Zustandsprädiktoren zur Kontextvorhersage in ubiquitären Systemen

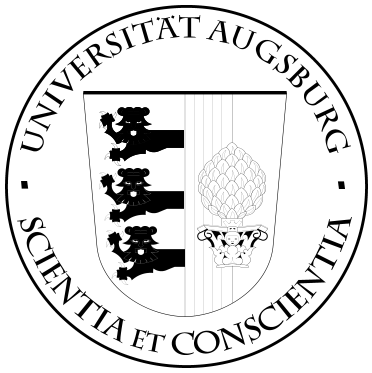
Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

der Fakultät für Angewandte Informatik

der Universität Augsburg



von

Jan Petzold

Erstgutachter: Prof. Dr. rer. nat. Theo Ungerer
Zweitgutachter: Prof. Dr. rer. nat. Bernhard Bauer

Tag der mündlichen Prüfung: 10. November 2005

Zusammenfassung

Ubiquitäre Computersysteme werden nach den Generationen der Großrechner und der Personal Computer als die dritte Rechnergeneration gesehen. Die Benutzer sind von vielen Rechnern in Alltagsgeräten umgeben, die - für den Menschen unsichtbar - eigenständig Unterstützungen im täglichen Leben bieten. Um sich in hohem Maße auf den Menschen einzustellen, werden Informationen der Umgebung benötigt, welche als Kontext bezeichnet werden. Eine weiterreichende Unterstützung besteht darin, dem Menschen Entscheidungen abzunehmen, die aus Gewohnheiten resultieren. Hierzu müssen Kontexte vorhergesagt werden, so dass das System in einem vom Benutzer gewünschten Maße proaktiv handeln kann.

Um Kontexte zu erlernen und damit vorherzusagen, können verschiedene Verfahren des Maschinellen Lernens eingesetzt werden. Neuronale Netze, Bayessche Netze sowie Markov-Modelle sind die wohl bekanntesten Vertreter. Vorhersageverfahren sind aber auch in anderen Bereichen der Informatik zu finden. Zur Vorhersage bedingter Sprungbefehle in aktuellen Mikroprozessoren werden Verfahren eingesetzt, die eine Treffergenauigkeit von bis zu 98% erreichen. Können diese Verfahren auf die Kontextvorhersage in ubiquitären Systemen übertragen werden? Können dabei auch solch gute Ergebnisse erzielt werden?

Die Zustandsprädiktoren sind durch verschiedene Verfahren der Sprungvorhersage motiviert. Sie zeigen eine Möglichkeit wie diese Sprungvorhersagetechniken auf die Kontextvorhersage übertragen werden können. Mit Erweiterungen der Zustandsprädiktoren durch Zuverlässigkeitsschätzung und Hybridverfahren wurde bei der Vorhersage des nächsten Aufenthaltsortes einer Person eine Vorhersagegenauigkeit von bis zu 90% erreicht. Im Vergleich mit bekannten Verfahren wie Neuronalen Netzen, Bayesschen Netzen und Markov-Modellen verhalten sich die Zustandsprädiktoren bezüglich der Vorhersagegenauigkeit ähnlich. Bei der An- und Umlerngeschwindigkeit liegen sie vor den anderen Verfahren.

Vorwort

Diese Arbeit entstand in den Jahren 2002 bis 2005 am Lehrstuhl für Systemnahe Informatik und Kommunikationssysteme der Universität Augsburg. Der Ausgangspunkt war die Idee, die einfachen und sehr genauen Verfahren der Sprungvorhersage auf die Kontextvorhersage zu übertragen. Mit den *Smart Doorplates* wurde 2003 eine erste Anwendung entwickelt, die die vorgeschlagenen Verfahren zur Kontextvorhersage nutzte. Ein Prototyp dieser Anwendung wurde auf der *Nacht der Wissenschaft* im Juni 2003 vorgeführt. 2004 wurde begonnen die Büros des Lehrstuhls für Systemnahe Informatik und Kommunikationssysteme sowie des Lehrstuhls für Multimedia-Konzepte und Anwendungen der Universität Augsburg mit den *Smart Doorplates* auszustatten. Momentan wird weiter an der Umsetzung der vorgeschlagenen Ideen für die *Smart Doorplates* gearbeitet. Die Kontextvorhersage bildet dabei einen essentiellen Bestandteil dieser Anwendung und ist bereits als Dienst integriert.

Ich möchte mich an dieser Stelle bei Herrn Prof. Dr. Theo Ungerer für seine erstklassige Betreuung und die vielen Diskussionen bedanken, ohne die diese Arbeit nicht möglich gewesen wäre. Bei Herrn Prof. Bauer möchte ich mich für die Übernahme des Zweitgutachtens bedanken. Prof. Dr. Lucian Vintan der Universität Sibiu in Rumänien danke ich für die Zusammenarbeit und die vielen hilfreichen Diskussionen während und nach seines Aufenthaltes in Augsburg. Dank geht auch an meine Kollegen Faruk Bagci und Wolfgang Trumler für die gemeinsame Entwicklung der Idee sowie dem damit verbundenen Aufbau der *Smart Doorplates*. Weiterhin gaben sie oft konstruktive Anregungen, die die Entwicklung dieser Arbeit vorangetrieben haben. Bedanken möchte ich mich bei Frau Mirjam Kuhlmann und Herrn Andreas Pietzowski, die mit ihren Diplomarbeiten einen wertvollen Beitrag zu dieser Arbeit geleistet haben. Meiner Mutter Gerlinde Petzold, meiner Schwester Ines Kühn sowie meiner Frau Linda Petzold danke ich für das sorgfältige Korrekturlesen.

Augsburg, August 2005 *Jan Petzold*

Inhaltsverzeichnis

1 Einführung

Computersysteme begegnen uns heute in allen denkbaren Situationen. Die meisten elektronischen Geräte werden mittels Mikrocontroller gesteuert. Selbst einfachste Küchengeräte enthalten schon einen Mikroprozessor. Auch in der Automobilindustrie sind Computer nicht mehr wegzudenken, hier werden die neuesten Fahrzeuge mit einer Vielzahl von Mikroprozessoren ausgestattet. Die zukünftige Entwicklung wird sich noch weiter in diese Richtung bewegen. Die Menschen werden von einer Vielzahl von Computern umgeben sein, die zum größten Teil nicht sichtbar sind. Diese Generation der Computersysteme werden mit dem Begriff ubiquitäre Systeme bezeichnet.

Diese allgegenwärtigen Computersysteme sind eine Erweiterung der eingebetteten Systeme. Sie zeichnen sich durch den Einsatz neuer tragbarer Geräte aus, welche drahtlos untereinander und mit Geräten in der Umgebung vernetzt sind. Besonders wichtig ist das Zusammenspiel vieler Sensoren, über die Informationen aus der natürlichen Umgebung der Geräte einbezogen werden können. Aus diesen Informationen wird durch Erfassung, Interpretation, Speicherung, Austausch und Verbindung der Sensordaten Umgebungswissen erstellt, welches ubiquitären Systemen erlaubt, sich in hohem Maße auf den Menschen einzustellen. Dieses Umgebungswissen der Geräte bzw. des Systems wird als Kontext bezeichnet. Eine weitere Fähigkeit der Systeme ist, dass die Geräte Informationen in Abhängigkeit vom jeweiligen Aufenthaltsort auswählen und anzeigen können, d.h. die Geräte passen sich in ihrem Verhalten der jeweiligen Umgebung an.

In dieser Arbeit soll der Einsatz von Mechanismen zur Kontextvorhersage in ubiquitären Systemen untersucht werden. Die Kombination verschiedenster bekannter Informationen soll dazu benutzt werden, aus einem augenblicklichen Kontext heraus zukünftige Kontexte vorherzusagen. Ziel ist es, ein lernfähiges System zu entwickeln, welches sich Fehlschätzungen merkt und seine Trefferwahrscheinlichkeit erhöhen kann. Die Fähigkeit Kontexte vorherzusagen, eröffnet im Bereich der ubiquitären Systeme Möglichkeiten, dem Menschen Entscheidungen abzunehmen, die aus Gewohnheiten resultieren. Kontextvorhersage kann damit künftig eine unerlässliche Unterstützung im täglichen Leben bieten.

Als mögliche Vorhersagetechniken fällt das Augenmerk auf Neuronale Netze, Bayessche Netze sowie Markov-Modelle. Diese Verfahren sollen anhand ausgewählter Kriterien mit einer neuen Technik zur Kontextvorhersage verglichen

werden, die im Zentrum dieser Arbeit steht. Die so genannten Zustandsprädiktoren sind durch Methoden der Sprungvorhersage aus dem Bereich der Prozessorarchitektur motiviert. Zur Bewertung der Verfahren werden zwei Benchmark-Sets verwendet. Die *Augsburg Indoor Location Tracking Benchmarks* wurden im Rahmen des *Smart Doorplate* Projektes erstellt und bestehen aus Bewegungsabläufen von Personen in einem Bürogebäude. Ein weiteres Benchmark-Set stellen die *Nokia Context Daten* dar. Diese Daten beinhalten Sequenzen von verschiedenen Kontexten, wie zum Beispiel die Bewegungsabläufe einer Person außerhalb von Gebäuden sowie eine Folge von Aktivitäten dieser Person.

Im nächsten Kapitel werden die Begriffe Kontext, Kontexterkennung und Kontextvorhersage definiert, sowie eine Abgrenzung zwischen der Kontexterkennung und der Kontextvorhersage dargestellt. Kapitel 3 beschreibt die Anwendung der *Smart Doorplates* und die Integration der Kontextvorhersage in diese Anwendung. Da die *Smart Doorplates* auch als Testumgebung für die Evaluierung dienen, werden hierbei die verwendeten Benchmarks vorgestellt. Kapitel 4 fasst die grundlegenden Techniken wie Sprungvorhersage, Markov-Ketten und *Prediction by Partial Matching* zusammen, die als Motivation für die Entwicklung der Zustandsprädiktoren dienten. Die Klasse der Zustandsprädiktoren selbst wird in Kapitel 5 vorgestellt und mit den verwandten Markov-Prädiktoren verglichen. Kapitel 6 zeigt verschiedene Möglichkeiten auf, die entwickelten Prädiktoren bezüglich ihrer Zuverlässigkeit zu bewerten. In Kapitel 7 werden dann mehrere Möglichkeiten beschrieben, wie mittels der Verwendung von Hybridverfahren Verbesserungen der einzelnen Prädiktoren erreicht werden können. Allein die Vorhersage eines Kontextes genügt in den meisten Anwendungen nicht. Vielmehr ist die Zeit von Interesse, zu der ein Kontextwechsel stattfindet. Kapitel 8 stellt diesbezüglich eine einfache Möglichkeit zur Vorhersage des Zeitpunktes eines Kontextwechsels vor. Ein Vergleich der Zustands- und Markov-Prädiktoren mit anderen Verfahren zur Kontextvorhersage wie dynamischen Bayesschen Netzen und Neuronalen Netzen wird in Kapitel 9 vorgenommen. Die Evaluierung wird parallel zur Entwicklung der Zustandsprädiktoren durchgeführt, d.h. die einzelnen Kapitel enthalten die Bewertungen der in ihnen vorgestellten Techniken. Die Arbeit endet mit einer Zusammenfassung und einem Ausblick über mögliche Erweiterungen der Zustandsprädiktoren sowie weiterreichenden Ideen bezüglich der Kontextvorhersage in ubiquitären Systemen.

2 Kontextvorhersage

Bevor die Begriffe Kontexterkennung und Kontextvorhersage näher beleuchtet und abgegrenzt werden können, muss zunächst hergeleitet werden, was unter Kontext in ubiquitären Systemen zu verstehen ist.

2.1 Kontext in ubiquitären Systemen

Der Begriff Kontext spielt in vielen Bereichen der Informatik eine große Rolle. Was bedeutet nun Kontext in Zusammenhang mit ubiquitären Systemen? Dazu soll zunächst eine geeignete Definition gefunden werden.

2.1.1 Kontextdefinition

Oft wird der Begriff Kontext in erster Linie nur für die Beschreibung der physischen Umgebung eines Benutzers oder Systems verwendet. In diesem Fall wird dann vom physischen Kontext (*physical context*) gesprochen [DRD+00]. Dieser kann mittels verschiedener Sensoren erfasst werden, wobei bereits ein einzelner Sensorwert (z.B. eine bestimmte Beleuchtungsstärke) einen Kontext darstellen kann [Bei02]. Weitere Beispiele sind die Temperatur, die Luftfeuchtigkeit oder die Beschleunigung. Zusätzlich kann ein situationsbezogener Kontext (*situational context*) betrachtet werden [GSB02, SBG99]. Hiermit wird die Situation beschrieben, in der sich der Benutzer oder das System augenblicklich befinden. Diese Situationen können meist nicht durch die reine Erfassung physischer Daten ermittelt werden, sondern es müssen verschiedene Sensordaten analysiert und kombiniert werden. Beispiele für situationsbezogene Kontexte sind Tätigkeiten wie Laufen oder Schlafen, welche aus der Kombination von Herzfrequenz, Beschleunigung des Körpers und Lage des Körper ermittelt werden könnten. Ein situationsbezogener Kontext kann auch der aktuelle Aufenthaltsort einer Person sein. Zum Beispiel könnte mittels der Helligkeit, des Luftdrucks und der Umgebungsgeräusche ermittelt werden, ob sich die Person im Freien oder in einem geschlossenen Raum aufhält. Auf die Erfassung, in welchem situationsbezogenen Kontext sich ein System oder Benutzer befindet, wird im nächsten Abschnitt unter dem Begriff der Kontexterkennung näher eingegangen. Diese Erklärungen

verdeutlichen wie schwer es ist, mit dem Begriff Kontext im Bereich der ubiquitären Systeme umzugehen.

Im Forschungsbereich, der sich mit Kontext in ubiquitären Systemen bzw. Kontextbewusstsein (*Context Awareness*) befasst, sind verschiedene Definitionen von Kontext zu finden. Einen guten Überblick über die Vielzahl der Definitionen gibt die Arbeit von Dey und Abowd [DA00]. Sie unterscheiden zunächst verschiedene Arten von Definitionen: Definitionen anhand von Beispielen, Synonyme für Kontext und operative Definitionen. Ihr Ziel ist es, eine Definition zu finden, mit der es einer Anwendung möglich ist, mit Kontext umzugehen. Die Definitionen anhand von Beispielen sowie Synonyme für Kontext erlauben dies nicht. Die operativen Definitionen erfüllen nach Dey und Abowd diese Anforderung.

Im Folgenden werden einige Definitionen anhand von Beispielen aufgeführt. Shilit und Theimer [ST94] bezeichnen Kontext als Ort, Identität der in der Nähe befindlichen Personen und Objekte, sowie Änderungen bezüglich dieser Objekte. Ähnlich definieren Brown et al. [BBC97] den Kontext eines Benutzers als seinen Aufenthaltsort, die Tageszeit, die Jahreszeit, die Temperatur usw. Ryan et al. [RPM98] definieren Kontext als den Aufenthaltsort des Benutzers, die Umgebung, die Identität und die Zeit. Dey [Dey98] spezifiziert Kontext als den emotionalen Zustand des Benutzers, den Grad der Aufmerksamkeit, den Ort und die Ausrichtung, Datum und Uhrzeit, Objekte und Menschen in der Umgebung des Benutzers.

