieser vorsichtigen Erweiterung der Hypothesensprache um nur eine weitere Intervallrelation, wurden selbst bei einer Minimalhäufigkeit von nur 10% soviel mehr häufige Sequenzen gefunden, dass eine Analyse der FGV-Daten auf Grund von Speicher- und Rechenzeiteinschränkungen nicht möglich war. Die restlichen Tarifkomponentendaten wurden mit der erweiterten Hypothesensprache und mit einer Minimalhäufigkeit von 10% und 1% und einer Minimalkonfidenz von 0,0 analysiert.

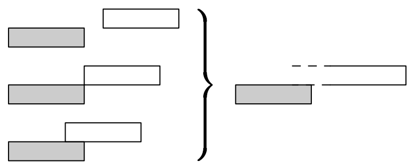


Abbildung 5.10: Intervallrelation der erweiterten Hypothesensprache

Leider brachte die Verwendung der erweiterten Hypothesensprache nicht den gewünschten Erfolg. Es wurden zwar Regeln mit der neuen Intervallrelation gelernt, die auch eine geringfügig höhere Häufigkeit hatten als ähnliche Regeln mit den „harten" Intervallrelationen. Allerdings ist es nicht so, dass die Regeln mit der erweiterten Hypothesensprache die Regelmäßigkeiten in den Daten besser erfassen können, zumindest nicht die in den hier analysierten Daten. Vielmehr tritt durch die Erweiterung der Hypothesensprache ein weiteres Problem auf: Erreicht ein Muster mit einer der Intervallrelationen *before*, *meets* oder *overlaps* die geforderte Minimalhäufigkeit, so auch ein Muster, bei dem diese Intervallrelation zu *older-&-survived-by* verallgemeinert sind. Hat ein häufiges Muster n Intervallrelationen *before*, *meets* oder *overlaps*, dann gibt es weitere $2^n - 1$ häufige Muster, bei denen eine oder mehrere dieser Intervallrelationen zu *older-&-survived-by* verallgemeinert ist. In den Regeln tritt somit eine weitere Form von Häufungen auf, bei denen es sich um Regeln handelt, die alle dieselbe Intervallkombination haben, die sich aber nur in den Intervallrelationen teilweise unterscheiden, und zwar an den Stellen, an denen die Intervallrelationen *before*, *meets*, *overlaps* oder *older-&-survived-by* auftreten.

Die erweiterte Hypothesensprache ist somit nicht geeignet, die wirklichen Regelmäßigkeiten in den hier analysierten Daten besser zu erfassen. Deshalb wird für die folgenden Untersuchungen wieder die alte Hypothesensprache verwendet. Der alternative Ansatz, die erweiterte Hypothesensprache erst bei der Regelberechnung zu verwenden, wird aus Zeitgründen nicht ausprobiert. Die im vorherigen Absatz beschriebene Häufung von ähnlichen Regeln würde aber auch beim alternativen Ansatz auftreten, wenn auch nicht in dem Maße wie bei der hier angewandten Vorgehensweise.

5.3.5 Alternative Regelbewertungskriterien

Da allein die Menge der gefundenen Regeln (vgl. Abbildung 5.4) eine detaillierte Analyse aller Ergebnisse unmöglich macht, müssen die Regeln nachbearbeitet werden, um hoffentlich interessante Zusammenhänge zu finden. Bisher wurde dazu das *J-measure* [66, 39] verwendet, das einen guten Kompromiss zwischen der Einfachheit und somit der Verständlichkeit einer Regel und ihrer Allgemeingültigkeit, d. h. ihrer Fähigkeit, die Regelmäßigkeiten in den Daten zu erfassen, darstellt.

Angesichts der Fülle an Regelbewertungskriterien, die in der Literatur zu finden sind ([36] gibt einen Überblick), und der bisherigen Ergebnisse stellt sich die Frage, ob vielleicht ein anderes Bewertungskriterium besser geeignet ist, aus allen Regeln die interessantesten herauszufiltern. Deshalb wurden die in Abschnitt 5.3.3 analysierten Regelmengen nochmals mit zwei weiteren Regelbewertungskriterien, dem *interest* und den *certainty factors* (vgl. Abschnitt 4.4.3), bewertet und die 20 besten Regeln analysiert.

Allerdings zeigt sich, dass auch die beiden alternativen Bewertungskriterien nicht geeignet sind, interessantere Regeln in den Regelmengen zu finden. Auf Grund der wenigen Regeln in den Regelmengen mit einer Minimalhäufigkeit von 10% unterscheiden sich die mit den alternativen Bewertungskriterien gewerteten Regeln nur leicht in der Reihenfolge, in der sie angeordnet sind. Deutlichere Unterschiede in den am höchsten bewerteten Regeln treten dagegen in den Regelmengen mit einer Minimalhäufigkeit von 1% auf. Hier werden sowohl vom *interest* als auch von den *certainty factors* relativ komplexe Regeln mit einer geringen Häufigkeit sehr hoch bewertet. Dabei bevorzugt der *interest* Regeln mit in etwa gleichlanger Prämisse und Konklusion, während die *certainty factors* Regeln mit langer Prämisse und einelementiger Konklusion hoch bewerten. Beim *J-measure* hingegen unterscheiden sich die am höchsten gewerteten Regeln zwischen den Regelmengen mit einer Minimalhäufigkeit von 10% und 1% kaum, wie schon in Abschnitt 5.3.3 erläutert.

Ein Blick auf die Formeln für die beiden alternativen Bewertungskriterien lässt erkennen, warum diese seltene, komplexe Regeln so hoch bewerten:

$$\begin{aligned}
\mathrm{interest}(P \to C) &:= \frac{\mathrm{supp}(P \wedge C)}{\mathrm{supp}(P)\mathrm{supp}(C)} \\
\mathrm{cf}(P \to C) &:= \begin{cases} \frac{\mathrm{conf}(P\to C)-\mathrm{supp}(C)}{1-\mathrm{supp}(C)} & \text{falls } \mathrm{conf}(P \to C) > \mathrm{supp}(C) \\ \frac{\mathrm{conf}(P\to C)-\mathrm{supp}(C)}{\mathrm{supp}(C)} & \text{falls } \mathrm{conf}(P \to C) < \mathrm{supp}(C) \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Werden Regeln komplexer, so sinkt ihre Häufigkeit und somit auch die Häufigkeit ihrer Prämisse und ihrer Konklusion. Angenommen, eine Regel hat eine Häufigkeit von 10%, wobei Prämisse und Konklusion jeweils eine Häufigkeit von 15% haben, so hat diese Regel einen *interest* von $\frac{0{,}1}{0{,}15 \cdot 0{,}15} = 4{,}\overline{4}$. Hat eine komplexere, längere Regel eine Häufigkeit von nur 5%, wobei ihre Prämisse und und Konklusion eine Häufigkeit von 7,5% haben, dann hat diese Regel einen *interest* von $\frac{0{,}05}{0{,}075 \cdot 0{,}075} = 8{,}\overline{8}$. Auch wenn man es analytisch wegen der gegenseitigen Beeinflussung der Häufigkeit einer ganzen Regel und der ihrer Prämisse bzw. Konklusion schlecht beweisen kann, so zeigt dieses Beispiel doch, warum der *interest* komplexe und somit seltene Regeln mit in etwa gleichlanger Prämisse und Konklusion bevorzugt.

Ähnlich verhält es sich bei den *certainty factors*. Diese bewerten Regeln mit einer hohen Konfidenz und einer niedrigen Häufigkeit der Konklusion sehr hoch. Eine hohe Konfidenz erreichen vor allem sehr lange und komplexe Regeln mit einer kurzen Konklusion. Deshalb bewerten die *certainty factors* Regeln mit langer Prämisse und einelementiger Konklusion sehr hoch.

Zusammenfassend lässt sich sagen, dass auch andere Regelbewertungskriterien zumindest auf den vorliegenden Daten nicht in der Lage sind, interessantere Regeln aus der Menge aller gefundenen Regeln besser herauszufiltern als das *J-measure*. Das liegt zum einen wahrscheinlich daran, dass bisher keine wirklich interesanten Regeln aus den Daten gelernt wurden. Zum anderen liegt es an den oben beschriebenen Nachteilen der beiden alternativen Regelbewertungskriterien. Deshalb wird für die nachfolgenden Untersuchungen wieder das *J-measure* zur Bewertung der Regeln verwendet.

5.3.6 Veränderung der Regeln mit der Zeit

Ein Nachteil sowohl der normalen als auch der erweiterten Hypothesensprache ist, dass nur die qualitativen Zusammenhänge der Zeitintervalle erfasst werden. Der Zeitpunkt an dem ein Muster in einer Eingabesequenz auftritt oder seine zeitliche Ausdehnung werden nicht berücksichtigt. Für viele Fragestellungen ist es aber gerade interessant zu untersuchen, ob ein Muster eher am Anfang der Eingabesequenzen auftritt oder eher am Ende, da in den Sequenzen in verschiedenen Zeitabschnitten durchaus unterschiedliche Muster auftreten können. Ebenso können Muster, die eine kurze zeitliche Ausdehnung besitzen, interessanter sein als solche mit einer langen Zeitdauer.

Um die quantitativen Aspekte der Zeit zu berücksichtigen kann man die Hypothesensprache um entsprechende Formalismen erweitern, ähnlich wie es in [42] für sequentielle Muster gemacht wird. Allerdings ist unklar, wie die entsprechenden Erweiterungen der Hypothesensprache aussehen könnten und welche Erweiterungen überhaupt sinnvoll wären und welche nicht. Eine andere, wenn auch nicht ganz so weitreichende Möglichkeit ist, die Daten entsprechend vorzuverarbeiten.

Im Folgenden soll an Hand der Tarifkomponentendaten gezeigt werden, wie durch verschiedene Arten der Filterung bezüglich der Zeit nach Veränderungen der in den Daten gefundenen Regeln gesucht werden kann. Hierbei spielen zwei Formen zeitlicher Information eine Rolle, die *Weltzeit* und die *Laufzeit*. Die Weltzeit ist die kalendarische Zeit. Werden die Daten bezüglich der Weltzeit gefiltert, so kann zum Beispiel untersucht werden, ob irgendwelche äußeren Einflüsse die in den Daten vorhandenen Muster beeinflusst und verändert haben. Die Laufzeit hingegen misst die Zeit relativ zum Anfangszeitpunkt einer Eingabesequenz. Werden die Daten bezüglich der Laufzeit gefiltert, so kann nach Mustern gesucht werden, die zum Beispiel nur am Anfang oder nur am Ende einer Eingabesequenz auftreten.

Filterung bezüglich der Weltzeit

Indem die Tarifkomponentendaten bzgl. der Weltzeit gefiltert werden, sollen zwei Fragestellungen untersucht werden. Die erste Fragestellung ist, ob sich die Änderungen der Tarifkomponenten mit der Zeit selber verändert haben. Die ältesten Tarifkomponenten stammen aus dem Jahr 1946, die jüngsten aus 2001 und Anfang 2002. Es könnte durchaus sein, dass sich gesellschaftliche, politische oder wirtschaftliche Veränderungen auf die Änderungsmuster ausgewirkt haben. Ein Indiz hierfür liefern die Histogramme in Abbildung 5.11. Sie zeigen die Anzahl an Änderungen der Tarifkomponenten pro Jahr, zum einen absolut und zum anderen als Durchschnitt pro laufender Tarifkomponente. Die durchschnittliche Anzahl an Änderungen pro laufender Tarifkomponente stellt dabei die interessantere Information dar, da die Zahl der Tarifkomponenten und somit die absolute Anzahl an Änderungen mit den Jahren gestiegen ist[3]. Abbildung 5.11 zeigt deutlich, dass die Tarifkomponenten der Kapitallebensversicherungen, der Renten- und der Krankenversicherungen die meiste Zeit kaum verändert worden sind. Die durchschnittliche Anzahl an Änderungen liegt immer unter 0,5, und erst ab Mitte der 90er Jahre des letzten Jahrhunderts steigt sie bei den KAV- und REV-Tarifkomponenten auf über 1,0 bzw. 1,5 an, während die KRV-Tarifkomponenten nur einen leichten Anstieg zeigen. Die Erwerbsunfähigkeitsversicherungen und die Fondsgebundenen Lebensversicherungen werden erst seit 1996 bzw. 1998 angeboten, weshalb fraglich ist, ob in ihnen überhaupt Veränderungen der Muster gefunden werden können.

Um zu untersuchen, ob sich die Änderungen der Tarifkomponenten mit den Jahren verändert haben, wird die Weltzeit in Dekaden unterteilt, d. h. in die Zeiträume 1940–1949, 1950–1959 usw. Die Änderungen der Tarifkomponenten werden dann bezüglich der Dekaden gefiltert und analysiert. Sollen zum Beispiel die Änderungen des Zeitraums 1980–1989 nach häufigen Änderungsmustern durchsucht werden, so werden nur die Änderungen analysiert, die diesen Zeitraum überschneiden. Wie in den vorherigen Untersuchungen werden die gefilterten Daten mit den Minimalhäufigkeiten 10% und 1% sowie einer Minimalkonfidenz von 0,0 analysiert.

Die Ergebnisse der Analysen sind nicht überraschend. Wie man sich an Hand von Abbildung 5.11 leicht erklären kann wurden für den Zeitraum 1940–1979 in den KAV- und KRV-Daten keine Regeln gefunden, da die Tarifkomponenten in diesen Dekaden einfach zu wenige Änderungen aufweisen, als dass häufige Intervallmuster

[3] Der abschließende steile Abfall der Histogramme ist dadurch zu erklären, dass die Daten Anfang 2002 aus dem Data Warehouse der Swiss Life extrahiert wurden, sodass für 2002 nur ganz wenige Änderungen vorliegen.

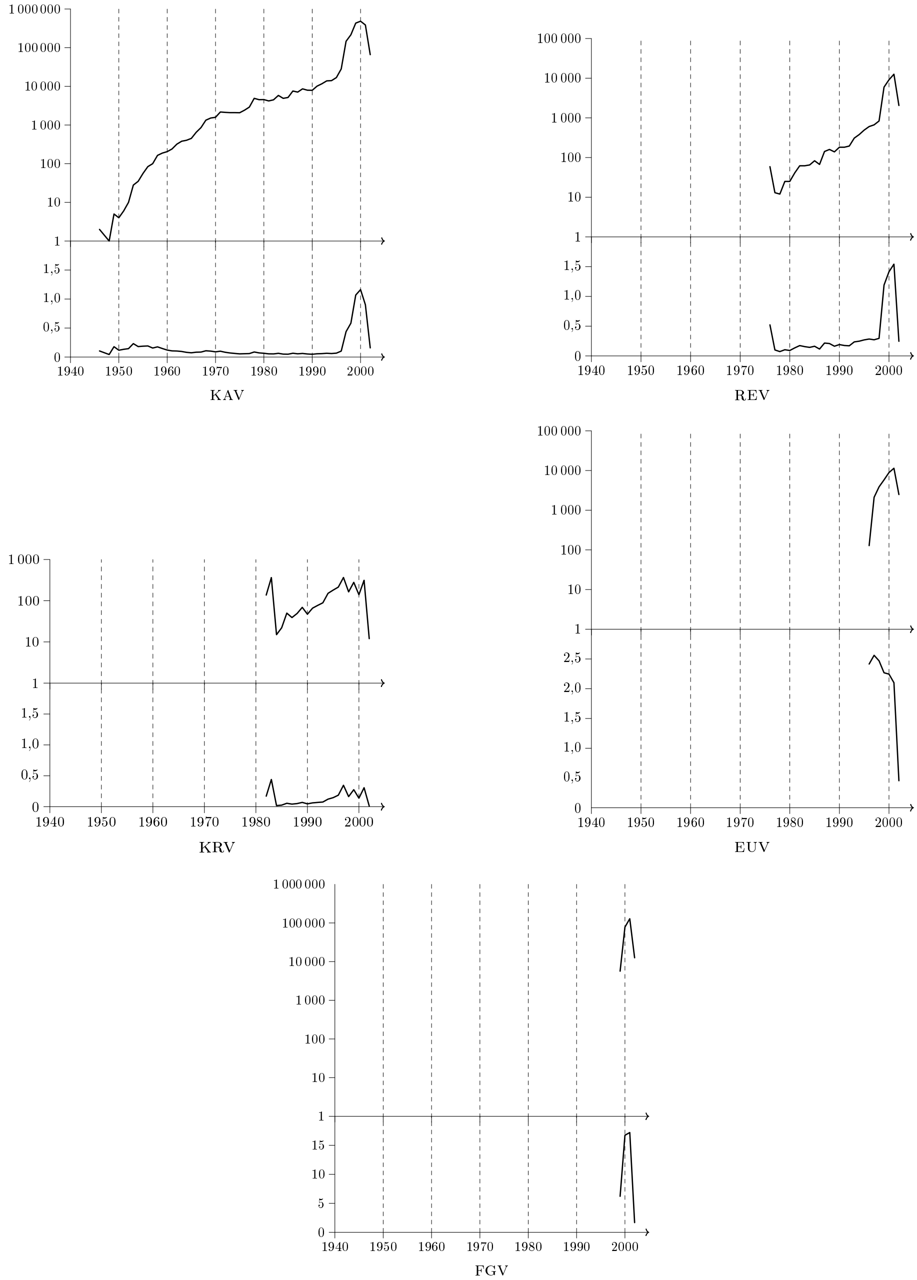


Abbildung 5.11: Anzahl an Änderungen pro Jahr, absolut und durchschnittlich pro laufender Tarifkomponente

mit mehr als einem Intervall gefunden werden könnten. Einzig in den REV-Daten wurden für den Zeitraum 1970–1979 einige Regeln gefunden, die aber den Regeln aus den nichtgefilterten REV-Daten sehr stark ähneln. Für die Zeiträume 1980–1989, 1990–1999 und 2000–2009 wurden in den KAV-, den REV- und den KRV-Daten zwar Regeln gefunden, die aber größtenteils den Regeln aus den jeweiligen nichtgefilterten Daten gleichen. Vereinzelt gibt es zwar Regeln, die Änderungsgründe enthalten, die in den in Abschnitt 5.3.3 vorgestellten Regeln nicht vorkommen. Diese Regeln liefern aber keine interessanten Informationen.

In den gefilterten EUV- und FGV-Daten wurden nur für die Zeiträume 1990–1999 und 2000–2009 Regeln gefundenen, die aber ebenfalls den in den nichtgefilterten Daten gefundenen Regeln sehr ähneln.

Zusammenfassend lässt sich sagen, dass sich die Regeln nicht mit der Zeit verändert haben. Da für die Zeit vor 1990 in fast allen Daten keine oder nur sehr wenige Regeln gefunden wurden ist es vermutlich sogar so, dass die in den nichtgefilterten Daten gefundenen Regeln größtenteils die Änderungen ab 1990 beschreiben. Durch mangelnde Dokumentation der Daten lässt es sich nicht belegen, aber eine Vermutung ist, dass die Änderungen der Tarifkomponenten, oder allgemein Änderungen an Versicherungsverträgen, erst seit Mitte der 90er Jahre des letzten Jahrhunderts detailliert aufgezeichnet werden. Vorher wurden wahrscheinlich nur sehr wichtige Änderungen in den Akten vermerkt. Änderungen, die nur normale Geschäftsprozesse widerspiegeln, wurden wahrscheinlich nicht dokumentiert. Insofern ist es natürlich fraglich, ob die Suche nach typischen Änderungsmustern in den vorliegenden Daten überhaupt Sinn macht, da nur Muster gefunden werden können, die Änderungen der letzten 10 Jahre beschreiben.

Die zweite, im Folgenden untersuchte Frage ist, ob der steile Anstieg in der Anzahl der Änderungen Mitte der 90er Jahre vielleicht nicht nur auf den Aufbau des Data Warehouse bei der Swiss Life zurückzuführen ist, sondern vielleicht auch auf die Einführung der sogenannten *Stempelsteuer* im April 1998. Die Stempelsteuer kann auf Wertpapiere, auf Quittungen von Versicherungsprämien und auf andere Urkunden des Handelsverkehrs erhoben werden und hat dazu geführt, dass viele Kunden ihre Versicherungen aus steuerlichen Gründen aufgelöst haben. Selbst wenn die vermehrten Änderungen nicht auf die Stempelsteuer zurückzuführen sind, ist es interessant zu untersuchen, ob die Einführung dieser Steuer die Änderungen der Tarifkomponenten beeinflusst hat.

Da nicht klar ist ab wann sich der Einfluss der Stempelsteuer in den Änderungen der Tarifkomponenten bemerkbar machen könnte, werden die Daten bezüglich verschiedener Schwellwertjahre gefiltert. Dazu werden die Änderungen bzgl. eines festen Schwellwertjahres aus dem Zeitraums 1996–2000 gefiltert, d. h. in die Zeiträume vor und nach diesem Schwellwertjahr eingeteilt, und für jeden Zeitraum werden die gefilterten Daten mit den Minimalhäufigkeiten 10% und 1% sowie einer Minimalkonfidenz von 0,0 analysiert.

Auch mit dieser Form der zeitlichen Filterung wurden keine interessanten Regeln gefunden. Bei der Filterung bezüglich des Schwellwertjahres 1996 wurden die meisten Regeln für den Zeitraum nach 1996 gefunden. Je weiter aber das Schwellwertjahr nach hinten verschoben wurde, um so mehr Regeln „sickerten“ aus dem Zeitraum nach dem Schwellwertjahr in den Zeitraum vor dem Schwellwertjahr durch, was aber nicht verwunderlich ist, da sich die Partitionierung der Daten mit dem Schwellwertjahr verändert. Die gefundenen Regeln ähneln außerdem sehr stark den in den nichtgefilterten Daten gefundenen Regeln, sodass auf eine detailliertere Erläuterung der Ergebnisse verzichtet wird.