Eine Definition, die nur ein Synonym für Kontext bietet, ist von Brown [Bro96]. Er sagt Kontext sind die Elemente der Umgebung eines Benutzers, die der Computer des Benutzers kennt. Franklin und Flachsbart [FF98] definieren Kontext als die Situation des Benutzers. Ward et al. [WJH97] betrachten Kontext als den Zustand der Umgebung der Anwendung. Rodden et al. [RCDD98] definieren Kontext als die Anwendungseinstellung. Hull et al. [HNBR97] sehen Kontext als Aspekte der aktuellen Situation.

Die folgenden drei operativen Definitionen bieten nach Dey und Abowd am ehesten einer Anwendung die Möglichkeit mit Kontext umzugehen. Schilit et al. [SAW94] unterteilen Kontext in drei wichtige Bestandteile: wo bist du, wer ist bei dir und welche Ressourcen befinden sich in der Nähe. Da auch Dinge außer dem Ort mobil sind und sich ändern, ist Kontext mehr als nur der Ort. Kontext kann zum Beispiel die Helligkeit, den Lautstärkepegel, die Netzwerkverbindung, die Kommunikationskosten, die Kommunikationsbandbreite und sogar eine soziale Situation (z.B. dein Chef oder dein Kollege ist bei dir) enthalten. Dey et al. [DAW98] definieren Kontext als physischen, sozialen, emotionalen und informationellen Zustand des Benutzers. Pascoe [Pas98] definiert Kontext als Teilmenge des physischen und konzeptionellen Zustandes, welcher im Interesse der jeweiligen Entität liegt.

Eine weitere Definition, die zu dieser Definitionsart zählt, ist im *Context Toolkit*

zu finden. Salber et al. [SDA99] definieren Kontext als Umgebungsinformation, die ein Teil der Betriebsumgebung einer Anwendung ist und die durch die Anwendung abgefragt werden kann. Dey und Abowd selbst geben folgende Definition an [DA00]:

> Kontext ist jede Information, die benutzt werden kann, um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, ein Platz oder ein Objekt, die für die Interaktion zwischen einem Benutzer und einer Anwendung relevant sind. Der Benutzer und die Anwendung sind dabei eingeschlossen.

Diese Vielzahl von Definitionen zeigen, dass es schwer ist, eine allgemeine Definition für Kontext zu finden. Die genannten Definitionen sind entweder sehr spezialisiert, wie die Definitionen an Beispielen zeigen, oder zu oberflächlich, wie an den Synonymen zu sehen war. Ein guter Mittelweg wurde mit den operativen Definitionen gefunden.

In dieser Arbeit soll eine operative Kontextdefinition verwendet werden, die leicht modifiziert schon in einer anderen Arbeit [Pet03] vorgestellt wurde.

Definition 2.1 *Ein Kontext in ubiquitären Systemen beschreibt den Zustand, der durch Auswertung von Informationen aus dem Umfeld eines Benutzers oder Systems entsteht.*

Das Umfeld kann dabei die physische Umgebung des Benutzers oder Systems sein wie zum Beispiel die Temperatur, die Helligkeit oder die sich in der Nähe befindlichen Personen oder Geräte. Das Umfeld beinhaltet aber auch elektronische Informationsspeicher wie zum Beispiel einen Terminkalender. Die Zustandsdaten müssen erfasst werden, dies kann unter anderem über Sensoren geschehen.

2.1.2 Kontextkategorien

Ein weiteres Problem der vorab genannten Definitionen ist, dass diese zum Einen zu allgemein sind, d.h. der Kontext besteht aus einer unbestimmten Anzahl von Bestandteilen, von denen nicht gesagt werden kann, welcher den wichtigsten Teil ausmacht. Zum Anderen zählen die Definitionen an Beispielen Bestandteile eines Kontextes auf, die sich von Definition zu Definition unterscheiden.

Dey und Abowd [DA00] unterscheiden deshalb zunächst primäre und sekundäre Kontexttypen. Als primäre Kontexttypen nennen sie

- Ort
- Identität

- Zeit
- Aktivität

Diese primären Kategorien können benutzt werden, um speziellere Kategorien zu finden, die so genannten sekundären Kontextkategorien. Zum Beispiel kann mit Hilfe des Ortes bestimmt werden, welche anderen Personen oder Objekte sich in der Nähe befinden. Über die Identität einer Person kann zum Beispiel die Telefonnummer, die Anschrift oder die E-Mail-Adresse in Erfahrung gebracht werden. Alle Kontexte, die nicht den vier primären Kategorien entsprechen, sind sekundäre Kontexte und können mittels der primären Kontexte indiziert werden, wie zum Beispiel die Telefonnummer eines Anwenders, welche über die Identität ermittelt werden kann. Ein Beispiel, das zwei primäre Kontexte zur Indizierung benötigt, ist die Wettervorhersage, welche als Kontext in einem *Tourist Guide* verwendet wird. Hier wird über den Ort und die Zeit die passende Wettervorhersage gefunden.

2.2 Kontexterkennung

Wie die verschiedenen Definitionen von Kontext zeigen, ist Kontext nicht nur reines Erfassen von Sensorwerten. Vielmehr müssen diese gewonnen Daten ausgewertet werden. Ein Kontext wie zum Beispiel „Ein Mitarbeiter ist in einer Besprechung“ kann nicht über die reine Erfassung von Sensorwerten erkannt werden. In diesem Beispiel müsste die Ortsinformation mehrerer Mitarbeiter verglichen werden. Wenn nun zum Beispiel bei 5 Mitarbeitern der gleiche Raum festgestellt wird, kann eine Besprechung vermutet werden. Kontexterkennung bedeutet somit, dass aus verschiedenen *low-level*-Informationen, wie zum Beispiel einzelnen Sensordaten, ein augenblicklicher *high-level*-Kontext hergeleitet wird, welcher nicht direkt aus den Sensoren gewonnen werden kann [GSB02].

Für die Kontexterkennung werden Verfahren wie Neuronale Netze, Bayessche Netze und Markov-Ketten eingesetzt. Kontexterkennung mit diesen Methoden klassifiziert Kontexte nur mit einer Wahrscheinlichkeit deutlich unter 100%. Laerhoven et al. [LAL01, LC00] entwickelten ein System, welches fähig ist die Aktivitäten eines Benutzers zu erlernen, wobei nur eine minimale Unterstützung des Benutzers nötig ist. Das System verwendet den Kohonen Self-Organizing-Map Algorithmus, um aus mehreren einzelnen Sensordaten (Bewegung, Licht, Geräusche, usw.) auf die Tätigkeit und den Aufenthaltsort eines Benutzers zu schließen. Zusätzlich setzen sie ein Markov-Modell ein, welches die Wahrscheinlichkeiten für den Übergang von einem Kontext zu einem anderen Kontext angibt. Wenn der Algorithmus einen Übergang berechnet, kann das System nun mit Hilfe des Markov-Modells kontrollieren, ob die Wahrscheinlichkeit für diesen

Übergang entsprechend hoch ist, d.h. einen bestimmten Schwellenwert übersteigt. Korpipää et al. [KKP+03] untersuchten die Einsatzfähigkeit von mobilen Endgeräten zur Kontexterkennung. Mittels Bayesscher Netze wurde der Kontext des Benutzers des mobilen Gerätes ermittelt, wobei alle Sensoren am mobilen Gerät selbst angeschlossen waren. Mäntyjärvi et al. [MHH03] verwenden ebenfalls ein mobiles Gerät zur Kontexterkennung. Sie beziehen aber zusätzlich den Kontext benachbarter mobiler Geräte mit ein. Clarkson et al. [CMP00, CP98] verwenden Hidden-Markov-Modelle, um den Aufenthaltsort (wie z.B. im Büro oder in der U-Bahn) und die Aktivität (wie z.B. eine Unterhaltung) des Benutzer eines *Wearable Computer* unter Verwendung einer Kamera und eines Mikrofons festzustellen.

2.3 Kontextvorhersage

Die Kontextvorhersage geht einen Schritt weiter. In Kombination mit verschiedensten bekannten Informationen soll aus dem augenblicklichen Kontext heraus der nächste Kontext vorhergesagt werden. Als eine Form bekannter Information kann die Sequenz der vergangenen Kontexte dienen, d.h. die Kontexte werden als Zeitreihe betrachtet. Dieses Konzept wird bezeichnet als Kontextvorhersage basierend auf der Kontexthistorie und kann formell wie folgt beschrieben werden. Der Kontext c_{t+1} soll aus dem augenblicklichen Kontext c_t und den vergangenen n Kontexten $c_{t-1}, \ldots, c_{t-n}$ vorhergesagt werden. Seien $c_{t+1_1}, \ldots, c_{t+1_k}$ die möglichen nächsten Kontexte, die eintreten können, dann soll der Kontext mit der größten Wahrscheinlichkeit

$$P(c_{t+1_i}|c_t, c_{t-1}, \ldots, c_{t-n}) \quad i = 1, \ldots, k$$

bestimmt und als nächster Kontext angeboten werden.

Der Begriff Kontextvorhersage wird im Bereich der ubiquitären Systeme oft mit der Kontexterkennung gleichgesetzt. Mit der Kontexterkennung wird aber der augenblickliche Kontext ermittelt. Wogegen die Kontextvorhersage den nächsten Kontext in der Zukunft bestimmt. Eine Kombination beider Verfahren kann zum Beispiel dazu dienen, mit der Kontextvorhersage die Kontexterkennung zu unterstützen, wie Laerhoven et al. [LAL01, LC00] zeigten.

2.3.1 Anwendungsbeispiele

Im Folgenden werden einige Szenarios genannt, in denen die Kontextvorhersage sinnvoll eingesetzt werden kann.

In einem *Smart Office Building* kann die Vorhersage des nächsten Aufenthaltsortes eines Angestellten dazu genutzt werden, einen Raum vorzubereiten, indem zum Beispiel ein Rechner hoch gefahren und ein Projektor vorgewärmt werden. Weiterhin könnte die Vorhersage des nächsten Raumes eines Angestellten genutzt werden, um das Telefon des Angestellten in diesen Raum umzuleiten, bevor er diesen betritt. Folgendes Beispiel spiegelt dieses Szenario wieder:

> Herr Schmidt verlässt sein Büro. Das System sagt das Büro von Herrn Müller als nächsten Raum vorher, den Herr Schmidt betreten wird. Herr Schmidt ist nun auf dem Weg dorthin.
>
> In Herrn Müller's Büro klingelt das Telefon. Er nimmt ab und antwortet dem Anrufer: „Nein, Herr Schmidt ist nicht hier." In diesem Moment betritt Herr Schmidt das Büro und Herr Müller spricht ins Telefon: „Wie auch immer, Herr Schmidt ist jetzt hier. Ich gebe Ihnen Herrn Schmidt."

Eine weitere Anwendungsmöglichkeit der Vorhersage des nächsten Aufenthaltsortes und des Zeitpunktes des Betretens dieses Ortes besteht darin, einen Kollegen oder einen Besucher eines Angestellten über die mögliche Rückkehr des Angestellten zu informieren. Hier wäre eine Vorhersage über mehrere Schritte in die Zukunft notwendig bis die Vorhersage wieder das Büro des Angestellten ist.

In einem *Smart Office Building* könnte für einen Fahrstuhl prognostiziert werden, auf welchem Stockwerk er als nächstes benötigt wird. Dafür müssten natürlich genügend Ressourcen, d.h. mehrere Fahrstühle vorhanden sein. So könnte ein Fahrstuhl eingesetzt werden, um in einem Stockwerk bereitzustehen, in dem in den nächsten Minuten mit vielen Fahrgästen zu rechnen ist.

Außerhalb von Gebäuden kann aufgrund von Bewegungsmustern die nächste Region vorhergesagt werden, in die eine Person eintreten wird. Im Bereich der Mobilfunknetze kann eine Vorhersage dazu genutzt werden, um zu bestimmen, welche Mobilfunkzelle ein Benutzer als nächstes betritt. Dies könnte zum Beispiel für regionale Informationen wie Verkehrsmeldungen verwendet werden. Eine weitere Anwendung dieser Vorhersage wäre zum Beispiel regionale Werbung. Ein Außendienstmitarbeiter fährt durch mehrere Funkzellen und es wird eine Funkzelle prognostiziert, in der er zur Mittagszeit sein wird. Hier kann nun ein Restaurant ein Angebot für ein Mittagsmenü auf das Mobiltelefon des Mitarbeiters schicken.

2.3.2 Anforderungen

An die Kontextvorhersage müssen gewisse Anforderungen gestellt werden. Dabei werden zunächst Anforderungen bezüglich des Lernverhaltens des eingesetzten Algorithmus aufgezeigt:

- Ein Anlernen sollte in einer angemessenen Zeit geschehen. Ein Benutzer kauft sich kein neues Produkt, wenn er auf die versprochenen Vorzüge ein halbes Jahr warten muss.
- Auch das Umlernen sollte in einer akzeptablen Zeit stattfinden. Wenn das System sich an eine Gewohnheit angepasst hat, darf bei einer Änderung der Gewohnheit kein halbes Jahr vergehen bis sich die neue Gewohnheit im System etabliert hat.

Die folgenden Anforderungen beziehen sich auf den Kontext selbst:

- Betrachtet man als Kontext die Personen, die sich in der Nähe eines Objekts befinden, könnte über die Historie dieses Kontextes auch versucht werden, vorherzusagen, welche Personen sich als nächstes in der Nähe des Objekts aufhalten. Bei Betrachtung der Kontextkategorien bildet dieser Kontext einen sekundären Kontext und kann somit aus einer primären Kontextkategorie – dem Ort – der einzelnen Personen ermittelt werden. Die Vorhersage sollte sich deshalb auf die primären Kontextkategorien Ort, Identität, Zeit und Aktivität beziehen, da aus diesen die sekundären Kategorien hergeleitet werden können.
- Der Algorithmus sollte eine Art offenes System darstellen. Betrachtet man als Beispiel die benutzten Verkehrsmittel einer Person auf dem Weg ins Büro, wobei sie an manchen Tagen das Auto und an anderen Tagen öffentliche Verkehrsmittel nutzt, muss der Algorithmus fähig sein, auch ein anderes Verkehrsmittel aufzunehmen. Kauft sich diese Person plötzlich ein Fahrrad und fährt auch mit diesem manchmal ins Büro, sollte das Fahrrad als neues Verkehrsmittel registriert und eventuell vorhergesagt werden können.
- Betrachtet man die Folge eines Kontextes, kann dies eine Folge kontinuierlicher oder diskreter Werte sein. Für die Vorhersage beider Arten gibt es verschiedene Verfahren. Eine Folge kontinuierlicher Werte kann auf diskrete Werte abstrahiert werden. Deshalb sollen hier Verfahren betrachtet werden, die mit diskreten Werten umgehen können.

Neben den Anforderungen an die Kontextvorhersage gibt es noch weitere wichtige Aspekte, auf die aufmerksam gemacht werden soll. Wenn ein Kontext falsch vorhergesagt wurde, gibt es Situationen, in denen ein Rückrollen der aufgrund der Vorhersage durchgeführten Aktionen notwendig ist. Zum Beispiel wird aufgrund einer falschen Vorhersage das Licht eingeschaltet, wer schaltet es nun wieder aus? Weiterhin muss überlegt werden, wie solch ein Rückrollen mit angemessenen Strafkosten belegt werden kann.