Obwohl die Filterung bezüglich der Weltzeit auf den vorliegenden Daten keine neuen, interessanten Informationen erbracht hat, kann man dennoch sagen, dass die hier geschilderte Vorgehensweise dazu geeignet ist, Veränderungen der in den Daten vorhandenen Muster, die durch äußere Einflüsse hervorgerufen wurden, zu finden.

Filterung bezüglich der Laufzeit

Eine weitere Möglichkeit, wie sich die in den Daten vorhandenen Muster und Regeln mit der Zeit verändern können, ist die Veränderung mit der Laufzeit. Es kann durchaus sein, dass am Anfang der Sequenzen andere Muster auftreten als zum Beispiel am Ende.

Um Sequenzen bezüglich der Laufzeit filtern zu können, müssen sie alle auf einen gemeinsamen Nullpunkt umdatiert werden. Im Falle der Tarifkomponenten ist es so, dass jede Änderung einer Tarifkomponente durch zwei Zeitintervalle gekennzeichnet ist. Das Datumspaar TKBEG/TKEND kennzeichnet den Zeitraum, in dem die Tarifkomponente gültig ist, also ihre Laufzeit, während das Datumspaar TKWIVON/TKWIBIS den Gültigkeitszeitraum der Änderung beschreibt. Für jede Tarifkomponente könnte man nun das Minimum der TKBEG-Daten und das Maximum der TKEND-Daten ihrer Änderungen bestimmen und hätte so ihre Laufzeit berechnet. Leider gibt es aber Änderungen, die vor dem Minimum der TKBEG-Daten bzw. nach dem Maximum der TKEND-Daten datiert sind, sodass die Laufzeit einer Tarifkomponente aus dem Minimum der TKBEG- und TKWIVON-Daten und dem Maximum der TKEND- und TKWIBIS-Daten berechnet wird. Alle Änderungsdaten werden dann so umdatiert, dass der Nullpunkt einer Eingabesequenz mit dem Anfangszeitpunkt der Laufzeit dieser Tarifkomponente zusammenfällt.

Die eigentliche Filterung bezüglich der Laufzeit ist relativ simpel. Die Laufzeit der Tarifkomponenten wird in drei gleichlange Abschnitte unterteilt, welche die Änderungen während des Anfangszeitraums der Tarifkomponente, nach einer gewissen Laufzeit und am Ende der Laufzeit der Tarifkomponente erfassen sollen. Bei der Analyse der Änderungen in einem der drei Zeitabschnitte werden nur die Änderungen miteinbezogen, die diesen Zeitraum überschneiden, und wie bisher werden die Daten mit den Minimalhäufigkeiten 10% und 1% sowie einer Minimalkonfidenz von 0,0 analysiert.

Auf Grund der zeitlichen Filterung wurden bei einer Minimalhäufigkeit von 10% sehr wenige Regeln gefunden. In den KAV-Daten wurden im ersten Drittel einige Regeln gefunden, die auf Grund der Änderung „Antrangsannahme“ die Änderungen „kalkulierte Prämien-/Leistungsanpassung“ sowie „sonstige planmäßige Änderung“ vorhersagen. Im zweiten Drittel wurde dagegen nur eine Regel gefunden und im letzten Drittel gar keine. Ein ähnliches Bild bieten die REV-Regeln. Hier wurden im ersten Drittel einige Regeln gefunden, die auf Grund einer „Antragsannahme“ die Änderungen „Fortschreibung Überschuss vorschüssig“ sowie „kalkulierte Prämien-/Leistungsanpassung“ vorhersagen. Die Regeln aus den anderen beiden Dritteln bestehen größtenteils nur aus den Änderungsgründen „Fortschreibung Überschuss vorschüssig“ und „kalkulierte Prämien-/Leistungsanpassung“. In den KRV-Daten wurden einige wenige Regeln gefunden, die eine „Richtigstellung von Daten in der Datenbank“ voraussagen, wobei dies im ersten Drittel auf Grund eines „Neueintritts“ geschieht und in den beiden anderen Drittel wegen einer „Tarifanpassung“. In den EUV- und FGV-Daten wurden auf Grund der sehr kurzen bisherigen Laufzeit der Tarifkomponenten nur im ersten Drittel Regeln gefunden, die deshalb den in den nichtgefilterten Daten gefundenen Regeln entsprechen.

Bei einer Minimalhäufigkeit von 1% wurden in allen Daten mehr Regeln gefunden. In den KAV-Daten wurden in allen Dritteln ähnliche Regeln wie in den nichtgefilterten Daten gefunden, die sich aber teilweise in den vorkommenden Änderungsgründen unterscheiden. Die Änderungen „Antragsannahme“ und „kalkulierte Prämien-/Leistungsanpassung“ treten nur in den Regeln aus dem ersten Drittel auf, während die Regeln aus dem zweiten Drittel größtenteils aus den Änderungen „Domino: Erhöhung Sparteil, Reduktion Risikoteil“, „sonstige planmäßige Änderung“ und „Nachführen aktueller Prämie EVBS“ bestehen. Im letzten Drittel wurden einige Regeln gefunden, die den „Rücktritt“ voraussagen, allerdings nur auf

Grund häufiger Änderungsgründe (siehe auch Abschnitt 5.3.7). Die bei einer Minimalhäufigkeit von 1% in den REV-Daten gefundenen Regeln ähneln denen bei einer Minimalhäufigkeit von 10% gefundenen, wobei die Regeln auf Grund der geringeren Minimalhäufigkeit länger geworden sind. Die Anzahl der in den KRV-Daten gefundenen Regeln nimmt vom ersten bis zum letzten Drittel hin ab, unterscheiden sich aber nicht signifikant von den in den nichtgefilterten Daten gefundenen Regeln. Bei den EUV- und FGV-Daten ist es ebenfalls so, dass die im ersten Drittel gefundenen Regeln den in den nichtgefilterten Daten gefundenen Regeln entsprechen, während in den restlichen Dritteln in den FGV-Daten keine Regeln gefunden wurden, und die in den EUV-Daten gefundenen Regeln nur aus dem Änderungsgrund „sonstige planmäßige Änderung“ bestehen.

Zusammenfassend lässt sich sagen, dass auch mit der Filterung bezüglich der Laufzeit Änderungen in den Mustern gefunden werden können. Auch wenn in den vorliegenden Daten keine signifikanten Unterschiede in den Mustern gefunden wurden, so zeigt zum Beispiel die Tatsache, dass Regeln mit der Änderung „Neueintritt“ in der Prämisse nur im ersten Drittel gefunden wurden, dass diese Form der Filterung bezüglich der Zeit funktioniert.

5.3.7 Vorhersage von Rückkauf

Der letzte Punkt der bei den Tarifkomponentendaten untersucht werden soll, ist die Vorhersage von Rückkauf. Wie schon mehrmals erwähnt versteht man unter Rückkauf die vorzeitige Beendigung einer kapitalbildenden Lebensversicherung wegen Rücktritt oder Kündigung [26], wobei dem Versicherungsnehmer der aktuelle Zeitwert der Versicherung auszuzahlen ist. Für die Versicherungsgesellschaft sind zurückgekaufte Versicherungen ein Verlustgeschäft, da kurzfristig Kapital beschafft werden muss bzw. entsprechende Finanzmittel nicht längerfristig und damit gewinnbringend angelegt werden können. Wenn es nun gelingt, den Rückkauf auf Grund der vorherigen Änderungen einer Tarifkomponente vorherzusagen, könnte die Versicherungsgesellschaft versuchen, den wahrscheinlichen Rückkauf zu verhindern, indem die betroffenen Versicherungsnehmer angesprochen und ihnen entsprechende Angebote unterbreitet werden. Allerdings ist fraglich, ob es sinnvoll ist zu versuchen, den Rückkauf auf Grund der Änderungen *einer* Tarifkomponente vorherzusagen, da der Rückkauf immer einen ganzen Versicherungsvertrag betrifft. Darum wird die Vorhersage des Rückkaufs in Abschnitt 5.4.2 nochmals auf Versicherungsvertragsebene untersucht.

Um den Rückkauf vorherzusagen, wurden die Tarifkomponenten der einzelnen Versicherungsarten[4] nochmals in zwei Gruppen unterteilt, die zurückgekauften und die nicht zurückgekauften Tarifkomponenten. Diese Unterteilung war nicht ganz trivial, da einige zurückgekaufte Tarifkomponenten wieder reaktiviert wurden, d. h. der Rückkauf selber wurde rückgängig gemacht. Von diesen reaktivierten Tarifkomponenten wurden dann schließlich doch einige zurückgekauft. Nach der Partitionierung der Tarifkomponentendaten wurden die verschiedenen Datenmengen mit den Minimalhäufigkeiten 10% und 1% sowie einer Minimalkonfidenz von 0,0 analysiert.

Wie schon in den vorherigen Analysen liefern auch die Ergebnisse dieser Untersuchung keine neuen Erkenntnisse. Die meisten der in den zurückgekauften Tarifkomponenten gefundenen Regeln sagen Rückkauf auf Grund von häufigen Änderungen voraus, die ebenfalls in der Prämisse vieler Regeln vorkommen, die in den Daten der nicht zurückgekauften Tarifkomponenten gefunden wurden. Somit ist es nicht möglich, allein auf Grund der Änderungen einer Tarifkomponente ihren Rückkauf vorherzusagen.

[4] Bis auf die Krankenversicherungen, hier gab es keine Rückkäufe.

5.3.8 Fazit der ersten Analysen

Obwohl die Ergebnisse der bisherigen Untersuchungen keine neuen Informationen erbracht haben wurde dennoch gezeigt, wie Intervallsequenzen mit Hilfe des im Rahmen dieser Arbeit implementierten Algorithmus auf häufige Muster und interessante Regeln hin untersucht werden können. Was bei der Vorverarbeitung der Daten berücksichtigt werden muss, wie die Parameter des Algorithmus gewählt werden müssen und wie die interessantesten Regeln in der Menge aller gelernten Regeln gefunden werden können, wurde erläutert, und mögliche Alternativen wurden aufgezeigt. Des weiteren wurde untersucht, ob eine Erweiterung der Hypothesensprache besser in der Lage ist, in den Daten vorhandene Muster zu finden, und wie sich durch Vorverarbeitung der Daten zeitliche Veränderungen der Muster selber finden lassen.

Im nächsten Abschnitt werden noch weitere Analysen erläutert, welche den gesamten Versicherungsvertrag auf zeitliche Änderungsmuster hin analysieren. Neben der komplizierteren Vorverarbeitung bereitete die große Menge an Daten Probleme bei der Analyse. Wie diese Probleme gelöst wurden und welche Ergebnisse erzielt wurden, schildert der nächste Abschnitt.