Weiterhin existieren Abhängigkeiten verschiedener Kontexte untereinander. Wenn zwei abhängige Kontexte vorhergesagt werden, kann die Vorhersage des

einen Kontextes für die Verifikation der Vorhersage des anderen Kontextes verwendet werden. Die Verwendung verschiedener Verfahren zu einer Vorhersage kann die Vorteile der Verfahren verbinden.

2.3.3 Stand der Forschung

Vorhersagetechniken werden in vielen Gebieten der Informatik eingesetzt: Sprungvorhersage in der modernen Prozessorarchitektur, Markov-Prädiktoren im Bereich der Datenkomprimierung, Lernalgorithmen im Bereich der Künstlichen Intelligenz usw. Zur Kontextvorhersage können die selben Methoden und Verfahren wie zur Kontexterkennung eingesetzt werden. Eingesetzte Verfahren sind verschiedene statistische Methoden bzw. Verfahren des Maschinellen Lernens wie zum Beispiel Neuronale Netze, Bayessche Netze oder Markov-Ketten. Im Folgenden werden Projekte aus dem Bereich der Kontextvorhersage aufgeführt, die die genannten Verfahren einsetzen.

Im Adaptive House Projekt [Moz98] der University of Colorado wurde ein Smart House entwickelt, das die Lebensweise und die Bedürfnisse seiner Bewohner beobachtet und lernt, um sie bestmöglich zu unterstützen. Das Verhalten der Bewohner wird mit Bewegungsdetektoren erfasst und ein Neuronales Netz, speziell ein Multi-Layer-Perzeptron, wird verwendet, um den nächsten Aufenthaltsort eines Bewohners im Haus sowie dessen nächste Tätigkeit vorherzusagen.

Patterson et al. [PLFK03] von der University of Washington präsentieren eine Methode, die ein Bayessches Netz verwendet, welches aufgrund der Bewegung einer Person in einer Stadt und der aktuellen Fortbewegungsart (zu Fuß, mit Bus oder mit Auto) den nächsten Aufenthaltsort in der Stadt vorhersagt.

Ashbrook und Starner [AS03] benutzen den Ort als Kontext und bestimmen mittels eines Markov-Modells die zukünftigen Bewegungen der Benutzer. Sie schlagen ihr Modell für verschiedene Anwendungen in Szenarios mit einzelnen und mehreren Benutzern vor. Ihre Vorhersage des nächsten Aufenthaltsortes ist momentan zeitunabhängig. Bhattacharya und Das [BD02] untersuchten das Problem der Mobilität in Mobilfunknetzen. Sie setzen ein Markov-Modell ein, um die nächste Zelle vorherzusagen, in die ein Mobilfunkanwender eintreten wird. Kaowthumrong et al. [KLH02] verwenden ein Markov-Modell, um automatisch das nächste Gerät zu wählen, mit dem der Benutzer interagieren wird. Die Vorhersage beruht auf vergangenem Verhalten der Benutzer.

Mayrhofer et al. [MRF03] schlagen eine Architektur vor, die die Kontexterkennung und die Kontextvorhersage verbindet. Zur Kontexterkennung aus mehreren Sensoren verwenden sie den GNG-Algorithmus (*Growing Neural Gas*). Für die Kontextvorhersage empfehlen sie einen Markov-Prädiktor.

In Tabelle 2.1 werden die bisherigen Ansätze anhand wichtiger Eigenschaften und der erfüllten Anforderungen bewertet.

Tabelle 2.1: Vergleich bisheriger Ansätze

	Mozer	Patterson et al.	Ashbrook und Starner	Bhattacharya und Das	Kaowthumrong et al.	Mayrhofer et al.	vorliegende Arbeit
Vorhersage-technik	MLP	BN	MM	MM	MM	MP	ZP
Anlern-verhalten	lang-sam	schnell	schnell	schnell	schnell	schnell	schnell
Umlern-verhalten	lang-sam	lang-sam	lang-sam	lang-sam	lang-sam	lang-sam	schnell
Vorhersage Ort	ja	nein	ja	ja	nein	nein	ja
Vorhersage Identität	nein	nein	nein	nein	nein	nein	nein
Vorhersage Zeit	nein	nein	nein	nein	nein	nein	ja
Vorhersage Aktivität	ja	ja	nein	nein	ja	nein	ja
Offenes System	nein	ja	ja	ja	ja	ja	ja
Speicher-aufwand	-	-	-	-	-	-	gering
Rechen-aufwand	hoch	hoch	gering	gering	gering	gering	gering
Vergleich mit anderen Verfahren	nein	nein	nein	nein	nein	nein	ja

Fazit: Alle bisherigen Ansätze modellieren nur für eine spezielle Anwendung einen Vorhersagealgorithmus. Es gibt keinen Vergleich der verschiedenen Methoden unter dem gleichen Modell. Systematische Untersuchungen zu Speicher- und

Rechenaufwand wurden nicht vorgenommen. Weiterhin sind keine Untersuchungen zur Dynamik bzw. Geschwindigkeit des Umlernens zu finden. Ein Problem bei Markov-Modellen ist zum Beispiel das unter Umständen sehr langsame Umlernen einer Gewohnheit. Außerdem betrachtet keiner der Ansätze die Zeit, zu der ein Kontext eintreten wird.

2.4 Lösungsansatz

Abbildung 2.1 stellt dar, wie eine kontextbewusste Anwendung durch Kontextvorhersage erweitert werden kann. Die Anwendung zeichnet über ein entsprechendes System zu festgelegten Zeitpunkten den aktuellen Kontext auf. Daraus ergibt sich dann die Kontexthistorie, aus welcher mit Hilfe der Kontextvorhersage der nächste mögliche Kontext ermittelt werden kann. Diese Arbeit schlägt Lösungen für die Kontextvorhersage vor. Dabei wird davon ausgegangen, dass die verwendeten Kontexthistorien vorliegen.

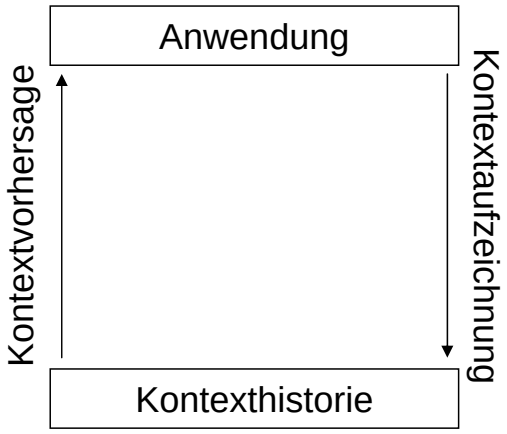


Abbildung 2.1: Überblick Kontextvorhersage

Im Hauptteil der vorliegenden Arbeit wird eine Vorhersagetechnik vorgestellt, die durch Sprungvorhersagetechniken aus dem Bereich der Prozessorarchitektur [BU02] motiviert wurde. Die Sprungvorhersagetechniken zeichnen sich durch ihre Vielfalt und ihre hohen Trefferwahrscheinlichkeiten aus. Bezüglich ihrer Vielfalt lassen sich einfache Verfahren nennen, die sehr schnell angelernt sind, und es gibt komplexere Verfahren, welche vorhandene Muster in der Folge der Sprungbefehle erkennen. Weiterhin zeichnen sich die Algorithmen der Sprungvorhersage durch ihren geringen Speicheraufwand aus. Die Funktionsweise ist meist sehr einfach, so dass kein großer Aufwand an Rechenleistung nötig ist. Diese Eigenschaft ist für die energiesparenden und langsamen ubiquitären Geräte von großer Bedeutung.

Im Folgendem werden zwei Beispiele vorgestellt, die zeigen wie die Methoden der Sprungvorhersage auf die Kontextvorhersage abgebildet werden können.

Erlernen einer einzelnen Information

Eine Person betritt einen Raum täglich zur gleichen Zeit und schaltet das Licht ein. Der Kontext ist hier die Aktivität *Licht anschalten*. In der Kontexthistorie steht nun am Ende mehrmals der Kontext *Licht angeschaltet*. Beim nächsten Betreten des Raumes durch diese Person wird das Licht automatisch eingeschaltet. Umgekehrt schaltet die Person nun beim Betreten mehrmals das Licht hintereinander wieder aus, soll auch das Licht beim nächsten Betreten nicht mehr angeschaltet werden. Eine Lösung stellt der *Zwei-Bit-Prädiktor mit Sättigung* [BU02] dar.

Erkennen von Bewegungsabläufen

Ein Angestellter folgt immer gleichen Verhaltensmustern. Morgens betritt er immer zuerst sein Büro, anschließend geht er in die Küche, um Kaffee zu kochen, dann geht er in das Sekretariat, um nach Post zu schauen. Schließlich geht er wieder in sein Büro. Um solche Muster zu erkennen und die nächste Aktivität vorherzusagen, kann die Idee eines *Zweistufig adaptiven Prädiktors* [BU02] als Lösung dienen.

3 Anwendung

Die *Smart Doorplates* beschreiben eine Anwendung, in der die Kontextvorhersage basierend auf Kontexthistorie eingesetzt wird. Die Anwendung der *Smart Doorplates* bildet auch die Grundlage der Evaluierung der entwickelten Verfahren.

3.1 Smart Doorplates

Ubiquitäre Systeme sind alltägliche Gegenstände, die mit integrierter Rechenleistung den Benutzer in seinem Handeln unterstützen. In diesem Zusammenhang ist es von besonderer Bedeutung, dass die zusätzlichen technischen Bestandteile nicht als Hindernis, sondern als Hilfe vom Benutzer wahrgenommen werden. Daraus entsteht die Forderung nach fließender Integration und nach Umsetzung in intuitiv bedienbare Geräte.

Ein Beispiel für die Erweiterung eines alltäglichen Gegenstandes um technische Funktionalitäten und der daraus resultierenden komfortablen Benutzung und der Schaffung eines deutlichen Mehrwertes ist die Vision der intelligenten Türschilder oder *Smart Doorplates* in einem Bürogebäude [TBPU03]. Ein *Smart Doorplate* soll ähnlich einer semitransparenten Tür die augenblickliche Arbeitssituation eines Mitarbeiters durchschimmern lassen (sitzt am Rechner, telefoniert, ist in einer Besprechung), wobei die nach außen transportierten Aktionen von dem Mitarbeiter selbst bestimmt werden können. Im Falle der Abwesenheit soll das Türschild in begrenztem Maße an Stelle des Mitarbeiters automatisiert handeln und Sekretariatstätigkeiten vornehmen (Nachrichten entgegennehmen/weiterleiten, Aufenthaltsort und Rückkehrzeit mitteilen).

In [TBPU03] werden eine Vielzahl von Szenarios beschrieben, in denen die *Smart Doorplates* Nutzen finden. In den *Smart Doorplates* wird die Kontextvorhersage bei der Abwesenheit eines Mitarbeiters eingesetzt. Kommt ein Besucher an das Büro eines Mitarbeiters und dieser ist nicht anwesend, zeigt das Türschild dem Besucher den momentanen Aufenthaltsort des Mitarbeiters an, sofern dieser sich im Gebäude befindet und von einem *Location Tracking System* erfasst werden kann. Möchte der Besucher dem Mitarbeiter folgen, ist die Information von Bedeutung, ob der Mitarbeiter in naher Zukunft in einen anderen Raum geht. So kann der Besucher, falls er den Mitarbeiter nicht mehr in dem angezeigten Raum findet, ihm in den vorhergesagten Raum folgen.

Weiterhin ist die Vorhersage des nächsten möglichen Aufenthaltsortes auch für die Entscheidung eines Besuchers wichtig, ob er auf den Mitarbeiter am Büro warten soll oder nicht. In diesem Fall muss die Vorhersage über mehrere Räume bis zum eigenen Büro des Mitarbeiters geschehen. Für beide Varianten ist eine Vorhersage des Zeitpunktes unerlässlich, zu dem der nächste Wechsel des Aufenthaltsortes bzw. die Ankunft im eigenen Büro stattfindet. Eine weitere Anwendung der Kontextvorhersage ist die Telefonweiterleitung. Hierbei soll bestimmt werden, in welchen Raum ein Mitarbeiter nach Verlassen eines Büros gehen wird, um sein Telefon in das prognostizierte Büro umzuleiten.

Abbildung 3.1: Anzeige *Smart Doorplate*

Abbildung 3.1 zeigt ein *Smart Doorplate*, welches Informationen wie ein traditionelles statisches Türschild anzeigt. Die Informationen sind unter anderem die Raumnummer, die Namen der Angestellten und die Sprechstunde. Zusätzlich zeigt ein Türschild mittels eines kleinen Symbols vor dem Namen des Angestellten an, ob dieser anwesend ist. Ist der Angestellte abwesend, kann ein Kollege oder Besucher durch Anklicken des Namen mehr Informationen über diesen erhalten. In Abbildung 3.2 ist die Anzeige mit diesen Informationen zu sehen. Dem Kollege oder Besucher wird der aktuelle Aufenthaltsort und die Richtung angezeigt, in der sich dieser befindet. Weiterhin wird mitgeteilt, welches der mögliche nächste Raum und die Zeit bis zum Eintritt in diesen ist.

Aktuell wird ein Vorhersagedienst zentral für alle Angestellten verwendet. Dies entspricht aber nicht der Vorstellung eines dezentralen Systems, wie es die *Smart Doorplates* darstellen. Ideal wäre ein Vorhersagedienst, der auf einem persönlichen tragbaren Gerät (z.B. PDA, Handy) oder auf dem persönlichen Rechner jedes Angestellten installiert ist. So hat zunächst jeder Angestellte die Möglichkeit zu entscheiden, ob er den Dienst anbieten möchte oder nicht. Auf diesem Gerät wird dann auch der Vorhersagealgorithmus ausgeführt. Eine Voraussetzung dafür ist, dass die Kontexthistorie auf diesem persönlichen Gerät gesammelt wird. Diese Kapselung der persönlichen Informationen auf dem eigenen Gerät

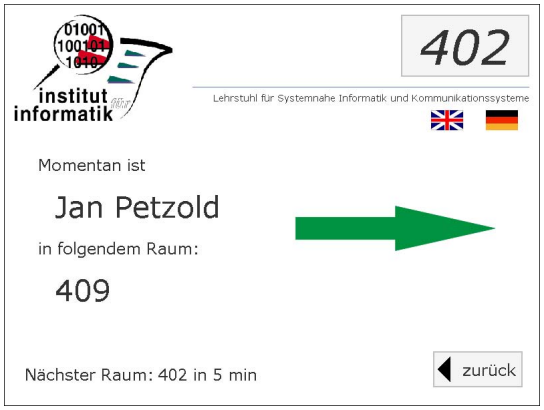


Abbildung 3.2: Erweiterte Anzeige *Smart Doorplate*

des Angestellten bewahrt das System auch vor eventuellen Problemen im Umgang mit personenbezogenen Daten. Weiterhin können durch die Einbeziehung eines persönlichen Gerätes sichere Daten über die Person wie Kalendereinträge zur Verbesserung der Vorhersage auf einfache Weise integriert werden.