5.4 Analyse der Versicherungsverträge

Die Analyse der Änderungsmuster der Tarifkomponenten hat zwar einige Erkenntnisse darüber geliefert, wie der Data Mining-Algorithmus angewendet werden kann, neues Wissen über die zeitliche Entwicklung der Tarifkomponenten wurde aber nicht gefunden. Deshalb sollen in einigen weiteren Untersuchungen die Versicherungsverträge nach interessanten zeitlichen Änderungsmustern durchsucht werden. Dazu muss zuerst die Vorverarbeitung angepasst werden, da ein Versicherungsvertrag aus mehreren Tarifkomponenten besteht und eine Änderung einer oder mehrerer Tarifkomponenten und/oder dem Vertrag zugeordnet sein kann (vgl. Abschnitt 5.2.2).

Im Folgenden definiert ein Versicherungsvertrag eine Eingabesequenz, und jede Änderung stellt wieder ein Intervall der Eingabesequenz dar. Die Intervallmarkierungen werden aber diesmal so definiert, dass die Änderung eindeutig einer Tarifkomponente und/oder dem gesamten Vertrag zugeordnet werden kann. Nur so ist es möglich, das „Zusammenspiel“ der Änderungen zu erfassen.

Durch die Kombination der Änderungen der Tarifkomponenten und Versicherungsverträge werden die Eingabesequenzen natürlich länger als in den bisherigen Untersuchungen. Tabelle 5.3 zeigt die durchschnittliche Länge der Sequenzen, aufgeteilt nach der Versicherungsart.

KAV	REV	KRV	EUV	FGV
15,90	8,31	7,17	15,02	52,17

Tabelle 5.3: Durchschnittliche Anzahl an Änderungen der Versicherungsverträge

Die Komplexität der Eingabesequenzen bereitete auch Probleme bei der Durchführung der Analysen. Der nichtoptimierte Algorithmus war nicht verwendbar, da so viele Kandidatenmuster erzeugt wurden, dass die Berechnung der Häufigkeit nicht mehr in annehmbarer Zeit durchführbar war. Anfängliche Data Mining-Experimente wurden nach zwei Tagen abgebrochen, da der Algorithmus immer noch mit der Häufigkeitsberechnung beschäftigt war. Der optimierte Algorithmus hingegen benötigte zur Analyse *aller* Eingabesequenzen zuviel Speicherplatz. Selbst als dem Programm 1 600 MB Speicher zur Verfügung gestellt wurden, konnten zum Beispiel bei niedrigen Minimalhäufigkeiten unter 30%, die erst interessante Einsichten in die Daten ermöglichen, nicht alle KAV-Daten analysiert werden. Deshalb wurde nur noch eine Stichprobe von 1 500 Versicherungsverträgen zum Lernen verwendet. Dennoch konnten selbst diese relativ kleinen Stichproben nur mit Minimalhäufigkeiten von 10% und 5% analysiert werden, da bei niedrigeren Minimalhäufigkeiten der Speicherplatz nicht ausreichte. Ansonsten wurden wie bisher alle gefundenen Regeln mit Hilfe des *J-measures* bewertet und die 20 am höchsten gewerteten Regeln näher analysiert. Die Daten der Fondsgebundenen Lebensversicherungen wurden allerdings nicht analysiert. Sie sind zum einen zu komplex, sodass selbst die Stichprobe von nur 1 500 Versicherungsverträgen bei 1 600 MB Speicherplatz und einer Minimalhäufigkeit von 10% nicht mit dem optimierten Algorithmus analysiert werden konnte. Der nichtoptimierte Algorithmus wurde bei den gleichen Parametereinstellungen nach einer Laufzeit von einem Tag abgebrochen, da er schon ca. 3 Mio. 4-Kandidatenmuster überprüfen musste und die Anzahl der 5-Muster noch größer gewesen wäre. Neben der Komplexität sprechen die Ergebnisse der Analyse der FGV-Tarifkomponenten (vgl. Abschnitt 5.3.3) gegen eine Untersuchung der FGV-Daten. Die bei der Analyse der FGV-Tarifkomponenten gefundenen Regeln stellten keine zeitlichen Zusammenhänge dar, weil die Intervalle meist zeitgleich waren. Einzig das erste Intervall endete bei einigen Regeln vor den restlichen Intervallen, vgl. Abbildung 5.7. Da viele Änderungen einer Tarifkomponente durch Änderungen des gesamten Versicherungsvertrags hervorgerufen werden, ist es wahrscheinlich, dass die bei der Analyse der FGV-Versicherungen gefundenen Regeln denen in Abbildung 5.7 sehr ähnlich wären. Weiterhin wurde schon mehrmals erwähnt, dass die FGV-Versicherungen erst seit 1998 angeboten werden, weshalb es fraglich ist, ob in ihnen überhaupt zeitliche Änderungsmuster gefunden werden können.

5.4.1 Typische Änderungsmuster in Versicherungsverträgen

Wie bei den Tarifkomponenten (vgl. Abschnitt 5.3) besteht auch bei der Analyse der gesamten Änderungen eines Versicherungsvertrags der erste Schritt darin, nach typischen Änderungsmustern zu suchen. Leider wurden auch hier keine wirklich überraschenden oder interessanten Regelmäßigkeiten gefunden, weder in den Regeln mit einer Minimalhäufigkeit von 10% noch in denen mit einer Minimalhäufigkeit von 5%. Tabelle 5.4 zeigt jeweils fünf der 20 am höchsten gewerteten Regeln aus den Untersuchungen der verschiedenen Versicherungsarten mit einer Minimalhäufigkeit von 5%. Die Intervalle, die Änderungen einzelner Tarifkomponenten zugeordnet sind, tragen die Endung „... _TK<*Tarifkomponenten-Nr.*>“, wohingegen Intervalle, die dem gesamten Vertrag zugeordnet sind, keine besondere Endung besitzen.

Tabelle 5.4: Fünf Beispielregeln pro Versicherungsart aus der kombinierten Analyse der Versicherungsvertrags- und Tarifkomponentenänderungen

KAV	Nachführen_aktuelle_Prämie_EVBS(I_1) $\mapsto$ Nachführen_aktuelle_Prämie_EVBS_TK1(I_2), I_1 *equals* I_2
	Neueintritt(I_1) $\mapsto$ Neueintritt_TK1(I_2), I_1 *equals* I_2

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

	Nachführen_aktuelle_Prämie_EVBS(I_1), Nachführen_aktuelle_Prämie_EVBS_TK1(I_2), I_1 *equals* I_2, Nachführen_aktuelle_Prämie_EVBS(I_3), I_1 *meets* I_3, I_2 *meets* I_3 $\mapsto$ Nachführen_aktuelle_Prämie_EVBS_TK1(I_4), I_1 *meets* I_4, I_2 *meets* I_4, I_3 *equals* I_4
	Antragsannahme_TK1(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *before* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *starts* I_3
	Antragsannahme(I_1), Antragsannahme_TK1(I_2), I_1 *starts* I_2, sonstige_planmäßige_Änderung(I_3), I_1 *before* I_3, I_2 *meets* I_3 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_4), I_1 *before* I_4, I_2 *meets* I_4, I_3 *starts* I_4
REV	Richtigstellung_von_Daten_in_DB(I_1) $\mapsto$ Richtigstellung_von_Daten_in_DB_TK1(I_2), I_1 *equals* I_2
	Antragsannahme_TK1(I_1), kalkulierte_Prämien-/Leistungsanpassung(I_2), I_1 *before* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig(I_3), I_1 *before* I_3, I_2 *starts* I_3
	Fortschreibung_Überschuss_vorschüssig_TK1(I_1), Fortschreibung_Überschuss_vorschüssig(I_2), I_1 *meets* I_2 $\mapsto$ Fortschreibung_Überschuss_vorschüssig_TK1(I_3), I_1 *meets* I_3, I_2 *equals* I_3
	Neueintritt_TK1(I_1), Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2 $\mapsto$ Richtigstellung_von_Daten_in_DB_TK1(I_3), I_1 *starts* I_3, I_2 *equals* I_3
	Neueintritt(I_1), Neueintritt_TK1(I_2), I_1 *equals* I_2, Richtigstellung_von_Daten_in_DB(I_3), I_1 *starts* I_3, I_2 *starts* I_3 $\mapsto$ Richtigstellung_von_Daten_in_DB_TK1(I_4), I_1 *starts* I_4, I_2 *starts* I_4, I_3 *equals* I_4

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

KRV	Tarifanpassung(I_1), Tarifanpassung_TK1(I_2), I_1 *equals* I_2 $\mapsto$ Tarifanpassung_TK2(I_3), I_1 *equals* I_3, I_2 *equals* I_3
	Neueintritt(I_1), Neueintritt_TK1(I_2), I_1 *equals* I_2 $\mapsto$ Neueintritt_TK2(I_3), I_1 *equals* I_3, I_2 *equals* I_3
	Tarifanpassung(I_1), Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2, Richtigstellung_von_Daten_in_DB_TK1(I_3), I_1 *start* I_3, I_2 *equals* I_3 $\mapsto$ Richtigstellung_von_Daten_in_DB_TK2(I_4), I_1 *starts* I_4, I_2 *equals* I_4, I_3 *equals* I_4
	Tarifanpassung_TK2(I_1), Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2 $\mapsto$ Richtigstellung_von_Daten_in_DB_TK2(I_3), I_1 *starts* I_3, I_2 *equals* I_3
	Tarifanpassung_TK2(I_1), Richtigstellung_von_Daten_in_DB(I_2), I_1 *starts* I_2, Richtigstellung_von_Daten_in_DB_TK1(I_3), I_1 *starts* I_3, I_2 *equals* I_3 $\mapsto$ Richtigstellung_von_Daten_in_DB_TK2(I_4), I_1 *starts* I_4, I_2 *equals* I_4, I_3 *equals* I_4
EUV	Antragsannahme_TK2(I_1), sonstige_planmäßige_Änderung_TK2(I_2), I_1 *before* I_2 $\mapsto$ sonstige_planmäßige_Änderung_TK1(I_3), I_1 *before* I_3, I_2 *equals* I_3
	Fortschreibung_Überschuss_vorschüssig(I_1), Antragsannahme_TK1(I_2), I_1 *equals* I_2, Antragsannahme_TK2(I_3), I_1 *equals* I_3, I_2 *equals* I_3, sonstige_planmäßige_Änderung_TK1(I_4), I_1 *before* I_4, I_2 *before* I_4, I_3 *before* I_4 $\mapsto$ sonstige_planmäßige_Änderung_TK2(I_5), I_1 *before* I_5, I_2 *before* I_5, I_3 *before* I_5, I_4 *equals* I_5

Fortsetzung auf der nächsten Seite...

...Fortsetzung von der vorherigen Seite

Fortschreibung_Überschuss_vorschüssig(I_1),
Antragsannahme_TK2(I_2),
I_1 *equals* I_2,
sonstige_planmäßige_Änderung_TK1(I_3),
I_1 *before* I_3, I_2 *before* I_3,
$\mapsto$ sonstige_planmäßige_Änderung_TK2(I_4),
I_1 *before* I_4, I_2 *before* I_4, I_3 *equals* I_4

Antragsannahme_TK1(I_1),
Antragsannahme_TK2(I_2),
I_1 *equals* I_2,
sonstige_planmäßige_Änderung_TK1(I_3),
I_1 *before* I_3, I_2 *before* I_3
$\mapsto$ sonstige_planmäßige_Änderung_TK2(I_4),
I_1 *before* I_4, I_2 *before* I_4, I_3 *equals* I_4

Fortschreibung_Überschuss_vorschüssig(I_1),
Antragsannahme_TK2(I_2),
I_1 *equals* I_2,
sonstige_planmäßige_Änderung(I_3),
I_1 *meets* I_3, I_2 *meets* I_3,
sonstige_planmäßige_Änderung_TK1(I_4),
I_1 *meets* I_4, I_2 *meets* I_4, I_3 *equals* I_4
$\mapsto$ sonstige_planmäßige_Änderung_TK2(I_5),
I_1 *meets* I_5, I_2 *meets* I_5, I_3 *meets* I_5, I_4 *equals* I_5

Die Regeln bestehen wie bisher aus verschiedenen Kombinationen der häufigen Änderungsarten wie zum Beispiel „Fortschreibung Überschuss vorschüssig", „Antragsannahme" oder „sonstige planmäßige Änderung", wobei die einzelnen Änderungen natürlich entweder dem gesamten Vertrag und/oder einer oder mehreren Tarifkomponenten zugeordnet sind. Da viele Verträge nur aus einigen wenigen Tarifkomponenten bestehen, betreffen die Änderungen an Tarifkomponenten größtenteils die erste oder zweite Komponente. Auch ein komplexes „Zusammenspiel" der Änderungen der einzelnen Tarifkomponenten wurde nicht gefunden. Häufig ist es so, dass eine Änderung sowohl den ganzen Vertrag als auch die einzelnen Tarifkomponenten betrifft. Solche Änderungen sind in den Regeln in Tabelle 5.4 dadurch zu erkennen, dass die entsprechenden Intervalle alle zeitgleich sind, also zueinander die Relation *equals* aufweisen.

Somit bestätigt sich der bei der Analyse der Änderungen der Tarifkomponenten gewonnene Eindruck, dass entweder keine interessanten Muster in den Daten vorhanden sind, oder dass die Definition der Eingabesequenzen die in den Daten vorhandene Information nicht entsprechend wiedergibt. Aus zeitlichen Gründen und wegen mangelnder Dokumentation der Daten wurde aber nicht versucht, andere, zum Lernen eventuell besser geeignete Eingabesequenzen zu konstruieren.

5.4.2 Vorhersage von Rückkauf

Mit einer weiteren Untersuchung soll überprüft werden, ob sich der Rückkauf von Versicherungsverträgen bei Einbeziehung der Änderungen aller Tarifkomponenten eines Vertrags besser vorhersagen lässt als auf der Ebene der Tarifkomponenten, vgl. Abschnitt 5.3.7. Wie dort wurden die Vertragsdaten der Kapitallebensversicherung, der Renten- sowie der Erwerbsunfähigkeitsversicherung nochmal in zwei Gruppen, die zurückgekauften und die nicht zurückgekauften Versicherungsverträge, unterteilt. Aus jeder Partition wurde dann eine Stichprobe von 1 500 Verträgen gezogen, die mit den Minimalhäufigkeiten 10% und 5% analysiert wurden. Alle gefundenen Regeln wurden mit Hilfe des *J-measures* bewertet und die 20 am höchsten gewerteten Regeln wurden näher untersucht.

Wie bei dem Versuch, den Rückkauf auf der Ebene der Tarifkomponenten vorherzusagen, liefern auch die bei dieser Untersuchung gefundenen Regeln keine Hinweise darauf, warum ein Versicherungsvertrag zurückgekauft wird. Die Prämissen der Regeln, die den Rückkauf eines Vertrags vorhersagen, bestehen wie in Abschnitt 5.3.7 schon erläutert, aus häufigen Änderungen, die ebenfalls in der Prämisse vieler Regeln vorkommen, die in den Daten der nicht zurückgekauften Versicherungsverträge gefunden wurden. Somit ist es auch auf der Ebene der Versicherungsverträge nicht möglich, den Rückkauf einer Versicherung allein auf Grund ihrer Änderungen vorherzusagen. Wahrscheinlich spielen beim Rückkauf einer Versicherung andere Einflüsse eine Rolle, die durch die vorliegenden Daten nicht widergespiegelt werden.

6 – Zusammenfassung und Diskussion

Wie in der Einleitung und in der Aufgabenstellung erläutert, war es das Ziel dieser Arbeit, Versicherungsvertragsdaten der Swiss Life/Rentenanstalt mit Hilfe eines zeitlichen Data Mining-Verfahrens zu analysieren. Da alle bisherigen Untersuchungen dieser Daten deren zeitliche Information gar nicht oder nur durch besondere Vorverarbeitungsschritte miteinbezogen haben sollte im Rahmen dieser Arbeit ein Data Mining-Verfahren angewandt werden, das die zeitliche Information der Daten explizit verwendet. Deshalb bestand ein Teil der Arbeit darin, ein geeignetes Verfahren auszuwählen. Da kein passender Data Mining-Algorithmus gefunden wurde, wurde durch Kombination zweier Verfahren ein neuer Data Mining-Algorithmus implementiert. Bei der Analyse der vorliegenden Daten wurde deshalb gleichzeitig der Data Mining-Algorithmus auf seine Anwendbarkeit hin überprüft. Im Folgenden werden sowohl der Algorithmus als auch die bei der Analyse der vorliegenden Daten erzielten Ergebnisse kritisch begutachtet, und Verbesserungsvorschläge und Ideen für weitere Forschungen vorgestellt.

6.1 Ergebnisse der Wissensentdeckung

Das Ziel der im Rahmen dieser Diplomarbeit durchgeführten Analyse der Swiss Life-Daten war, diese nach häufigen Änderungsmustern zu durchsuchen und aus diesen Mustern Regeln zu erzeugen, die die zeitliche Entwicklung von Intervallsequenzen vorhersagen können (vgl. Abschnitt 3.3). Nachdem bei den ersten Untersuchungen die Daten falsch interpretiert wurden, beschränkten sich die meisten Untersuchungen auf die Suche nach typischen Änderungsmustern und Regeln in den Tarifkomponenten, vgl. Abschnitt 5.3.

Neben der Untersuchung technischer Fragestellungen, wie dem Einfluss der Parameter des Algorithmus auf die Anzahl der gefundenen Muster oder der Laufzeiteigenschaften des Verfahrens, wurde versucht, die Anwendung des implementierten Algorithmus an Hand eines ausführlichen Beispiels zu demonstrieren.

Um einen ersten Überblick über die vorhandenen Änderungsmuster zu bekommen, begann die Analyse damit, dass die Tarifkomponenten der fünf verschiedenen Versicherungsarten nach häufigen Intervallmustern durchsucht wurden. Die dabei gefundenen Regeln lieferten zwar keine neuen, nützlichen Informationen über die zeitliche Entwicklung der Tarifkomponenten, zeigten aber zwei Eigenheiten des Data Mining-Algorithmus auf, die bei der Analyse von Intervallsequenzen eine wichtige Rolle spielen.

Zum einen wurden bei niedrigen Minimalhäufigkeiten sogenannte Regelhaufen gefunden, die ein Artefakt der Suche nach häufigen Intervallmustern sind. Das Auftreten von Regelhaufen ließe sich durch Nachbearbeitung der gefundenen Regelmengen vermeiden. Dabei könnte nach Regelhaufen gesucht werden, und von allen Regeln eines Regelhaufens würde dann nur eine beibehalten, zum Beispiel die längste Regel. Diese Möglichkeit wurde aber aus zeitlichen Gründen nicht näher untersucht.

Das andere Phänomen waren Regeln, die sich nur in einer Intervallrelation unterschieden. Häufig waren zum Beispiel nur die Relationen *before* und *meets* bei zwei Regeln gegeneinander ausgetauscht. Solche ähnlichen Regeln könnten daraufhin deuten, dass die Unterscheidung zwischen diesen beiden Intervallrelationen gar nicht so wichtig ist und gelockert werden könnte, d. h. diese Intervallrelation kann als Disjunktion verschiedener Relationen aufgefasst werden. Deshalb wurde die Hypothesensprache des Data Mining-Algorithmus dahingehend erweitert, dass sich auch Disjunktionen von Intervallrelationen darstellen lassen. Bei der Erweiterung des Algorithmus gab es zwei Möglichkeiten: Zum einen konnte die erweiterte Hypothesensprache schon bei der Suche nach häufigen Intervallmustern verwendet werden, was die Anzahl der zu überprüfenden Muster erhöht. Zum anderen konnten häufige Sequenzen, die sich nur in einer Intervallrelation unterschieden, zu einer Sequenz in der erweiterten Hypothesensprache kombiniert werden, wodurch aber eventuell häufige Muster nicht gefunden werden, da es passieren kann, dass die einzelnen Muster die geforderte Minimalhäufigkeit nicht erreichen, das aus ihnen kombinierte Muster in der erweiterten Hypothesensprache diese Minimalhäufigkeit aber schon erreichen würde. Deshalb wurde die erweiterte Hypothesensprache schon bei der Suche nach häufigen Sequenzen verwendet, wobei im hier besprochenen Anwendungsfall die Hypothesensprache nur um die Intervallrelation *older-&-survived-by* ergänzt wurde, die der Disjunktion (*before* ∨ *meets* ∨ *overlaps*) entspricht. Bei der Analyse der Tarifkomponenten erbrachte die erweiterte Hypothesensprache zwar nicht den erhofften Erfolg, aber bei der Analyse anderer Daten kann die sie durchaus besser geeignet sein, die vorhandenen Muster wiederzugeben.

Da der Data Mining-Algorithmus im Allgemeinen eine so große Menge an Regeln gefunden hat, so dass diese nicht vollständig nach interessanten Regeln durchgesehen werden konnte, wurden die gefundenen Regeln mit Hilfe eines Bewertungskriteriums sortiert, und nur die am höchsten gewerteten Regeln wurden näher untersucht. Dazu wurde bei den bisherigen Versuchen das *J-measure* verwendet. Da in der Literatur aber eine ganze Reihe von Regelbewertungskriterien beschrieben sind, wurden einige Kriterien bzgl. der von ihnen am höchsten gewerteten Regeln miteinander verglichen. Eventuell konnten andere Bewertungskriterien interessantere Regeln aus der Menge aller Regeln herausfiltern. Dazu wurden neben dem *J-measure* der *interest* und die *certainty factors* ausgewählt, und die Mengen der von den drei verschiedenen Bewertungskriterien am höchsten gewerteten Regeln wurden miteinander verglichen. Dabei zeigte sich, dass jedes Bewertungskriterium andere Regeln bevorzugt. Die Menge der vom *J-measure* am höchsten gewerteten Regeln variierte bei unterschiedlichen Minimalhäufigkeiten nur sehr gering, d. h. der „Kern“ der interessantesten Regeln blieb in etwa gleich. Nicht so beim *interest* und bei den *certainty factors*, die bei niedrigen Minimalhäufigkeiten längere und komplexere Regeln bevorzugen als bei hohen Minimalhäufigkeiten, d. h. die Menge der am höchsten gewerteten Regeln variiert hier sehr stark. Aufgrund dieser Eigenschaften von *interest* und den *certainty factors* wurde bei den folgenden Untersuchungen weiterhin das *J-measure* zur Bewertung der gefundenen Regeln verwendet. Aber die vom *interest* bzw. von den *certainty factors* am höchsten gewerteten Regeln könnten für andere Anwendungsfälle interessant sein. Hierzu müssten weitere Untersuchungen über die Anwendbarkeit der verschiedenen Regelbewertungskriterien durchgeführt werden.