3.2 Augsburg Benchmarks

Die so genannten *Augsburg Indoor Location Tracking Benchmarks* (im Folgenden verkürzt als Augsburg Benchmarks bezeichnet) beinhalten Bewegungssequenzen von vier Personen durch ein Bürogebäude. Die Bewegungsdaten wurden von Juli 2003 bis Januar 2004 am Institut für Informatik der Universität Augsburg aufgezeichnet [Pet04].

In der ersten Entwicklungsphase der Zustandsprädiktoren wurde die Evaluierung mit synthetischen Bewegungsabläufen durchgeführt. Das Problem von synthetischen Daten besteht nun darin, dass diese immer ein perfektes Szenario darstellen. Da die entwickelten Prädiktoren aber in einer realen Anwendung eingesetzt werden sollten, musste dieses Problem beseitigt werden. Dazu sollten entsprechend dem späteren Einsatz in den *Smart Doorplates* die Bewegungen mehrerer Personen durch ein Bürogebäude aufgezeichnet werden.

3.2.1 Akquirierung

Die Messungen der Bewegungen wurden manuell durchgeführt, d.h. es wurde kein automatisches *Location Tracking System* verwendet. Deshalb wurde ein kleines Programm mit einfacher Benutzeroberfläche für einen PDA implementiert, welches den vierten Stock des Institutsgebäudes zeigt, auf dem sich der Lehrstuhl

für Systemnahe Informatik befindet (siehe Abbildungen 3.4 und 3.3). Alle vier Testpersonen wurde mit einem PDA ausgestattet. Falls eine Person einen neuen Raum betrat, musste sie auf die entsprechende Bezeichnung auf dem PDA klicken. Bei jedem Klick speicherte das Programm das Datum und die Uhrzeit des Eintritts, den Namen des neuen Raumes, den Namen der Person sowie den Zeitpunkt des Eintritts in Millisekunden, welcher später für Berechnungen der Aufenthaltsdauer in einem Raum verwendet werden kann.

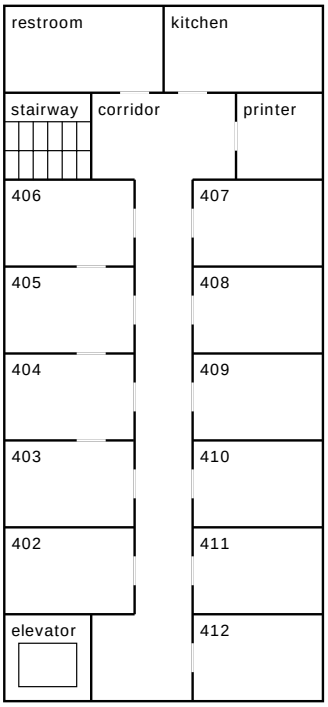


Abbildung 3.3: Grundriss vierter Stock

Abbildung 3.4: Oberfläche PDA

Ein allgemeines Problem bei dieser manuellen Messung war, dass die Personen oft vergaßen den PDA mitzunehmen oder zu klicken, wenn sie einen neuen Raum betraten. In diesem Fall holten sie die Eingaben zu einem späteren Zeitpunkt soweit sie sich erinnern konnten nach, oder sie vergaßen selbst dies zu tun. Aus diesem Grund sind die Zeitangaben nicht immer korrekt, und die Bewegungssequenzen sind manchmal unvollständig.

Das Experiment war in zwei Teile unterteilt, die Sommer- und die Herbstmessungen. Im Sommer 2003 wurde die Messungen über zwei Wochen durchgeführt. Im Herbst deckte das Experiment einen längeren Zeitraum ab, hier wurden die Messungen zwischen vier und neun Wochen durchgeführt.

3.2.2 Struktur

Das Experiment wurde von vier Personen durchgeführt, die mit A, B, C und D benannt sind. Für jede Person gibt es zwei Dateien, eine für die Sommerdaten und eine für die Herbstdaten, d.h. die Benchmarks beinhalten insgesamt acht Dateien:

```
a_summer.data
a_fall.data
b_summer.data
b_fall.data
c_summer.data
c_fall.data
d_summer.data
d_fall.data
```

Jede Zeile der Dateien enthält die Daten für den Eintritt in einen Raum. Eine Zeile ist wie folgt aufgebaut:

```
yyyy.mm.dd hh.mm.ss;location;person;timestamp
```

Die einzelnen Werte sind durch Semikolons getrennt. Der erste Wert ist der Zeitpunkt des Eintritts der Person. Er besteht aus dem Datum und der Uhrzeit im folgenden Format: Jahr mit vier Ziffern, Monat mit zwei Ziffern und Tag mit zwei Ziffern, Leerzeichen als Trennung zwischen Datum und Uhrzeit, Stunde, Minute und Sekunde mit je zwei Ziffern.

Der nächste Wert ist der Raum, welchen die Person betreten hat. Dann kommt die ID der Person. Der letzte Wert ist der Zeitpunkt des Eintritts in den Raum. Dieser Wert beschreibt die Zeit seit 01.01.1970 in Millisekunden.

Als Beispiel betrachten wir die Testperson B. Sie betritt die Küche am 16. Juli 2003 um 14:27:59 Uhr. Den entsprechenden Eintrag zeigt die folgende Zeile.

```
2003.07.16 14:27:59;kitchen;B;1058358479907
```

Die Benchmarks enthalten die Räume aus Tabelle 3.1. In der zweiten Spalte ist eine kurze Beschreibung wichtiger Räume zu finden. Der vierte Stock kann über das Treppenhaus (*stairway*) sowie den Fahrstuhl (*elevator*) verlassen werden (siehe Abbildung 3.3). Diese zwei Optionen sind in *away* kombiniert.

Tabelle 3.2 fasst die Daten der Benchmarks kurz zusammen. Jede Datei enthält die Bewegungsdaten einer einzelnen Person, die in der zweiten Spalte angegeben ist. Die Zeitspanne gibt an, über wie viel Wochen die Messungen für diese Datei

Tabelle 3.1: Räume im vierten Stock

Raum	Beschreibung
402	Büro von Person A und B
403	Büro von Person D
404	Sekretariat
405	
406	
407	
408	
409	Besprechungsraum
410	
411	
412	Büro von Person C
corridor	
printer	
kitchen	
restroom	
away	Person ist nicht im vierten Stock

durchgeführt wurden. Der Eintrag in der letzten Zeile ist in zwei Teile getrennt, da diese Messung durch zwei Wochen unterbrochen wurde. Die Anzahl der Bewegungen gibt die Anzahl der betretenen Räume durch die Person während des Experiments an. Der Vollständigkeit halber ist in der letzten Spalte das eigene Büro der zugehörigen Person angegeben.

Tabelle 3.2: Zusammenfassung Augsburg Benchmarks

Datei	Person	Zeitspanne	Bewegungen	Büro
`a_summer.data`	A	eine Woche	101	402
`a_fall.data`	A	vier Wochen	432	402
`b_summer.data`	B	zwei Wochen	448	402
`b_fall.data`	B	fünf Wochen	982	402
`c_summer.data`	C	zwei Wochen	351	412
`c_fall.data`	C	vier Wochen	911	412
`d_summer.data`	D	zwei Wochen	158	403
`d_fall.data`	D	sieben + zwei Wochen	848	403

3.3 Nokia Context Daten

Für die *Smart Doorplates* wird die Vorhersage des nächsten Aufenthaltsortes in Gebäuden betrachtet. Auch außerhalb von Gebäuden gibt es einige Anwendungen, in denen eine Vorhersage des nächsten Aufenthaltsortes bzw. der nächsten Region benötigt wird. Dies kann zum Beispiel eine Anwendung sein, die dem Benutzer Verkehrsmeldungen für die Region liefert, in die er als nächstes mit seinem Fahrzeug hineinfahren wird. Eine weitere Anwendung ist regionale Werbung. Ein Restaurant aus einer Region, die der Benutzer zur Mittagszeit durchqueren wird, kann ihm zum Beispiel ein Angebot für ein Mittags-Menü schicken.

Eine weitere Anwendung, in der wiederkehrende Muster für die Vorhersage verwendet werden könnten, wäre die Folgende: Ein Angestellter fährt von Montag bis Freitag immer in sein Büro, am Wochenende fährt er in sein Wochenendhaus. Aufgrund dieser Regelmäßigkeit und zusätzlicher Informationen wie zum Beispiel Ferienbeginn und damit Stau auf der bevorzugten Strecke zum Wochenendhaus des Angestellten, könnte sein Mobiltelefon schon am Freitag eine alternative Route ermitteln. Für diese andere Strecke benötigt er aber eine Stunde länger. So kann ihm sein Mobiltelefon dies schon am Freitagabend mitteilen. Dann kann er, wenn er zur gleichen Zeit wie immer ankommen möchte, eine Stunde früher abfahren.

Zunächst muss für diese Anwendungen eine Unterteilung in Regionen gefunden werden. Dazu eignet sich besonders gut der Mobilfunk. Mobilfunknetze bestehen aus Zellen, deren Größe bei GSM von 500 m (Stadt) bis 35 km (ländliches Gebiet) reicht. Diese Zellen sind wiederum zu Gebieten (Location Areas) zusammengefasst. Die *Nokia Context Daten* [Fla04] bieten diese beiden Informationen.

Die Nokia Context Daten bestehen aus 43 verschiedenen aufgezeichneten Sessions. In jeder Session führte der selbe Benutzer ein Mobiltelefon, eine Sensorbox und einen Laptop mit sich und ging von zu Hause zu seinem Arbeitsplatz und umgekehrt. Bei Gelegenheit wählte er leicht oder komplett andere Wege sowie verschiedene Arten der Fortbewegung (zu Fuß, Bus, U-Bahn, Auto). Während jeder Session wurde mit der Sensorbox die Bewegungsgeschwindigkeit, der Luftdruck, die Temperatur, die Luftfeuchtigkeit, die Umgebungsgeräusche gemessen. Mit dem Mobiltelefon wurde der Aufenthaltsort mittels der Zell-ID (CID) sowie des *Location Area Code* (LAC) aufgezeichnet. Weiterhin wurde die Benutzerinteraktion mit dem Mobiltelefon wie Anrufe, SMS, WAP usw. sowie der Zeitpunkt der Interaktion gemessen. Nach jeder Session wurden die aufgezeichneten Daten bearbeitet und bestimmte Merkmale extrahiert wie zum Beispiel die Aktivität des Benutzers. Für jede Session gibt es zwei Dateien. In der ersten Datei sind die folgenden Merkmale zu jeder Sekunde gespeichert:

Zeitstempel
Wochentag
Datum

Monat
Tageszeit
Stunde
Halbe Stunde
Zell-ID (CID)
Location Area Code (LAC)
Temperatur
Relative Luftfeuchtigkeit
Taupunkt
Luftdruck
Orientierung
Aktivität des Benutzers
Durchschnittliches Lautstärkepegel

Die zweite Datei speichert die Interaktion des Benutzers mit dem Terminal während jeder Session. Dabei werden drei Interaktionen unterschieden: Anwendungen wie ein Spiel, Zugriff auf eine Internetseite und Starten einer Kommunikation wie SMS.

Die 43 Sessions der Nokia Context Daten bestehen aus insgesamt 212719 Datensätzen. Diese Zahl kommt zustande, da für jede Sekunde die Merkmale angegeben sind. Bei einer Vorhersage der nächsten Mobilfunkzelle, die der Benutzer betritt, werden nur die Datensätze benötigt, bei denen sich die Zell-ID ändert. In den vorliegenden Daten wechselt der Benutzer 2317 mal die Zelle, bei 125 verschiedenen Zellen. Wenn ein Benutzer eines Mobiltelefons sich auf der Grenze zwischen zwei Zellen befindet, tritt das Problem auf, dass die registrierte Zell-ID zwischen den IDs der beiden Zellen pendeln kann, obwohl das Mobiltelefon sich nicht in Bewegung befindet, also stationär ist. Da mehrere Zellen zu einem Gebiet zusammengefasst sind, ist die Gebietsangabe (Location Area Code) grobkörniger als die Zellenangabe (Zell-ID). In den vorliegenden Daten wechselt der Benutzer 207 mal das Gebiet bei 6 verschiedenen Gebieten. Nach einer Analyse der Daten wurde festgestellt, dass es sich bei den Zell-IDs um globale IDs handelt, d.h. eine Zell-ID kann nicht in zwei verschiedenen Location Areas auftreten.

Diese beiden Ortsinformationen, zum einen die feinkörnigen Mobilfunkzellen und zum anderen die grobkörnigen Gebiete, bieten die Möglichkeit einer Untersuchung bezüglich der Fein- bzw. Grobkörnigkeit eines eingesetzten Vorhersageverfahrens.

Weiterhin bildet auch die Benutzeraktivität eine fortlaufende Sequenz von Kontexten, die ähnlich der Ortsinformation vorhersagbar ist. Eine Vorhersage der nächsten Aktivität eines Benutzers könnte eingesetzt werden, um für die vorhergesagte Aktivität benötigte Geräte zu initialisieren bzw. andere Vorbereitungen zu treffen. In den 43 Sessions der Nokia Context Daten wechselt der Benutzer 2092 mal seine Aktivität. Die Aktivität in den Nokia Context Daten spiegelt nur

die Bewegungsgeschwindigkeit des Benutzers wieder. Hierbei wurde die Geschwindigkeit in fünf Stufen unterteilt. Dabei bedeutet die Stufe 1, dass die Person sich nicht bewegt, also ruht, bei Stufe 5 fährt die Person in einem Auto mit zügiger Geschwindigkeit. In den Daten wurden 6 verschiedene Stufen gefunden im Gegensatz zur Beschreibung, wo nur von 5 Stufen die Rede ist.

4 Grundlegende Techniken

In diesem Kapitel werden die grundlegenden Techniken vorgestellt, welche die Zustandsprädiktoren motiviert haben. Dabei werden verschiedene Verfahren der Sprungvorhersage, die Idee der Markov-Ketten sowie der Algorithmus *Prediction by Partial Matching* beschrieben.

4.1 Sprungvorhersage

Für heutige und zukünftige Mikroprozessoren ist eine exzellente Behandlung von Sprungbefehlen sehr wichtig. Deshalb sind die Sprungvorhersagetechniken in den letzten 20 Jahren sehr intensiv erforscht worden.

Die Gesamtleistung einer Sprungvorhersage hängt zum Einen von der Genauigkeit der Vorhersage und zum anderen von den Kosten für das Rückrollen bei einer Fehlspekulation ab. Die Genauigkeit kann durch eine qualitativ bessere Sprungvorhersagetechnik erhöht werden. Der Einsatz größerer Informationstabellen über die bisherigen Sprungverläufe führt bei diesen Techniken auch zu weniger Fehlspekulationen [ŠRU99, BU02].

4.1.1 Dynamische Sprungvorhersagetechniken

Bei der dynamischen Sprungvorhersagetechnik wird die Entscheidung über die Spekulationsrichtung eines Sprunges in Abhängigkeit vom bisherigen Programmablauf getroffen. Die Sprungverläufe werden beim Programmablauf in Sprungverlaufstabellen gesammelt und auf der Grundlage der Tabelleneinträge werden aktuelle Sprungvorhersagen getroffen. Bei Fehlspekulationen werden die Tabellen per Hardware abgeändert.

Als dynamische Sprungvorhersagetechniken kommen heute neben dem einfachen Zwei-Bit-Prädiktor zunehmend zweistufig adaptive Prädiktoren und Hybrid-Prädiktoren zum Einsatz. Diese Techniken werden in den nachfolgenden Abschnitten beschrieben (siehe auch [ŠRU99, BU02]).

4.1.2 Ein- und Zwei-Bit-Prädiktoren

Die einfachste dynamische Sprungvorhersagetechnik ist der Ein-Bit-Prädiktor, der für jeden Sprungbefehl die zwei Zustände *genommen* oder *nicht genommen* in einem Bit speichert. Diese Zustände beziehen sich dabei immer auf die zeitlich letzte Ausführung des Sprungbefehls. Die Zustände eines Ein-Bit-Prädiktors sind in Abbildung 4.1 dargestellt. Dabei steht 1 für *genommen* und 0 für *nicht genommen*. Die Zustände geben die Vorhersage an.

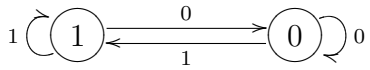


Abbildung 4.1: Ein-Bit-Prädiktor

Der Ein-Bit-Prädiktor benötigt eine Sprungverlaufstabelle, die mit einem Teil der Sprungbefehlsadresse indiziert wird und pro Eintrag ein Bit enthält, das die Sprungvorhersage bestimmt. Ist dieses Bit gesetzt, wird der Sprung als *genommen* vorhergesagt, ist es nicht gesetzt, als *nicht genommen*. Im Falle einer Fehlspekulation wird das Bit invertiert und damit die Richtung der Vorhersage umgekehrt.

Der Ein-Bit-Prädiktor sagt jeden Sprung am Ende einer Schleifeniteration richtig voraus, solange die Schleife iteriert wird. Der Prädiktor des Sprungbefehls steht auf *genommen* (1). Wird die Schleife verlassen, so ergibt sich eine falsche Vorhersage und damit eine Invertierung des Vorhersagebits des Sprungbefehls, auf *nicht genommen* (0). Damit kommt es in geschachtelten Schleifen jedoch in der inneren Schleife zu einer weiteren falschen Vorhersage. Beim Wiedereintritt in die innere Schleife steht am Ende der ersten Iteration der inneren Schleife die Vorhersage noch auf *nicht genommen* (0). Die zweite Iteration wird damit falsch vorhergesagt, denn erst ab dieser zweiten Iteration steht der Prädiktor des Sprungbefehls wieder auf *genommen* (1). Mit einem Zwei-Bit-Prädiktor wird bei geschachtelten Schleifen eine dieser zwei Fehlvorhersagen vermieden.

Beim Zwei-Bit-Prädiktor werden für die Zustandscodierungen der bedingten Sprungbefehle zwei Bits pro Eintrag in der Sprungverlaufstabelle verwendet. Damit ergeben sich die vier Zustände *sicher genommen* (11), *vielleicht genommen* (10), *vielleicht nicht genommen* (01) und *sicher nicht genommen* (00). Befindet sich ein Sprungbefehl in einem *sicheren* Vorhersagezustand, also 11 oder 00, so sind zwei aufeinander folgende Fehlspekulationen nötig, um die Vorhersagerichtung umzudrehen. Damit kommt es bei inneren Schleifen einer Schleifenschachtelung nur beim Austritt aus der Schleife zu einer falschen Vorhersage.

Es gibt zwei Ausprägungen des Zwei-Bit-Prädiktors, die sich in der Definition der Zustandsübergänge unterscheiden. Das Schema mit einem Sättigungszähler ist in Abbildung 4.2 dargestellt. Die zweite, Hysteresezähler genannte Variante verdeutlicht Abbildung 4.3.

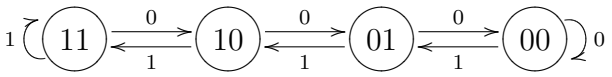


Abbildung 4.2: Zwei-Bit-Prädiktor mit Sättigungszähler

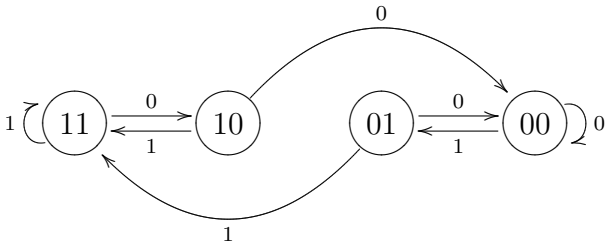


Abbildung 4.3: Zwei-Bit-Prädiktor mit Hysteresezähler

Der Zwei-Bit-Prädiktor mit Sättigungszähler erhöht jedes Mal, wenn der Sprung genommen wurde, den Zähler und erniedrigt ihn, falls er nicht genommen wurde. Durch die Sättigungsarithmetik fällt der Zähler nie unter null (00) oder wird größer als drei (11). Das höchstwertige Bit gibt die Richtung der Vorhersage an.

Die zweite Variante, die Hysteresemethode, unterscheidet sich von der des Sättigungszählers dadurch, dass direkt von einem *unsicheren* Zustand in den *sicheren* Zustand der entgegengesetzten Richtung gewechselt wird. Damit kommt man von einem *sicheren* Zustand in den anderen *sicheren* Zustand durch zwei Fehlspekulationen.

Die Technik der Zwei-Bit-Prädiktoren lässt sich leicht auf n Bits erweitern. Es zeigte sich jedoch, dass dabei für die Sprungvorhersage so gut wie keine Verbesserungen mehr erzielbar sind.

4.1.3 Korrelationsprädiktoren

Die Zwei-Bit-Prädiktoren ziehen für eine Vorhersage immer nur den Verlauf des Sprunges selbst in Betracht. Die Beziehungen zwischen verschiedenen Sprüngen werden nicht berücksichtigt. Untersuchungen haben jedoch gezeigt, dass bei Auswertung dieser Beziehungen eine bessere Sprungvorhersage durchgeführt werden kann.

Die von Pan et al. [PSR92] entwickelten Korrelationsprädiktoren berücksichtigen neben der eigenen Vergangenheit eines Sprungbefehls auch die Historie benachbarter, im Programmlauf vorhergegangener Sprünge. Korrelationsprädiktoren erzielen gewöhnlich bei ganzzahlintensiven Programmen eine höhere Treffergenauigkeit als die Zwei-Bit-Prädiktoren. Sie benötigen dabei nur wenig mehr

Hardware.

4.1.4 Zweistufig adaptive Prädiktoren

Die zweistufig adaptiven Prädiktoren wurden von Yeh und Patt [YP92] etwa zur gleichen Zeit wie die eng verwandten Korrelationsprädiktoren entwickelt. Wie der Korrelationsprädiktor ist ein zweistufig adaptiver Prädiktor aus zwei Tabellenebenen aufgebaut, wobei der Eintrag in der ersten Tabelle dazu dient, die Vorhersagebits auf der zweiten Tabellenebene zu selektieren.

Für die zweistufig adaptiven Prädiktoren gibt es neun Varianten mit einem heute oft verwendeten Benennungsschema ([YP93]), welches wie folgt aufgebaut ist: Ein zweistufig adaptiver Prädiktor wird mit $X\mathrm{A}x$ bezeichnet, wobei X für einen der Großbuchstaben G, P oder S und x für einen der Kleinbuchstaben g, p oder s steht. Die Großbuchstaben bezeichnen den Mechanismus auf der ersten Tabellenebene, A steht für adaptiv und der Kleinbuchstabe benennt die Organisationsform der zweiten Tabellenebene. Tabelle 4.1 zeigt alle neun Varianten der von Yeh und Patt [YP93] definierten zweistufig adaptiven Prädiktoren.

Tabelle 4.1: Varianten zweistufig adaptiver Prädiktoren [YP93]

	global PHT	per-set PHTs	per-address PHTs
global BHR	GAg	GAs	GAp
per-set BHT	SAg	SAs	SAp
per-address BHT	PAg	PAs	PAp

Die grundlegenden, zweistufig adaptiven Prädiktorschemata benutzen ein einziges globales Sprungverlaufsregister (BHR = *Branch History Register*) von k Bit Länge als Index, um einen Eintrag in einer Sprungverlaufstabelle (PHT = *Branch Pattern History Table*) mit Zwei-Bit-Prädiktoren zu adressieren. Das globale BHR wird nach jeder Ausführung eines (bedingten) Sprungbefehls geändert, d.h. der Inhalt des als Schieberegister organisierten BHR wird bitweise von rechts nach links geschoben und der Bitwert 1 für einen genommenen und 0 für einen nicht genommenen Sprung nach jeder Sprungentscheidung an der rechten Bitposition eingefügt. Damit wird, wie bei den Korrelationsprädiktoren, nicht nur der Verlauf eines einzelnen Sprungbefehls, sondern die Aufeinanderfolge von Sprungbefehlen für die Sprungvorhersage einbezogen. Alle zweistufig adaptiven Prädiktoren, die ein globales BHR benutzen, gehören zu den globalen Verlaufsschemata und entsprechen den Korrelationsprädiktoren.

Im einfachsten Fall gibt es ein einziges globales BHR (als G bezeichnet) und eine einzige globale PHT (als g bezeichnet). Dieser einfache Prädiktor wird GAg-Prädiktor genannt. Alle PHT-Implementierungen der zweistufig adaptiven

Prädiktoren von Yeh und Patt benutzen Zwei-Bit-Prädiktoren. Eine Implementierung eines GAg-Prädiktors mit einem 4 Bit langen BHR (deshalb als GAg(4) bezeichnet) wird in Abbildung 4.4 gezeigt.

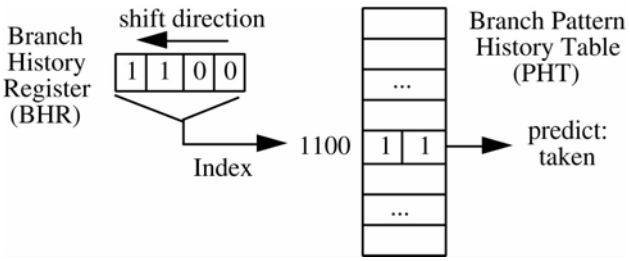


Abbildung 4.4: Implementierung eines GAg(4)-Prädiktors [BU02]

Ein Problem bei der Verwendung des GAg-Prädiktor ist, dass sich an verschiedenen Stellen des Programms das gleiche Bitmuster im BHR einstellen kann. Damit wird der selbe Eintrag im PHT adressiert, womit PHT-Interferenzen entstehen, d.h. Sprungbefehle, die nichts gemeinsam haben, beeinflussen den PHT-Eintrag. Dieses Phänomen kann durch Erweiterungen in der zweiten Tabellenebene bei Beibehaltung des globalen BHR geschwächt bzw. verhindert werden.

Zum Einen kann zusätzlich zum Bitmuster eines einzigen globalen BHR die gesamte Sprungbefehlsadresse zur PHT-Adressierung genutzt werden. Dies kommt letztlich einer eigenen PHT pro Sprungbefehl gleich und schließt Interferenzen vollständig aus. Da die PHTs über die volle Sprungbefehlsadresse selektiert werden, wird auch von *per-address PHTs* gesprochen. Das Verfahren wird als GAp-Prädiktor bezeichnet.

Zum Anderen können neben der Adressierung über das BHR-Bitmuster die PHT-Einträge jeweils auf eine Untermenge der Sprungbefehle eingeschränkt werden, z.B. durch das Nutzen eines Adressteils des Sprungbefehls. Dieses Verfahren wird als GAs-Prädiktor bezeichnet. Da die PHTs von der Untermengenbildung abhängen, wird auch von *per-set PHTs* gesprochen.

Als Alternative zur globalen BHR kann die gesamte Sprungbefehlsadresse verwendet werden, um mehrere BHRs zu unterscheiden. Bei diesen Prädiktoren unterscheidet die erste Ebene des Prädiktors die letzten k Vorkommen desselben Sprungbefehls. Somit wird jedem Sprungbefehl ein eigenes BHR zugeordnet. Die per-address BHRs werden in einer Tabelle zusammengefasst, die als per-address BHT bezeichnet wird. In der zweiten Ebene können alle Varianten verwendet werden, die auch mit globalen BHR verwendet wurden, d.h. globale PHT, per-address PHT sowie per-set PHT. Damit ergeben sich drei Varianten mit per-address BHT, die als PAg-Prädiktor, PAp-Prädiktor und PAs-Prädiktor bezeichnet werden (siehe Tabelle 4.1).

Als weitere Technik kann eine Untermenge der Sprungbefehle gewählt werden, um mehrere BHT-Einträge zu unterscheiden. Hier speichert die erste Ebene der Prädiktoren die letzten k Vorkommen der Sprungbefehle derselben Sprungmenge, da jeder BHT-Eintrag mit einer Sprungmenge verknüpft ist. Die Sprungmenge kann durch verschiedene Kriterien definiert werden: durch den Opcode des Sprungbefehls, durch vom Compiler zugeordnete Sprungklassen oder durch einen Teil der Sprungbefehlsadresse. Die per-set BHRs werden wieder in einer Tabelle zusammengefasst, die als per-set BHT bezeichnet wird. Erneut werden durch Organisation der zweiten Ebene drei Prädiktorvarianten mit per-set BHT definiert, der SAg-Prädiktor, der SAs-Prädiktor und der SAp-Prädiktor.

4.1.5 Sprungvorhersage mit Markov-Ketten

Chen et al. [CCM96] verglichen die zweistufig adaptiven Prädiktoren mit Markov-Prädiktoren (siehe Abschnitt 4.2) aus den Bereich der Datenkomprimierung. In ihrer Arbeit zeigten sie, dass ein zweistufig adaptiver Prädiktor ein vereinfachter Markov-Prädiktor ist. Weiter stellten sie fest, dass der PPM-Algorithmus (*Prediction by Partial Matching*) (siehe Abschnitt 4.3), der im Bereich der Datenkomprimierung bestehend aus Markov-Prädiktoren eine optimale Vorhersagegenauigkeit liefert, auch angewandt auf zweistufig adaptive Prädiktoren eine optimale Treffergenauigkeit aufweist. Um diesen mathematischen Zusammenhang im weiteren Verlauf einsetzen zu können, soll der nächste Abschnitt eine kleine Einführung in Markov-Ketten und Markov-Prädiktoren geben. Anschließend soll der PPM-Algorithmus beschrieben werden.

4.2 Markov-Ketten

Im folgenden soll anhand einiger Definitionen der Begriff Markov-Kette hergeleitet werden [Bra97, Glo02, Beh99]. Sei Ω der *Ereignisraum*, d.h. die Menge der *elementaren Ereignisse*.

Definition 4.1 *Ein stochastischer Prozess oder Zufallsprozess ist eine Folge von elementaren Zufallsereignissen* $X_1, X_2, ..., X_i \in \Omega, i = 1, 2, ...$

Definition 4.2 *Die möglichen Zufallswerte in einem stochastischen Prozess heißen Zustände des Prozesses. Der Prozess befindet sich zum Zeitpunkt* t *in Zustand* X_t*.*

Definition 4.3 *Eine Markov-Kette* r*-ter Ordnung ist ein spezieller stochastischer Prozess, bei dem zu jedem Zeitpunkt die Wahrscheinlichkeit für den*

nächsten Zustand nur von den vorhergehenden r Zuständen abhängt (Markov-Eigenschaft), d.h. es gilt

$$P(X_t = x_t | X_1 = x_1, X_2 = x_2, ..., X_{t-1} = x_{t-1}) \\ = P(X_t = x_t | X_{t-r} = x_{t-r}, ..., X_{t-1} = x_{t-1})$$

Oder anders ausgedrückt: der Prozess vergisst frühere Ereignisse.