Ein Nachteil der vom Algorithmus verwendeten Hypothesensprache ist, dass nur die qualitativen Zusammenhänge der Zeitintervalle erfasst werden. Der Zeitpunkt an dem ein Muster in einer Eingabesequenz auftritt oder seine zeitliche Ausdehnung werden nicht berücksichtigt. Für viele Fragestellungen ist es aber interessant, ob ein Muster eher am Anfang der Eingabesequenzen auftritt oder eher am Ende, oder ob Sequenzen, die in verschiedenen Zeitabschnitten erfasst wurden, unterschiedliche Änderungsmuster aufweisen. Um diese quantitativen Aspekte der Zeit zu berücksichtigen kann natürlich die Hypothesensprache um entsprechende Formalismen erweitert werden, ähnlich wie es in [42] für sequentielle Muster gemacht wird. Da aber unklar ist, wie geeignete Erweiterungen der Hypothesensprache aussehen könnten, wurden die Tarifkomponenten vorverarbeitet, um entsprechende zeitliche Veränderungen zu erfassen. Zum einen wurden die Tarifkomponenten bezüglich der Weltzeit gefiltert. Dabei sollte überprüft werden, ob sich die Änderungsmuster der Tarifkomponenten selber mit der Zeit verändert haben. Dazu wurde die Weltzeit in Dekaden unterteilt und die Änderungen, die in einer Dekade auftraten, wurden nach typischen Änderungsmustern durchsucht. Dabei zeigte sich, dass vor dem Jahr 1990 die meisten Tarifkomponenten kaum geändert wurden. Die meisten Regeln wurden in den Zeiträumen 1990–1999 und 2000–2009 gefunden. Dies führte zu der Vermutung, dass die Änderungen der Tarifkomponenten – oder allgemein Änderungen an den Versicherungsverträgen – erst seit Mitte der 90er Jahr des letzten Jahrhunderts detailliert aufgezeichnet wurden. Vorher wurden wahrscheinlich nur sehr wichtige Änderungen in den Akten vermerkt. Auf Grund des kurzen Zeitraums, den die gefundenen Regeln widerspiegeln, ist es natürlich fraglich, ob die Suche nach typischen Änderungsmustern überhaupt Sinn macht, da nur Regeln gefunden werden können, die Änderungen der letzten 10 Jahre beschreiben.

Dennoch wurden noch einige weitere Untersuchungen bzgl. der zeitlichen Veränderung der Änderungsmuster durchgeführt. 1998 wurde in der Schweiz die sogenannte Stempelsteuer eingeführt, die u. a. auf Quittungen von Versicherungsprämien erhoben wird und dazu geführt hat, dass viele Kunden ihre Versicherungsverträge gekündigt haben. Es wurde untersucht, ob sich die Änderungsmuster der Tarifkomponenten durch die Einführung der Stempelsteuer geändert haben. Aber auch bei diesen Analysen wurden keine interessanten Ergebnisse erzielt. Schließlich wurden die Tarifkomponenten noch auf Änderungen der häufigen Intervallmuster mit der Laufzeit der Tarifkomponenten untersucht. Dazu wurde die Laufzeit der Tarifkomponenten in drei Abschnitte unterteilt, und jeder Abschnitt wurde nach interessanten Regeln durchsucht. Die gefundenen Regeln zeigten zwar teilweise unterschiedliche Änderungen, da z. B. die Änderung „Neueintritt“ verständlicherweise nur im ersten Drittel der Laufzeit einer Tarifkomponente auftritt, aber wie bei allen bisherigen Untersuchungen wurden keine wirklich neuen, interessanten Regeln gefunden. Dennoch haben die Untersuchungen bzgl. Veränderungen der Änderungsmuster mit der Zeit gezeigt, wie Schwächen der Hypothesensprache eines Algorithmus durch geeignete Vorverarbeitung der Daten umgangen werden können.

Die letzten Untersuchungen der Tarifkomponenten beschäftigten sich mit der Vorhersage des Rückkaufs eines Versicherungsvertrags. Unter einem Rückkauf versteht man die vorzeitige Beendigung einer Versicherung wegen Rücktritt oder Kündigung, wobei dem Versicherungsnehmer der aktuelle Zeitwert der Versicherung auszuzahlen ist. Da der Rückkauf von Versicherungen für die Versicherungsgesellschaft immer ein Verlustgeschäft ist, ist diese daran interessiert, den Rückkauf von Versicherungen zu verhinden. Deshalb wurde untersucht, ob es bestimmte Änderungsmuster gibt, die auf den möglichen Rückkauf eines Versicherungsvertrags hindeuten. Dazu wurden die Tarifkomponenten der einzelnen Versicherungsarten nochmals in zwei Gruppen unterteilt, die zurückgekauften und die nichtzurückgekauften Tarifkomponenten, und jede Gruppe wurde nach typischen Änderungsmustern durchsucht. Leider stellte sich heraus, dass der Rückkauf nicht auf Grund von typischen Änderungesmustern vorhersagbar ist, weil die in den zurückgekauften Tarifkomponenten gefundenen Regeln Prämissen enthielten, die in gleicher Form auch in den Regeln aus den nichtzurückgekauften Tarifkomponenten auftraten.

Es kann natürlich dennoch sein, dass einige der gefundenen Regeln den Rückkauf eines Versicherungsvertrags vorhersagen können, auf Grund der Auswertung der gefundenen Regeln aber übersehen wurden. Da bei der Vorhersage des Rückkaufs Regeln gesucht wurden, die zwei disjunkte Datenmengen voneinander unterscheiden, könnte man versuchen, das Verfahren aus [73] auf Intervallmuster zu übertragen. Vielleicht ist es dann möglich, sehr allgemeine und uninteressante Regeln aus der Menge aller gefundenen Regeln herauszufiltern und die wirklich interessanten Regelmäßigkeiten zu finden. Entsprechende Untersuchungen wurden aber im Rahmen dieser Arbeit aus zeitlichen Gründen nicht durchgeführt.

Neben der Analyse der Tarifkomponenten, bei der gleichzeitig eine Anleitung zur Anwendung des für diese Arbeit implementierten Data Mining-Algorithmus gegeben wurde, wurden noch einige Untersuchungen durchgeführt, mit denen Änderungsmuster der Versicherungsverträge analysiert werden sollten. Dazu musste die Vorverarbeitung dahingehend angepasst werden, dass die einzelnen Änderungen eindeutig einer Tarifkomponente und/oder dem gesamten Vertrag zugeordnet werden konnten. Durch die Kombination der Änderungen der Tarifkomponenten und der Versicherungsverträge wurden die Eingabesequenzen natürlich länger als bei den bisherigen Untersuchungen, was Probleme bei der Anwendung der Data Mining-Algorithmen bereitete. Der nichtoptimierte Algorithmus war nicht verwendbar, da soviele Kandidatenmuster erzeugt wurden, dass die Berechnung der Häufigkeit nicht mehr in annehmbarer Zeit durchführbar war. Der optimierte Algorithmus hingegen benötigte zur Analyse aller Eingabesequenzen zuviel Speicher. Deshalb wurde nur eine Stichprobe von 1 500 Sequenzen pro Versicherungsart zum Lernen verwendet. Die Versicherungsverträge wurden zum einen nach typischen Änderungsmustern durchsucht, und zum anderen wurde versucht, den Rückkauf eines Versicherungsvertrags auf Grundlage der Änderungsmuster aller beteiligten Tarifkomponenten vorherzusagen. Beide Untersuchungen erbrachten aber ähnlich schlechte Ergebnisse wie die entsprechenden Analysen der Tarifkomponenten, vgl. Abschnitt 5.4.

Zusammenfassend lässt sich sagen, dass alle Untersuchungen der Tarifkomponenten und ebenso die Untersuchungen der Versicherungen kein neues, nützliches Wissen erbrachten. Die gefundenen Regeln enthielten ausschließlich Intervalle, die typische Änderungen repräsentieren, und auch der Rückkauf ließ sich nicht auf Grund besonderer Änderungsmuster vorhersagen. Diese Ergebnisse lassen zwei mögliche Rückschlüsse zu: Zum einen könnten die Daten nicht adäquat repräsentiert worden sein, d. h. in den Daten stecken möglicherweise nützliche Informationen, die aber in den durchgeführten Analysen nicht verwendet wurden. Zum anderen kann es natürlich sein, dass wirklich keine interessanten Informationen in den Daten vorhanden sind. Eine Erklärung für die Ergebnisse der Untersuchungen lässt sich nur mit Hilfe eines Fachmanns von der Swiss Life finden. Er könnte vielleicht Hinweise auf mögliche Fehler in der Interpretation der Daten liefern und Vorschläge für weitere Untersuchungen machen.

Dennoch können auch ohne die Hinweise eines Experten Vorschläge für weitere Datenanalysen gegeben werden. So könnten zum Beispiel die demographischen Daten Informationen enthalten, mit denen sich der Rückkauf von Versicherungen besser vorhersagen lässt, da vielleicht Personen mit einem höheren Einkommen ihre Verträge seltener zurückkaufen als Personen mit geringem Einkommen. Andere Untersuchungen, welche die Informationen über die Partner miteinbeziehen, könnten z. B. nach typischen Sequenzen in den Vertragsabschlüssen eines Partners suchen. Vielleicht schließen viele Kunden Versicherungen in einer typischen Reihenfolge ab, und die Swiss Life könnte diese Information nutzen, ihre Kunden besser zu betreuen.

6.2 Lernen häufiger zeitlicher Intervallmuster

Neben der Analyse der Versicherungsvertragsdaten bestand ein sehr wichtiger Teil dieser Diplomarbeit darin, ein geeignetes Data Mining-Verfahren zu finden. Das Verfahren sollte die Aufgabe des Lernens häufiger zeitlicher Intervallmuster aus einer Menge von Intervallsequenzen lösen können.

Eine Literaturrecherche ergab, dass es nur sehr wenige Data Mining-Algorithmen gibt, die Muster in Intervallsequenzen suchen können. Eine detaillierte Analyse der in der Literatur beschriebenen Algorithmen ergab, dass diese nicht geeignet sind, die gestellte Lernaufgabe zu lösen. Das Verfahren aus [43] kann zwar Muster aus einer Menge von Eingabesequenzen lernen, verwendet aber eine ambivalente Hypothesensprache, die auf Grund ihres Aufbaus bestimmte Muster nicht unterscheiden kann. Der Algorithmus aus [39, 37] besitzt zwar eine gut geeignete Hypothesensprache, kann aber nur Muster aus einer langen Intervallsequenz lernen. Deshalb wurden beide Verfahren miteinander kombiniert, um einen Data Mining-Algorithmus zu implementieren, der die gestellte Lernaufgabe lösen kann. Dieses Verfahren verwendet die Hypothesensprache aus [39, 37], um die Intervallmuster zu beschreiben, nach denen in einer Menge von Eingabesequenzen gesucht wird. Aus der Menge der gefundenen Intervallmuster werden in einem weiteren Schritt Regeln berechnet, die Voraussagen über die mögliche weitere Entwicklung einer Intervallsequenz machen.

Bei der Analyse der Versicherungsvertragsdaten stellte sich heraus, dass die Intervallrelationen (vgl. [7]), mit denen die Muster beschrieben werden, für manche Anwendungen zu strikt sind. Deshalb wurde unter Verwendung von Ideen aus [25] eine erweiterte Hypothesensprache entwickelt, mit der „weichere“ Intervallrelationen beschrieben werden können. Die erweiterte Hypothesensprache brachte bei der Analyse der vorliegenden Daten zwar nicht die erhofften Ergebnisse, ist aber für andere Anwendungen durchaus geeignet, die wirklichen Muster in den Daten besser zu erfassen als die ursprüngliche Hypothesensprache.

Ein Nachteil des Algorithmus – sowohl bei Verwendung der ursprünglichen als auch der erweiterten Hypothesensprache – ist das exponentielle Wachstum der Menge der zu überprüfenden Muster. Dieses exponentielle Wachstum wird dadurch verursacht, dass die Hypothesensprachen die Struktur der möglichen Muster in keiner Weise einschränken. Indem bestimmte Nebenbedingungen für die möglichen Muster formuliert werden, könnte die Größe des Hypothesenraums beschränkt und das Finden interessanter Muster und Regeln erheblich beschleunigt werden. Ebenso könnten Nebenbedingungen dafür sorgen, dass zum Beispiel nur nach solchen Mustern gesucht wird, die am Anfang einer Intervallsequenz auftreten, oder die eine bestimmte zeitliche Ausdehnung nicht überschreiten. [42] könnte hier einige Anregungen geben.

Auf den ersten Blick ungewohnt sind sicherlich auch die aus häufigen Mustern P und Q erzeugten Regeln $P \mapsto Q$, da die Prämisse P Bestandteil der Konklusion Q ist. Hier könnte untersucht werden, ob eine Änderung der Regelsemantik dahingehend, dass Prämisse und Konklusion disjunkt sind, für manche Anwendungen besser geeignet ist. Eine Diskussion hierzu findet sich in Abschnitt 4.4.3.

Ein Werkzeug, dass bei der Suche nach interessanten Regeln eine große Hilfe sein könnte, ist ein sogenannter *Regelbrowser* [49]. Mit ihm könnten die gefundenen Regeln interaktiv analysiert werden. So könnten die Regeln zum Beispiel nach bestimmten Mustern in der Prämisse oder der Konklusion gefiltert oder mit objektiven Bewertungskriterien sortiert werden. Ein Regelbrowser wäre aber nicht nur geeignet, Intervallregeln interaktiv zu analysieren, sondern könnte auch mit anderen Regelmengen wie Assoziationsregeln arbeiten. Voraussetzung hierfür wäre, dass die Regeln in einem allgemeinen Format vorliegen. Die RuleML [17] könnte hierzu einen Ansatz bieten.

Neben der Erweiterung des Algorithmus wäre es wünschenswert, das in dieser Arbeit verwendete Data Mining-Verfahren auf anderen Daten einzusetzen, um seine Praxistauglichkeit weiter zu überprüfen. Da Intervallsequenzen in vielen Anwendungsbereichen auftreten, ist der implementerte Algorithmus mit den beiden Hypothesensprachen ein gut geeignetes Werkzeug, solche Daten nach interessanten Mustern und Regeln zu durchsuchen.

Literatur

[1] Rakesh Agrawal, Christos Faloutsos und Arun Swami: *Efficient Similarity Search In Sequence Databases.* In: *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, Band 730 der Reihe *Lecture Notes in Computer Science*, Seiten 69–84. Springer Verlag, 1993.

[2] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney und Kyuseok Shim: *Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases.* In: *Proceedings of 21st International Conference on Very Large Data Bases*, Seiten 490–501. Morgan Kaufmann, 1995.

[3] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen und A. Inkeri Verkamo: *Fast Discovery of Association Rules.* In: Usama M. Fayyad, Gregory Piatetsky-Shapiro und Padrahic Smyth (Herausgeber): *Advances in Knowledge Discovery in Databases*, Seiten 307–328. AAAI/MIT Press, 1996.

[4] Rakesh Agrawal, Guiseppe Psaila, Edward L. Wimmers und Mohamed Zaït: *Querying Shapes of Histories.* In: *Proceedings of 21st International Conference on Very Large Data Bases*, Seiten 502–514. Morgan Kaufmann, 1995.

[5] Rakesh Agrawal und Ramakrishnan Srikant: *Fast Algorithms for Mining Association Rules.* In: Jorge B. Bocca, Matthias Jarke und Carlo Zaniolo (Herausgeber): *Proceedings of the 20th International Conference on Very Large Data Bases*, Seiten 487–499. Morgan Kaufmann, 1994.

[6] Rakesh Agrawal und Ramakrishnan Srikant: *Mining Sequential Patterns.* In: *Proceedings of the 11th International Conference on Data Engineering*, Seiten 3–14. IEEE Computer Society, 1995.

[7] James F. Allen: *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM, 26(11):832–843, 1983.

[8] Steffan Baron und Myra Spiliopoulou: *Monitoring Change in Mining Results.* In: *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, Band 2114 der Reihe *Lecture Notes in Computer Science*, Seiten 51–60. Springer Verlag, 2001.

[9] Markus Bauer: *Simultane Ausreißer- und Interventionsidentifikation bei Online-Monitoring-Daten.* Doktorarbeit, Fachbereich Statistik, Universität Dortmund, 1999.

[10] Fabian Bauschulte, Ingrid Beckmann, Stefan Haustein, Christian Hüppe, Zoulfa El Jerroudi, Hanna Köpcke, Philip Look, Katharina Morik, Boris Shulimovich, Klaus Unterstein und Daniel Wiese: *Endbericht PG-402 – Wissensmanagement.* Technischer Bericht, Universität Dortmund, Fachbereich Informatik, 2002.

[11] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider und Bernhard Seeger: *The R^*-Tree: An Efficient and Robust Access Method for Points and Rectangles*. In: *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, Seiten 322–331. ACM Press, 1990.

[12] Pavel Berkhin: *Survey of Clustering Data Mining Techniques*. Technischer Bericht, Accrue Sotware, 2002. Online verfügbar unter http://www.accrue.com/products/researchpapers.html.

[13] Donald J. Berndt und James Clifford: *Finding Patterns in Time Series: A Dynamic Programming Approach*. In: Usama M. Fayyad, Gregory Piatetsky-Shapiro und Padrahic Smyth (Herausgeber): *Advances in Knowledge Discovery in Databases*, Seiten 229–248. AAAI/MIT Press, 1996.

[14] Michael Berthold und David J. Hand (Herausgeber): *Intelligent Data Analysis*. Springer Verlag, 1999.

[15] Fernando Berzal, Ignacio Blanco, Daniel Sánchez und María Amparo Vila Miranda: *A New Framework to Assess Association Rules*. In: *Proceedings of the 4th International Conference on the Advances in Intelligent Data Analysis*, Band 2189 der Reihe *Lecture Notes in Computer Science*, Seiten 95–104. Springer Verlag, 2001.

[16] Hendrick Blockeel, Johannes Fürnkranz, Alexia Prskawetz und Francesco Billari: *Detecting Temporal Change in Event Sequences: An Application to Demographic Data*. In: Luc De Raedt und Arno Siebes (Herausgeber): *Proceedings of the 5th European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 2168 der Reihe *Lecture Notes in Computer Science*, Seiten 29–41. Springer Verlag, 2001.

[17] Harold Boley, Said Tabet und Gerd Wagner: *Design Rationale of RuleML: A Markup Language for Semantic Web Rules*. In: *Proceedings of the International Semantic Web Working Symposium*, 2001. Online verfügbar unter http://www.semanticweb.org/SWWS/program/index.html.

[18] Sergey Brin, Rajeev Motwani und Craig Silverstein: *Beyond Market Baskets: Generalizing Association Rules to Correlations*. In: *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Seiten 265–276. ACM Press, 1997.

[19] Kin-pong Chan und Ada Wai-chee Fu: *Efficient Time Series Matching by Wavelets*. In: *Proceedings of the 15th International Conference on Data Engineering*, Seiten 126–133. IEEE Computer Society Press, 1997.

[20] Xiaodong Chen und Ilias Petrounias: *Mining Temporal Features in Associtaion Rules*. In: *Proceedings of the 3rd European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 1704 der Reihe *Lectures Notes in Computer Science*, Seiten 295–300. Springer Verlag, 1999.

[21] Gautam Das, Dimitrios Gunopulos und Heikki Mannila: *Finding Similar Time Series*. In: *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, Band 1263 der Reihe *Lectures Notes in Computer Science*, Seiten 88–100. Springer Verlag, 1997.

[22] Sažo Džeroski und Nada Lavrač (Herausgeber): *Relational Data Mining*. Springer Verlag, 2001.

[23] Christos Faloutsos, M. Ranganathan und Yannis Manolopoulos: *Fast Subsequence Matching in Time-Series Databases.* In: *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Seiten 419–429. ACM Press, 1994.

[24] Usama M. Fayyad, Gregory Piatetsky-Shapiro und Padrahic Smyth: *From Data Mining to Knowledge Discovery: An Overview.* In: Usama M. Fayyad, Gregory Piatetsky-Shapiro und Padrahic Smyth (Herausgeber): *Advances in Knowledge Discovery in Databases*, Seiten 1–34. AAAI/MIT Press, 1996.

[25] Christian Freksa: *Temporal Reasoning Based on Semi-Intervals.* Artificial Intelligence, 54(1):199–227, 1992.

[26] Frank von Fürstenwerth und Alfons Weiss: *Versicherungsalphabet.* Verlag Versicherungswirtschaft, 10. Auflage, 2001.

[27] Thomas Gärtner, Shaomin Wu und Peter Flach: *Data Mining on the Sisyphus Dataset: Evaluation and Integration of Results.* In: *Proceedings of IDDM-01 – Workshop on Integration of Data Mining, Decision Support, and Meta-Learning*, Freiburg, Germany, 2001. Online verfügbar unter http://www.informatik.uni-freiburg.de/~ml/ecmlpkdd/WS-Proceedings/w04/index.html.