Bemerkung 4.4 Oft wird mit dem Begriff Markov-Kette nur die Markov-Kette erster Ordnung bezeichnet, hierbei hängen dann zu jedem Zeitpunkt die Wahrscheinlichkeiten für zukünftige Zustände nur vom momentanen Zustand ab, es gilt somit

$$P(X_t = x_t | X_1 = x_1, X_2 = x_2, ..., X_{t-1} = x_{t-1}) \\ = P(X_t = x_t | X_{t-1} = x_{t-1})$$

Eine Markov-Kette r-ter Ordnung über einer Menge S von Zuständen ist äquivalent zu einer Markov-Kette erster Ordnung über einer Menge S^r von r-Tupeln aus S. Dies folgt direkt aus

$$P(X_{t-r+1}, ..., X_{t-1}, X_t | X_{t-r}, ..., X_1) \\ = P(X_t | X_{t-r}, ..., X_{t-1})$$

Beispiel 4.5 Sei $S = \{0, 1\}$ die Menge von Zuständen. Betrachtet man auf diesen Zuständen eine Markov-Kette zweiter Ordnung, dann ist eine Sequenz dieser Kette äquivalent zu einer Sequenz von Paaren aus einer Markov-Kette erster Ordnung (Abbildung 4.5). Die Sequenz 01101 beispielsweise entspricht $01 - 11 - 10 - 01$.

In dieser Kette sind nicht alle Übergänge erlaubt, da auf ein Symbol jeweils nur zwei andere folgen dürfen. Auf den Zustand 01 können beispielsweise nur die Zustände 11 und 10 folgen, sonst wäre keine Äquivalenz zu einer Kette zweiter Ordnung möglich.

Obwohl sich alle Markov-Ketten r-ter Ordnung auf Markov-Ketten erster Ordnung zurückführen lassen, ist es in der Anwendung manchmal praktischer, das Modell der höheren Ordnung zu benutzen. Theoretisch können sie aber immer gleich behandelt werden.

Definition 4.6 *Die Übergangswahrscheinlichkeiten einer Markov-Kette können in einer Übergangsmatrix dargestellt werden. Betrachte eine Markov-Kette mit*

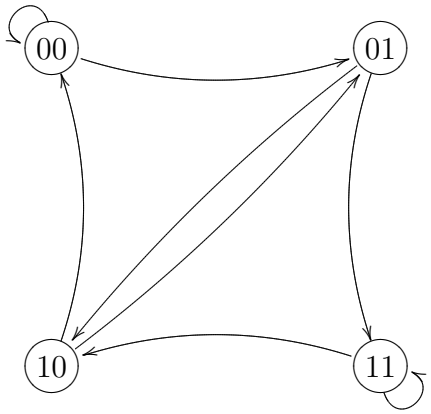


Abbildung 4.5: Markov-Kette erster Ordnung für 2-Tupel mit 4 Zuständen

n Zuständen $z_1, ..., z_n$. Sei $p_{ij}(t)$ die Übergangswahrscheinlichkeit von Zustand z_i nach Zustand z_j zum Zeitpunkt t, dann sieht die Übergangsmatrix wie folgt aus:

$$P(t) = \begin{pmatrix} p_{11}(t) & ... & p_{1n}(t) \\ ... & ... & ... \\ p_{n1}(t) & ... & p_{nn}(t) \end{pmatrix}, \quad p_{ij}(t) \geq 0, \quad \sum_{j=1}^{n} p_{ij}(t) = 1, i = 1, ..., n$$

Eine Matrix mit diesen Eigenschaften wird auch stochastische Matrix genannt.

Beispiel 4.7 Betrachtet man die Übergangswahrscheinlichkeiten für die Markov-Kette erster Ordnung aus Abbildung 4.5, ergibt sich folgende Übergangsmatrix zum Zeitpunkt t:

$$P(t) = \begin{pmatrix} p_{00\to00}(t) & p_{00\to01}(t) & 0 & 0 \\ 0 & 0 & p_{01\to10}(t) & p_{01\to11}(t) \\ p_{10\to00}(t) & p_{10\to01}(t) & 0 & 0 \\ 0 & 0 & p_{11\to10}(t) & p_{11\to11}(t) \end{pmatrix}$$

Die Zeilensummen der Matrix müssen wieder 1 ergeben. Übergänge, die nie auftreten können, erhalten die Wahrscheinlichkeit 0.

Definition 4.8 *Sind die Übergangswahrscheinlichkeiten zu jedem Zeitpunkt gleich, d.h. unabhängig von t, wird von einer (zeitlich) homogenen Markov-Kette gesprochen.*

Eine Markov-Kette bezeichnet man als inhomogen, wenn sich die Übergangswahrscheinlichkeiten periodisch ändern, d.h. die Übergangswahrscheinlichkeiten hängen von der Position i in der Sequenz und einer Periodenlänge T ab.

Inhomogene Markov-Ketten könnte man so interpretieren, dass man eine Reihe von T verschiedenen Markov-Ketten mit gleichen Zuständen, aber unterschiedlichen Übergangswahrscheinlichkeiten betrachtet. Jedes Mal, wenn ein Zustand gewechselt wird, wechselt man auch zur Markov-Kette in der Reihe. Nach dem T-ten Übergang wechselt man wieder zur ersten Kette. Inhomogene Markov-Ketten werden im Folgenden nicht weiter betrachtet. Eine einfache Markov-Kette ist der Markov-Prädiktor [CCM96, Ros85].

Definition 4.9 *Ein Markov-Prädiktor r-ter Ordnung ist eine Markov-Kette r-ter Ordnung, bei der die Wahrscheinlichkeit für den nächsten Zustand z_t durch die relativen Häufigkeiten des Auftretens der Zustände nach den vorhergehenden r Zuständen $z_{t-r}z_{t-r+1}\ldots z_{t-2}z_{t-1}$ in der Sequenz $z_1z_2\ldots z_{t-2}z_{t-1}$ angegeben werden.*

Beispiel 4.10 Sei $S = \{0,1\}$ die Menge von Zuständen und sei $z_1z_2\ldots z_{t-2}z_{t-1} = 01101011$ die Sequenz der eingetretenen Zustände. Abbildung 4.6 zeigt einen entsprechenden Markov-Prädiktor zweiter Ordnung, wobei f die relative Häufigkeit angibt. Der Markov-Prädiktor bestimmt den Zustand 0 als nächsten Zustand.

$z_1z_2\ldots z_{t-2}z_{t-1}$
0110101<u>1</u>

$z_{t-2}z_{t-1}$	z_t	f
00	0	-
	1	-
01	0	0,5
	1	0,5
10	0	0,0
	1	1,0
11	0	1,0
	1	0,0

Abbildung 4.6: Markov-Prädiktor zweiter Ordnung

4.3 Prediction by Partial Matching

Der PPM-Algorithmus (*Prediction by Partial Matching*) entstand ursprünglich aus einem Kompressionsalgorithmus [ZL77]. Ein PPM-Algorithmus der Ordnung n besteht aus $n+1$ Markov-Prädiktoren der Ordnungen 0 bis n. In jedem Schritt wählt der Algorithmus den Markov-Prädiktor n-ter Ordnung und sucht in diesem das Muster der letzten n eingetretenen Zustände. Enthält der Markov-Prädiktor dieses Muster, wird er zur Vorhersage des nächsten Zustandes verwendet. Enthält

der Markov-Prädiktor n-ter Ordnung das Muster nicht, wird n dekrementiert. Dies wird bis $n = 1$ durchgeführt. Wird im Markov-Prädiktor erster Ordnung der letzte Zustand, also das Muster der Länge 1, nicht gefunden, wird der Markov-Prädiktor 0-ter Ordnung zur Vorhersage verwendet. Der Markov-Prädiktor 0-ter Ordnung gibt den Zustand als Vorhersage an, der in der vergangenen Sequenz am häufigsten auftrat.

Algorithmus 4.11 *Prediction by Partial Matching*

Eingabe: $z_1 \dots z_{t-1}$ (Sequenz von Zuständen), n (Ordnung)

Ausgabe: z_t (nächster Zustand)

`REPEAT`

Wähle Markov-Prädiktor n-ter Ordnung

Suche $z_{t-n} \dots z_{t-1}$ in Markov-Prädiktor n-ter Ordnung

Falls vorhanden, verwende Markov-Prädiktor zur Vorhersage `BREAK`

Andernfalls, setze $n = n - 1$

`UNTIL` $n = 0$

Falls $n = 0$, verwende Markov-Prädiktor 0-ter Ordnung zur Vorhersage

Beispiel 4.12 Sei $S = \{0, 1\}$ die Menge von Zuständen und sei $z_1 z_2 \dots z_7 z_8 = 01101011$ die Sequenz der eingetretenen Zustände. Dann kann der PPM-Algorithmus 5-ter Ordnung durch das Flussdiagramm in Abbildung 4.7 dargestellt werden. Der Algorithmus sucht zunächst im Markov-Prädiktor 5-ter Ordnung nach dem Muster $z_4 z_5 z_6 z_7 z_8 = 01011$, welches aber nicht gefunden wird, da es in der Sequenz noch nie aufgetreten ist. Jetzt sucht der Algorithmus im Markov-Prädiktor 4-ter Ordnung nach dem Muster 1011, welches auch nicht gefunden wird. Die Suche nach dem Muster 011 im Markov-Prädiktor 3-ter Ordnung ist erfolgreich, da dieses Muster zu Beginn der Sequenz aufgetreten ist. Nun kann der Markov-Prädiktor 3-ter Ordnung den nächsten Zustand vorhersagen.

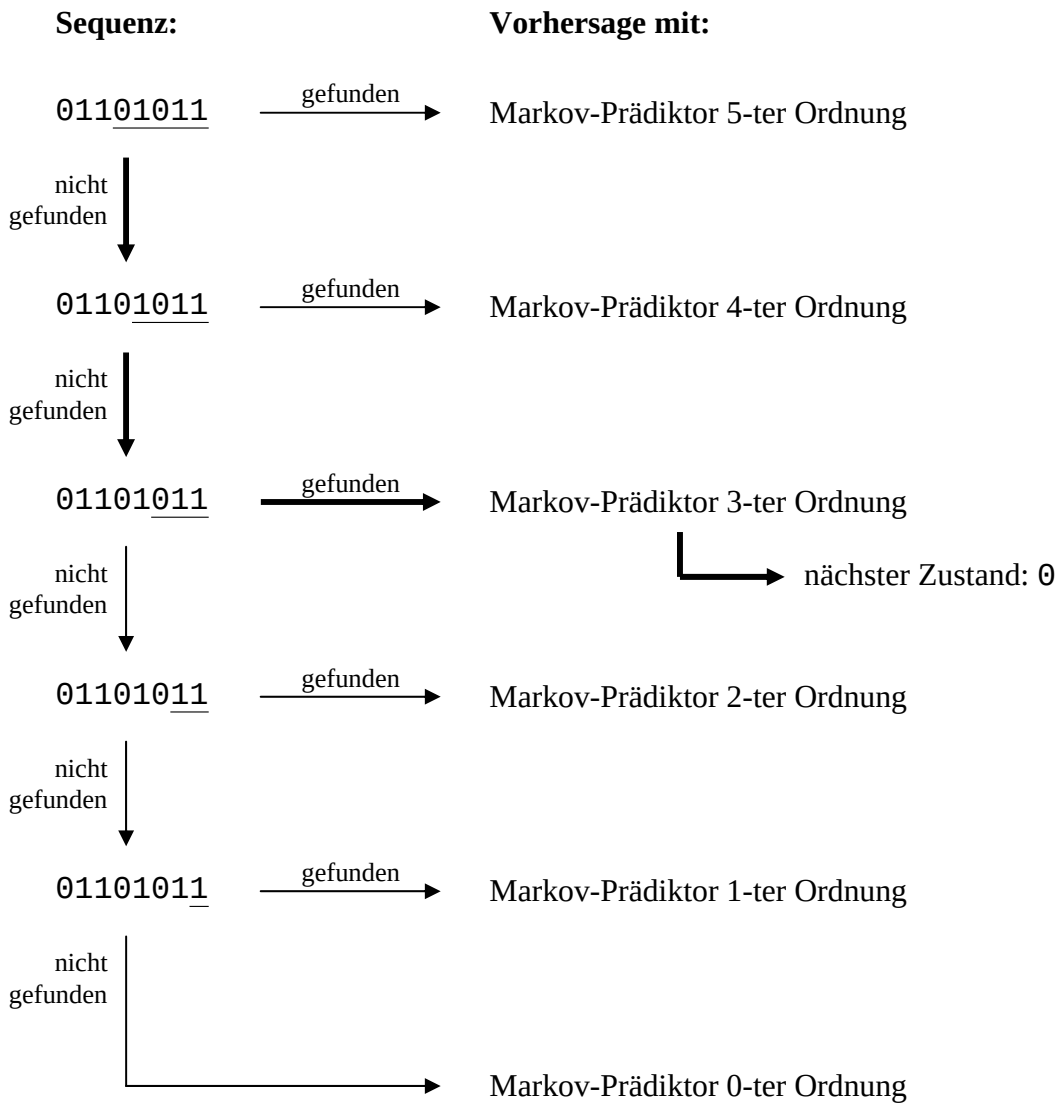


Abbildung 4.7: PPM-Algorithmus 5-ter Ordnung

5 Zustandsprädiktoren

In diesem Kapitel wird die neue Klasse der Zustandsprädiktoren [PBTU03a, PBTU03b, PBTU03c, PBT$^+$04] eingeführt, welche durch die Sprungvorhersagetechniken aus dem Bereich der Prozessorarchitektur motiviert wurden. Im Folgenden wird davon ausgegangen, dass die Historie eines Kontext bekannt ist.

Die Klasse der Zustandsprädiktoren gliedert sich in verschiedene Unterklassen (siehe Abbildung 5.1). Zunächst gibt es eine Unterteilung in einstufige und zweistufige Zustandsprädiktoren. Die zweistufigen Prädiktoren gliedern sich weiter in lokale und globale Prädiktoren. Die einstufigen Prädiktoren gibt es nur als lokale Prädiktoren.