[28] Pierre Geurts: *Pattern Extraction for Time Series Classification.* In: *Proceedings of the 5th European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 2168 der Reihe *Lecture Notes in Computer Science*, Seiten 115–127. Springer Verlag, 2001.

[29] Xianping Ge und Padhraic Smyth: *Deformable Markov Model Templates for Time-Series Pattern Matching.* In: *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 81–90. ACM Press, 2000.

[30] Hans P. Goll, W. Gilbert und H.-J. Steinhaus: *Handbuch der Lebensversicherung.* Verlag Versicherungswirtschaft, 11. Auflage, 1992.

[31] Günther Görz, Claus-Rainer Rollinger und Josef Schneeberger (Herausgeber): *Handbuch der Künstlichen Intelligenz*, Kapitel Maschinelles Lernen und Data Mining. Oldenbourg Verlag, 2000.

[32] Valery Guralnik und Jaideep Srivastava: *Event Detection from Time Series Data.* In: *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 33–42. ACM Press, 1999.

[33] Antonin Guttmann: *R-Trees: A Dynamic Index Structure for Spatial Searching.* In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, Seiten 47–57. ACM Press, 1984.

[34] Joachim Hartung, Bärbel Elpelt und Karl-Heinz Klösener: *Statistik.* Oldenbourg, 12. Auflage, 1999.

[35] David Heckerman: *Bayesian Networks for Knowledge Discovery.* In: Usama M. Fayyad, Gregory Piatetsky-Shapiro und Padrahic Smyth (Herausgeber): *Advances in Knowledge Discovery in Databases*, Seiten 273–305. AAAI/MIT Press, 1996.

[36] Robert J. Hilderman und Howard J. Hamilton: *Knowledge Discovery and Interestingness Measures: A Survey.* Technischer Bericht, Department of Computer Science, University of Regina, 1999.

[37] Frank Höppner: *Learning Temporal Rules from State Sequences.* In: *IJCA Workshop on Learning from Temporal and Spatial Data*, Seiten 25–31, 2001.

[38] Frank Höppner: *Discovery of Core Episodes from Sequences.* In: *Pattern Detection and Discovery*, Band 2447 der Reihe *Lecture Notes in Computer Science*, Seiten 199–213. Springer Verlag, 2002.

[39] Frank Höppner und Frank Klawonn: *Finding Informative Rules in Interval Sequences.* In: Frank Hoffmann, David J. Hand, Niall M. Adams, Douglas Fisher und Gabriela Guimarães (Herausgeber): *Proceedings of the 4th International Conference on the Advances in Intelligent Data Analysis*, Band 2189 der Reihe *Lecture Notes in Computer Science*, Seiten 125–134. Springer Verlag, 2001.

[40] Frank Höppner und Frank Klawonn: *Intelligent Systems: Techniques and Applications*, Kapitel Learning Rules about the Development of Variables over Time. CRC Press, 2002.

[41] Christian S. Jensen, James Clifford, Shashi K. Gadia, Arie Segev und Richard T. Snodgrass: *A Glossary of Temporal Database Concepts.* SIGMOD Record, 21(3):35–43, 1992.

[42] Mahesh Joshi, George Karypis und Vipin Kumar: *A Universal Formulation of Sequential Patterns.* In: *Proceedings of the KDD'2001 Workshop on Temporal Data Mining.* Online verfügbar unter http://www.acm.org/sigkdd/kdd2001/Workshops/TemporalMiningWorkshop.html.

[43] Po-shan Kam und Ada Wai-chee Fu: *Discovering Temporal Patterns for Interval-based Events.* In: Yahiko Kambayashi, Mukesh K. Mohania und A. Min Tjoa (Herausgeber): *DaWaK 2000*, Band 1874 der Reihe *Lecture Notes in Computer Science*, Seiten 317–326. Springer Verlag, 2000.

[44] Eamonn Keogh: *Exact Indexing of Dynamic Time Warping.* In: *Proceedings of the 28th International Conference on Very Large Data Bases*, Seiten 406–417. Morgan Kaufmann, 2002.

[45] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani und Sharad Mehrotra: *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases.* Knowledge and Information Systems, 3(3):263–286, 2001.

[46] Eamonn Keogh, Selina Chu, David Hart und Michael Pazzani: *An Online Algorithm for Segmenting Time Series.* In: *Proceedings of the 2001 IEEE International Conference on Data Mining*, Seiten 289–296. IEEE Computer Society, 2001.

[47] Eamonn Keogh und Michael Pazzani: *Scaling up Dynamic Time Warping for Datamining Applications.* In: *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 285–289. ACM Press, 2000.

[48] Joerg-Uwe Kietz und Martin Staudt: *KDD Sisyphus – Data Preparation, Preprocessing and Reasoning for Real-World Data.* Geplanter Workshop im Rahmen der 10th European Conference on Machine Learning, 1998. Workshop-Seite http://research.swisslife.ch/kdd-sisyphus/ nicht mehr erreichbar.

[49] Mika Klemettinen, Heikki Mannila und A. Inkeri Verkamo: *Association Rule Selection in a Data Mining Environment.* In: *Proceedings of the 3rd European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 1704 der Reihe *Lecture Notes in Computer Science*, Seiten 372–377. Springer Verlag, 1999.

[50] Georg Lausen, Itzok Savnik und Aldar Dougarjapov: *MSTS: A System for Mining Sets of Time Series.* In: *Proceedings of the 4th European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 1910 der Reihe *Lecture Notes in Computer Science*, Seiten 289–298. Springer Verlag, 2000.

[51] Fredrik Linåker und Lars Niklasson: *Time Series Segmentation Using an Adaptive Resource Allocating Vector Quantization Network Based on Change Detection.* In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Seiten 323–328. IEEE Computer Society, 2000.

[52] Yingjiu Li, X. Sean Wang und Sushil Jajodia: *Discovering Temporal Patterns in Multiple Granularities.* In: *Proceedings of the 1st International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, Band 2007 der Reihe *Lecture Notes in Computer Science*, Seiten 5–19. Springer Verlag, 2000.

[53] Heikki Mannila, Hannu Toivonen und A. Inkeri Verkamo: *Discovery of Frequent Episodes in Event Sequences.* Data Mining and Knowledge Discovery, 1(3):259–289, 1997.

[54] Katharina Morik: *The Representation Race – Preprocessing for Handling Time Phenomena.* In: Ramon López de Mántaras und Enric Plaza (Herausgeber): *Proceedings of the 11th European Conference on Machine Learning*, Band 1810 der Reihe *Lecture Notes in Computer Science*, Seiten 4–19. Springer Verlag, 2000.

[55] Tim Oates, Laura Firoiu und Paul R. Cohen: *Clustering Time Series with Hidden Markov Models and Dynamic Time Warping.* Vortrag im Rahmen des *IJCAI'99 Workshop on Neural, Symbolic, and Reinforcement Methods for Sequence Learning*, 1999. Online verfügbar unter http://www.neci.nj.nec.com/homepages/giles/IJCAI99/workshop.html.

[56] Tim Oates, Laura Firoiu und Paul R. Cohen: *Using Dynamic Time Warping to Bootstrap HMM-Based Clustering of Time Series.* In: *Sequence Learning – Paradigms, Algorithms, and Applications*, Band 1828 der Reihe *Lecture Notes in Computer Science*, Seiten 35–52. Springer Verlag, 2001.

[57] Banu Özden, Sridhar Ramaswamy und Avi Silberschatz: *Cyclic Associtaion Rules.* In: *Proceedings of the 14th International Conference on Data Engineering*, Seiten 412–421. IEEE Computer Society, 1998.

[58] Balaji Padmanabhan und Alexander Tuzhilin: *Pattern Discovery in Temporal Databases: A Temporal Logic Approach.* In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Seiten 351–354. AAAI Press, 1996.

[59] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang und D. Stott Parker: *Landmarks: A New Model for Similarity-Bsed Pattern Querying in Time Series Databases.* In: *Proceedings of the 16th International Conference on Data Engineering*, Seiten 33–42. IEEE Computer Society, 2000.

[60] Richard J. Povinelli: *Identifying Temporal Patterns for Characterization and Prediction of Financial Time Series Events.* In: *First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, Band 2007 der Reihe *Lecture Notes in Computer Science*, Seiten 46–61. Springer Verlag, 2000.

[61] J. Ross Quinlan: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[62] John F. Roddick, Kathleen Hornsby und Myra Spiliopoulou: *An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research.* In: *Proceedings of the 1st International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, Band 2007 der Reihe *Lecture Notes in Computer Science*, Seiten 147–164. Springer Verlag.

[63] John F. Roddick und Myra Spiliopoulou: *A Survey of Temporal Knowledge Discovery Paradigms and Methods.* IEEE Transactions on Knowledge and Data Engineering, 14(4):750–767, 2001.

[64] Apkar Salatian und Jim Hunter: *Deriving Trends in Historical and Real-Time Continuously Sampled Medical Data.* Journal of Intelligent Information Systems, 13(1–2):47–71, 1999.

[65] Rainer Schlittgen und Bernd H. J. Streitberg: *Zeitreihenanalyse.* Oldenbourg, 9. Auflage, 2001.

[66] Padhraic Smyth und Rodney M. Goodman: *Rule Induction Using Information Theory.* In: Gregory Piatetsky-Shapiro und William J. Frawley (Herausgeber): *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

[67] Myra Spiliopoulou: *Managing Interesting Rules in Sequence Mining.* In: *Proceedings of the 3rd European Conference on the Principles of Data Mining and Knowledge Discovery*, Band 1704 der Reihe *Lecture Notes in Computer Science*, Seiten 554–560. Springer Verlag, 1999.

[68] Ramakrishnan Srikant und Rakesh Agrawal: *Mining Sequential Patterns: Generalizations and Performance Improvements.* In: *Advances in Database Technology – EDBT'96, Proceedings of the 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996*, Band 1057 der Reihe *Lecture Notes in Computer Science.* Springer Verlag, 1996.

[69] Abdullah Uz Tansel, James Clifford, Sushil Gadia, Shashi ans Jajodia, Arie Segev und Richard T. Snodgrass: *Temporal Databases: Theory, Design and Implementation.* Benjamin Cummings, 1993.

[70] Vladimir Vapnik: *Statistical Learning Theory.* Wiley, 1998.

[71] Roy Villafane, Kien A. Hua und Duc Tran: *Knowledge Discovery from Series of Interval Events.* Journal of Intelligent Information Systems, 15(1):71–89, 2000.

[72] Andreas S. Weigend und Neil A. Gershenfeld (Herausgeber): *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1994.

[73] Mohammed J. Zaki, Neal Lesh und Mitsunori Ogihara: PlanMine*: Sequence Mininig for Plan Failures.* In: *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, Seiten 369–374. AAAI Press, 1998.