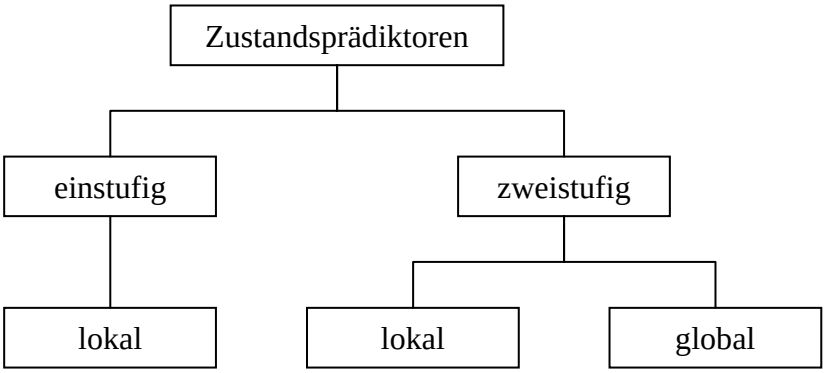


Abbildung 5.1: Unterteilung der Zustandsprädiktoren

Der Unterschied zwischen lokalen und globalen Prädiktoren bezieht sich auf die betrachtete Folge von Kontexten. Eine globale Folge von Kontexten ist einfach die Folge, in der die Kontexte aufgetreten sind. Eine lokale Folge von Kontexten bezieht sich auf einen bestimmten Kontext. Sie besteht aus den Nachfolgern des Bezugskontextes in der globalen Folge.

Beispiel 5.1 Sei

$$ACBCACBCABCAB$$

eine Folge von Kontexten. Dann ist

$$BCAB$$

die globale Folge der letzten vier aufgetretenen Kontexte. Die Kontexte

$$CCBB$$

bilden die lokale Folge von Kontexten bezogen auf den Kontext A.

5.1 Einstufige Zustandsprädiktoren

Zunächst werden die einstufigen Zustandsprädiktoren beschrieben. Es gibt mehrere einstufige Zustandsprädiktoren, die anhand der Anzahl ihrer Zustände unterschieden werden. Im Folgenden wird der One-State-Prädiktor, der Two-State-Prädiktor sowie der n-State-Prädiktor beschrieben. Die einstufigen Prädiktoren werden nur lokal betrachtet, d.h. es werden Folgen von Kontexten betrachtet, die aus den Nachfolgekontexten eines bestimmten Kontextes bestehen.

5.1.1 One-State-Prädiktor

Sei $\mathcal{C}$ eine Menge von Kontexten und sei $Z \in \mathcal{C}$ ein Kontext. Ziel ist es nun vorherzusagen, welcher Kontext nach dem Kontext Z eintreten wird. Beim ersten Auftreten des Kontextes Z kann keine Vorhersage über den nächsten Kontext gemacht werden. Tritt jetzt ein Kontext $A \in \mathcal{C}$ ein, wird dieser als Zustand des One-State-Prädiktor gespeichert. Beim nächsten Auftreten von Z wird der Kontext A als nächster Kontext vorhergesagt. Erweist sich die Vorhersage als korrekt, bleibt der One-State-Prädiktor im Zustand A und sagt beim nächsten Auftreten von Z den Kontext A wieder vorher. War die Vorhersage falsch und es ist ein Kontext B eingetreten, setzt der Prädiktor B als neuen Zustand und sagt diesen das nächste Mal vorher.

Da der Prädiktor für jeden Kontext einen Zustand besitzt, wird er als One-State-Prädiktor bezeichnet. Sei $Z \in \mathcal{C}$ ein Kontext und seien $A, B \in \mathcal{C}$ die zwei einzigen Kontexte, die nach dem Kontext Z auftreten können, dann kann der One-State-Prädiktor bezüglich des Kontextes Z durch den Vorhersagegraph in Abbildung 5.2 dargestellt werden. In diesem Fall ist der One-State-Prädiktor analog zum Ein-Bit-Prädiktor der Sprungvorhersage.

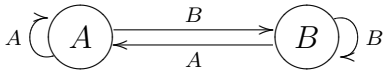


Abbildung 5.2: One-State-Prädiktor für die Kontexte A und B

Wenn zusätzlich zu A und B ein weiterer Kontext C nach dem Bezugskontext Z auftreten kann, muss der Vorhersagegraph des One-State-Prädiktors bezüglich Z wie in Abbildung 5.3 erweitert werden.

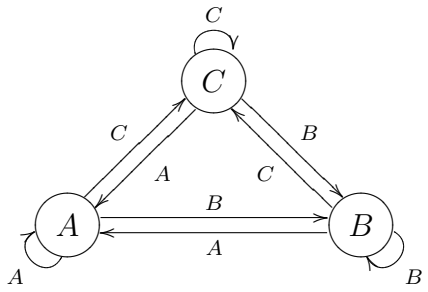


Abbildung 5.3: One-State-Prädiktor für die Kontexte A, B und C

Bei Hinzunahme weiterer Kontexte kann die Erstellung des Vorhersagegraphes analog fortgesetzt werden. Jeder Kontext bildet einen Knoten im Graph. Der Vorhersagegraph ist ein vollständig gerichteter Graph.

Beispiel. *Location Prediction*

Im Folgenden soll der One-State-Prädiktor am Beispiel der Vorhersage des nächsten Aufenthaltsortes in einem Bürogebäude beschrieben werden. Der Bezugskontext Z ist in diesem Fall der aktuelle Aufenthaltsort, d.h. jeder Aufenthaltsort besitzt einen One-State-Prädiktor, der von diesem aus den nächsten Aufenthaltsort vorhersagt.

Betritt eine Person das erste Mal einen Raumes R, kann keine Vorhersage gemacht werden. Beim Verlassen des Raumes R wird dann der Zielraum T als aktueller Zustand des One-State-Prädiktors gesetzt. Kommt die Person wieder in den Raum R wird nun als nächster Raum der Raum T vorhergesagt.

Betrachten wir zunächst den Flur mit den beiden Nachbarräumen Sekretariat S und Büro des Chefs C (siehe Abbildung 5.4).

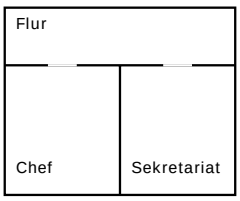


Abbildung 5.4: Flur mit Sekretariat und Büro des Chefs

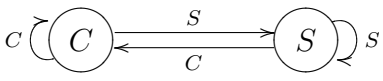


Abbildung 5.5: One-State-Prädiktor des Flurs

Abbildung 5.5 zeigt den Vorhersagegraph des One-State-Prädiktors des Flurs. Dieser ist wie folgt zu verstehen: Die Zustände geben an, welcher Raum vorhergesagt wird. Geht eine Person vom Flur in das Büro vom Chef C, wird der Prädiktor mit dem Zustand C initialisiert. Befindet sich diese Person nun wieder im Flur, wird vorhergesagt, dass sie als nächstes in das Büro des Chefs C geht. Ist dies der Fall, bleibt der Prädiktor im Zustand C. Andernfalls, d.h. die Person geht in das Sekretariat S, wechselt der Prädiktor in den Zustand S und sagt somit das nächste Mal das Sekretariat voraus.

Hat der Flur nun als weiteren Nachbarraum das Büro des Mitarbeiters M (siehe Abbildung 5.6), muss der One-State-Prädiktor wie in Abbildung 5.7 erweitert werden.

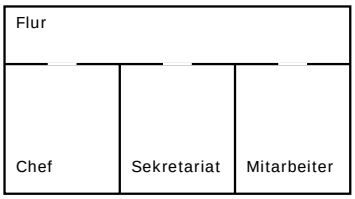


Abbildung 5.6: Flur mit Sekretariat, Büro des Chefs und Büro des Mitarbeiters

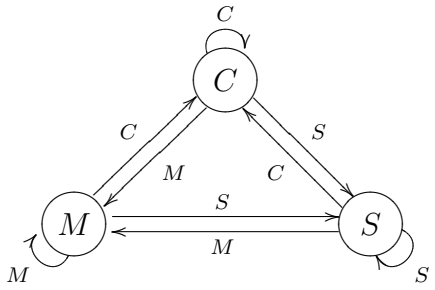


Abbildung 5.7: One-State-Prädiktor des Flurs

Bei weiteren Nachbarräumen kann das Prinzip analog fortgesetzt werden. Jeder Nachbarraum bildet einen Knoten des Graphs, der vollständig gerichtet ist.

Speicher- und Rechenaufwand. Speicher- und Rechenaufwand des One-State-Prädiktor sind sehr gering. Es muss für jeden Kontext nur die ID des nachfolgenden Kontextes gespeichert werden. Der Rechenaufwand besteht nur aus der Selektion des Prädiktors entsprechend dem aktuellen Kontext sowie dem Auslesen und Anpassen des Zustandes des Prädiktors.

Vorteile. Ein Vorteil des One-State-Prädiktors ist das schnelle Anlernen. Ein Kontext, für den der nachfolgende Kontext vorhergesagt werden soll, muss in der Vergangenheit nur einmal mit einem Nachfolgekontext aufgetreten sein, dann kann eine Vorhersage über den nächsten Kontext getroffen werden.

Nachteile. Ein möglicher Nachteil ist das zu schnelle Umlernen. Auf einen Kontext folgt immer der gleiche Kontext. Wenn jetzt einmalig auf den Kontext ein anderen Kontext folgt, sagt der Prädiktor das nächste Mal den falschen Kontext

vorher, d.h. der Prädiktor ist sehr sensibel gegenüber einmaligen Abweichungen von der Gewohnheit. Eine Abhilfe schaffen die Two- oder k-State-Prädiktoren.

5.1.2 Two-State-Prädiktor

Der Two-State-Prädiktor hat für jeden möglichen Nachfolgekontext zwei Zustände, in denen der entsprechende Kontext vorhergesagt wird. Diese beide Zustände werden als *schwacher* und *sicherer* Zustand bezeichnet. Zur Markierung des schwachen bzw. sicheren Zustandes wird ein Bit benötigt, welches neben der ID für den Nachfolgekontext den Zustand des Prädiktor beschreibt.

Sei $\mathcal{C}$ eine Menge von Kontexten und sei $Z \in \mathcal{C}$ ein Kontext. Ziel ist es wieder vorherzusagen, welcher Kontext nach dem Kontext Z eintreten wird. Beim ersten Auftreten des Kontextes Z kann keine Vorhersage über den nächsten Kontext gemacht werden. Tritt jetzt ein Kontext $A \in \mathcal{C}$ ein, wird der schwache Zustand $A0$ als Initialzustand des Two-State-Prädiktors gesetzt. Beim nächsten Auftreten von Z wird der Kontext A als nächster Kontext vorhergesagt. Erweist sich die Vorhersage als korrekt, wechselt der Two-State-Prädiktor in den sicheren Zustand $A1$ und sagt beim nächsten Auftreten von Z den Kontext A wieder vorher. Ist diese Vorhersage korrekt, bleibt der Prädiktor im Zustand $A1$. Andernfalls wechselt der Prädiktor in den schwachen Zustand $A0$. Beim nächsten Auftreten von Kontext Z wird aber noch Kontext A vorhergesagt. Ist nun die Vorhersage in einem schwachen Zustand falsch, d.h. es tritt der Kontext B anstatt des Kontextes A ein, setzt der Two-State-Prädiktor den schwachen Zustand $B0$ als neuen Zustand und sagt beim nächsten Auftreten von Kontext Z den Kontext B vorher.

Sei $Z \in \mathcal{C}$ ein Kontext und seien A, $B \in \mathcal{C}$ die zwei einzigen Kontexte, die nach dem Kontext Z auftreten können, dann kann der Two-State-Prädiktor bezüglich des Kontextes Z durch den Vorhersagegraph in Abbildung 5.8 beschrieben werden. In diesem Fall ist der Two-State-Prädiktor analog zum Zwei-Bit-Prädiktor mit Sättigungszähler.

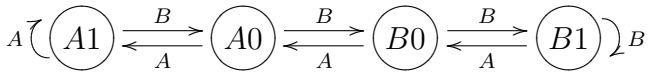


Abbildung 5.8: Two-State-Prädiktor für die Kontexte A und B

Wenn zusätzlich zu A und B ein weiterer Kontext C nach dem Bezugskontext Z auftreten kann, muss der Vorhersagegraph des Two-State-Prädiktors bezüglich Z wie in Abbildung 5.9 erweitert werden.

Bei Hinzunahme weiterer Kontexte kann die Erstellung des Vorhersagegraphes analog fortgesetzt werden. Für jeden Kontext gibt es zwei Knoten im Graph, einen

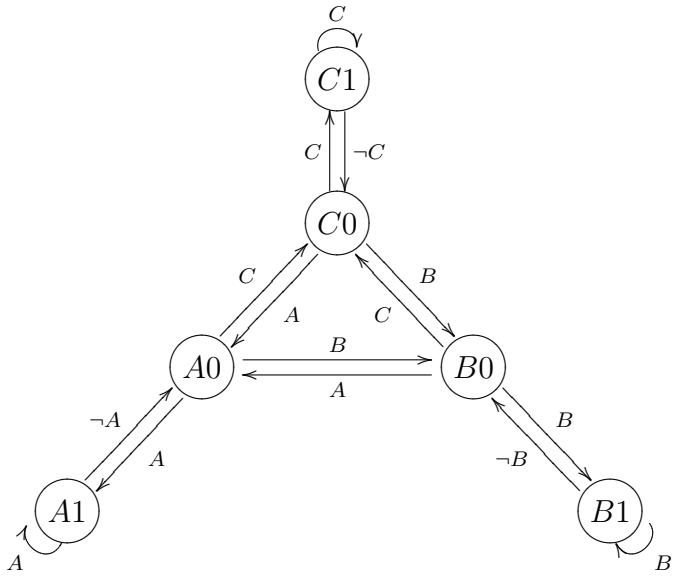


Abbildung 5.9: Two-State-Prädiktor für die Kontexte A, B und C

für den schwachen und einen für den sicheren Zustand. Die schwachen Zustände des Two-State-Prädiktors bilden wiederum einen vollständig gerichteten Graph.

Beispiel. *Location Prediction*

Im Folgenden soll der Two-State-Prädiktor am Beispiel der Vorhersage des nächsten Aufenthaltsortes in einem Bürogebäude beschrieben werden. Der Bezugskontext Z ist in diesem Fall der aktuelle Aufenthaltsort, d.h. für jeden Aufenthaltsort gibt es einen Two-State-Prädiktor, der von diesem aus den nächsten Aufenthaltsort vorhersagt.

Betrachten wir zunächst den Flur mit den beiden Nachbarräumen Sekretariat S und Büro des Chefs C (siehe Abbildung 5.4).

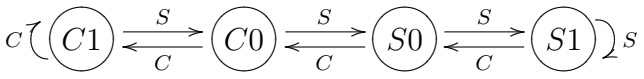


Abbildung 5.10: Two-State-Prädiktor des Flurs

Abbildung 5.10 zeigt den Vorhersagegraph des Two-State-Prädiktors. Befindet sich eine Person das erste Mal im Flur und geht in das Büro des Chefs C, wird der Zustand $C0$ als Initialzustand gesetzt. Befindet sich die Person nun wieder im Flur, wird vorhergesagt, dass sie als nächstes in das Büro des Chefs C geht. Ist dies der Fall, wird in den Zustand $C1$ umgeschaltet, d.h. es wird in den sicheren

Zustand gewechselt. Es wird somit das nächste Mal wieder das Büro des Chefs C vorhergesagt. Geht die Person nun irgendwann einmal vom Flur in das Sekretariat S, wird der Zustand wieder auf $C0$ gesetzt. Damit wird immer noch das Büro des Chefs vorhergesagt. Erst wenn die Person zweimal aufeinander folgend vom Flur in das Sekretariat S geht, wird der Zustand des Two-State-Prädiktors auf $S0$ gesetzt wird, und es wird als nächstes das Sekretariat vorhergesagt.

Hat der Flur nun als weiteren Nachbarraum das Büro des Mitarbeiter M siehe Abbildung 5.6), muss der Two-State-Prädiktor erweitert werden.

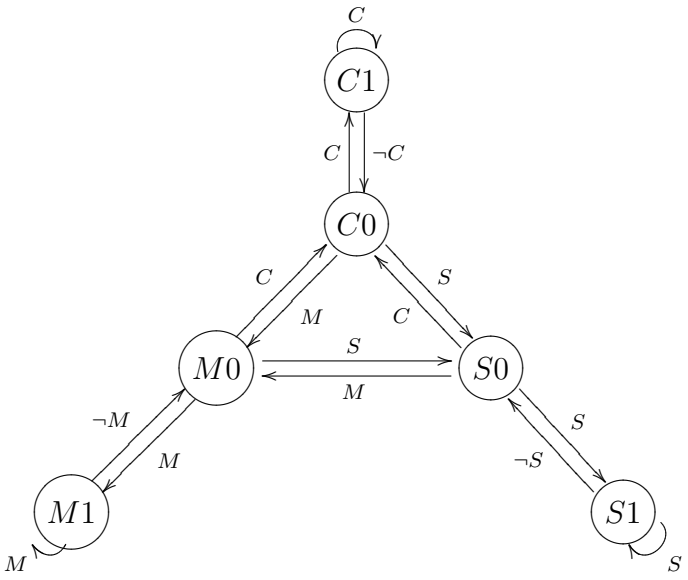


Abbildung 5.11: Two-State-Prädiktor des Flurs

Der Vorhersagegraph in Abbildung 5.11 ist wie folgt zu verstehen: Die Initialisierung und der Übergang in einen sicheren Zustand ist wie oben beschrieben. Eine Person ist mehrmals vom Flur in das Büro vom Chef gegangen, der Prädiktor befindet sich also im sicheren Zustand $C1$. Wenn sie das nächste Mal vom Flur in einen anderen Raum als das Büro vom Chef geht, wechselt der Prädiktor in den Zustand $C0$, sagt also immer noch das Büro vom Chef voraus. Geht die Person nun vom Flur in das Sekretariat, schaltet der Prädiktor in den Zustand $S0$ unabhängig von dem vorher vom Flur aus betretenen Raum, und sagt somit das Sekretariat als nächstes voraus. Befindet sich der Prädiktor zum Beispiel im Zustand $C1$ und die Person betritt das nächste Mal nicht den Raum C, d.h. es wird der Raum M oder S betreten ($\neg C = M \vee S$). Die Information, um welchen Raum es sich handelt, geht dabei verloren.

Bei weiteren Nachbarräumen kann das Prinzip analog fortgesetzt werden. Für jeden Nachbarraum gibt es zwei Knoten im Vorhersagegraph. Die schwachen Zustände bilden einen vollständig gerichteten Graphen.

Speicher- und Rechenaufwand. Beim Two-State-Prädiktor muss zusätzlich zur ID des zu vorhersagenden Kontextes das Bit für die Unterscheidung zwischen schwachen und sicheren Zustand gespeichert werden, d.h. auch hier ist der Speicheraufwand sehr gering. Der Rechenaufwand besteht wiederum aus der Selektion des Prädiktors entsprechend dem aktuellen Kontext sowie dem Auslesen und Anpassen des Zustandes des Two-State-Prädiktors.

Vorteile. Auch der Two-State-Prädiktor wird schnell angelernt. Das Umlernen wird verlangsamt, so dass einmaliges Ändern einer Gewohnheit keine Auswirkung hat. Ein annähernd dauerhaftes Erlernen einer Gewohnheit mit Abweichungen, die übergangen werden sollen, kann mit einem k-State-Prädiktor erreicht werden.

Nachteile. Muster in einem Verhaltensablauf können nicht dauerhaft erlernt werden. Bei zwei aufeinander folgenden Abweichungen der Gewohnheit hat das System sich die Änderung gemerkt. Dies kann in einigen Fällen zu schnell sein. Auch das Pendeln zwischen zwei Kontexten kann dazu führen, dass immer der falsche Kontext vorhergesagt wird.

5.1.3 k-State-Prädiktor

Die ersten beiden Prädiktoren waren Spezialfälle des k-State-Prädiktor, der für jeden Kontext k Zustände besitzt, in denen dieser vorhergesagt wird. Der k-State-Prädiktor entspricht für zwei mögliche Nachfolgekontexte dem n-Bit-Prädiktor mit Sättigungszähler, wobei $k = 2^{n-1}$. Neben der ID des Kontextes wird in jedem Zustand ein Zähler gespeichert. Ist der Zähler gesättigt, werden k Fehlvorhersagen benötigt, um einen anderen Kontext vorherzusagen.

Sei $Z \in \mathcal{C}$ ein Kontext und seien A, B, $C \in \mathcal{C}$ die drei einzigen Kontexte, die nach dem Kontext Z auftreten können, dann kann der k-State-Prädiktor bezüglich des Kontextes Z durch den Vorhersagegraph in Abbildung 5.12 beschrieben werden. Dabei sei $h = k - 1$.

Der Vorhersagegraph ist analog zu dem des Two-State-Prädiktors zu verstehen. Der Unterschied liegt darin, dass der Prädiktor erst den sichersten Zustand erreicht hat, wenn der Zähler auf k steht. In diesem Fall werden nun $k - 1$ aufeinander folgende Übergänge in andere Nachfolgekontexte benötigt, bis ein anderer Kontext vorhergesagt wird. Der Speicheraufwand für den k-State-Prädiktor ist unwesentlich größer als der des Two-State-Prädiktors. Hier muss die ID des zu vorhersagenden Kontextes und der Zähler gespeichert werden. Der Rechenaufwand gleich dem des Two-State-Prädiktors.

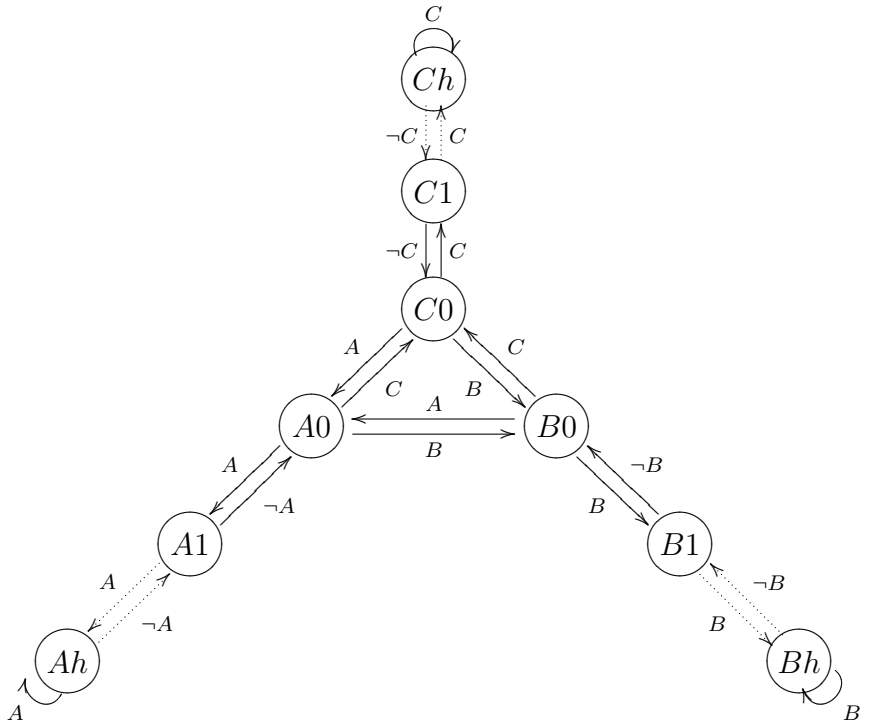


Abbildung 5.12: k-State-Prädiktor für die Kontexte A, B und C

Im Bereich der Sprungvorhersage folgt auf den Zwei-Bit-Prädiktor, der für zwei Kontexte dem Two-State-Prädiktor mit $2^{2-1} = 2$ Zuständen entspricht, der Drei-Bit-Prädiktor, der für zwei Kontexte dem Vier-State-Prädiktor mit $2^{3-1} = 4$ Zuständen entsprechen würde. Da man bei der Kontextvorhersage aber der Einschränkung auf Bits nicht unterliegt, kann der Sättigungszähler nicht nur Zweierpotenzen als Wert annehmen, sondern alle natürlichen Zahlen. Somit gibt es beispielsweise auch den Three-State-Prädiktor, für den es keinen entsprechenden Prädiktor in der Sprungvorhersage gibt, da die Anzahl der daraus resultierenden Bits ($\log_2 3 + 1$) keine natürliche Zahl ist. Es gibt also nur für k-State-Prädiktoren einen entsprechenden Prädiktor im Bereich der Sprungvorhersage, wenn gilt: $\log_2 k + 1 \in \mathbb{N}$.

Mit k (Sättigungszähler) wird sozusagen angegeben, wie schnell eine Gewohnheit geändert werden soll. Eine weitere Idee beim k-State-Prädiktor ist auch das Erlernen von k, d.h. es soll erlernt werden, mit welcher Geschwindigkeit eine Gewohnheit geändert werden soll. In der Sprungvorhersage wurden mit dem Drei-Bit-Prädiktor, welcher acht Zustände hat, keine wesentlichen Verbesserungen erzielt [ŠRU99]. Deshalb soll auch hier in der Evaluierung nicht auf mehr als zwei Zustände pro Kontext eingegangen werden.

5.2 Zweistufige Zustandsprädiktoren

Bei den zweistufigen Zustandsprädiktoren wird eine Folge der zuletzt eingetretenen Kontexte betrachtet. Anhand der betrachteten Folge wird zwischen lokalen und globalen zweistufigen Zustandsprädiktoren unterschieden. Die Länge der Folge wird durch die Ordnung r bestimmt, d.h. es werden die letzten r eingetretenen Kontexte zur Vorhersage herangezogen. Zur Bestimmung des nächsten Kontextes werden alle möglichen Muster aufeinander folgender Kontexte in einer so genannten Musterverlaufstabelle abgespeichert. Zur Vorhersage des nächsten Kontextes gibt es dann in der zweiten Stufe einen Two-State-Prädiktor. Zur Auswahl des Musters aus der Tabelle gibt es ein Schieberegister, in dem die r zuletzt eingetretenen Kontexte gespeichert sind. Tritt ein neuer Kontext ein, werden alle Einträge des Register nach links verschoben und der neue Kontext wird rechts eingetragen.

5.2.1 Globale zweistufige Zustandsprädiktoren

Sei $\mathcal{C}$ die Menge der Kontexte mit $|\mathcal{C}| = n$ und sei r die Ordnung, dann gibt es n^r Muster. Da aber bei den globalen zweistufigen Prädiktoren diejenigen Muster nie eintreten, bei denen zwei aufeinander folgende Einträge gleich sind (z.B. A - B - B), ist die Anzahl der möglichen Muster $n \cdot (n-1)^{r-1}$. Dies ist also die maximale Größe der Musterverlaufstabelle unter der Annahme, dass jeder Kontext auf jeden anderen folgen kann. Da dies in der Regel nicht der Fall sein wird, reduziert sich die Größe der Tabelle in der Realität noch weiter.

In der zweiten Stufe wird ein Two-State-Prädiktor eingesetzt, d.h. zu jedem Muster in der Musterverlaufstabelle gibt es einen Two-State-Prädiktor (siehe Abschnitt 5.1.2), der mittels des Eintrags im Schieberegister selektiert wird, und damit den nächsten Kontext vorhersagt bzw. zur Anpassung mit dem tatsächlich eingetretenen Kontext verwendet wird.

Sei $\mathcal{C} = \{A, B, C\}$ und $r = 3$, dann gibt es $3 \cdot 2^2 = 12$ Muster. Sei

$$ABCACBABACBACB$$

die Folge der zuletzt aufgetretenen Kontexte. Das Schieberegister und die Musterverlaufstabelle eines entsprechenden globalen zweistufigen Zustandsprädiktor zeigt Abbildung 5.13. Das Schieberegister enthält A C B und selektiert damit das entsprechende Muster in der Musterverlaufstabelle. Der zugehörige Two-State-Prädiktor sagt den Kontext A als nächsten Kontext vorher. Mit dem dann tatsächlich eintretenden Kontext wird dieser Two-State-Prädiktor angepasst.

Der globale zweistufige Zustandsprädiktor mit Two-State-Prädiktor in der zweiten Stufe und Ordnung 1 entspricht dem einstufigen Two-State-Prädiktor. Statt

Schieberegister

Musterverlaufstabelle

Muster	Two-State-Prädiktor
$A\ B\ A$	$C0$
$A\ B\ C$	$A0$
$A\ C\ A$	$-$
$A\ C\ B$	$A1$
$B\ A\ B$	$A0$
$B\ A\ C$	$B1$
$B\ C\ A$	$C0$
$B\ C\ B$	$-$
$C\ A\ B$	$-$
$C\ A\ C$	$B0$
$C\ B\ A$	$C0$
$C\ B\ C$	$-$

Abbildung 5.13: Globaler zweistufiger Zustandsprädiktor

eines Two-State-Prädiktors in der zweiten Stufe kann auch ein beliebiger k-State-Prädiktor verwendet werden. Die globalen zweistufigen Zustandsprädiktoren entsprechen den GAg-Prädiktoren der Sprungvorhersage, die ein einziges globales Schieberegister und eine einzige globale Musterverlaufstabelle verwenden.

Beispiel. *Location Prediction*

Wir betrachten die drei Räume Flur F, Sekretariat S und Büro des Chefs C (siehe Abbildung 5.14). Die Ordnung sei 3, dann gibt es $3 \cdot 2^2 = 12$ Muster. Sei

$$FSCFCSFSFCSFCS$$

die abgelaufene Raumfolge einer Person, dann zeigt Abbildung 5.15 das Schieberegister und einen Ausschnitt der Musterverlaufstabelle. Die Vorhersage ist nun, dass die Person als nächsten Raum den Flur F betreten wird.

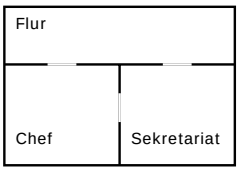


Abbildung 5.14: Flur, Sekretariat und Büro des Chefs

Muster	2S
$\cdots$	$\cdots$
$C\ S\ F$	$C0$
$\cdots$	$\cdots$
$F\ C\ S$	$F1$
$\cdots$	$\cdots$

Abbildung 5.15: Globaler zweistufiger Zustandsprädiktor

Speicher- und Rechenaufwand. Der Speicheraufwand ist abhängig von der Anzahl der möglichen Kontexte und von der gewählten Ordnung. Sei die Anzahl der Kontexte n und die Ordnung r, dann müssen $n \cdot (n-1)^{r-1}$ Muster und genauso viele Two-State-Prädiktoren gespeichert werden. Der Rechenaufwand setzt sich aus der Suche des entsprechenden Musters in der Musterverlaufstabelle sowie dem Auslesen und Anpassen des verwendeten Prädiktor in der zweiten Stufe zusammen.

Vorteile. Komplizierte Muster können erkannt werden. Da jedes Muster separat behandelt wird, kann es nicht zu unerwünschten Interferenzen [BU02] zwischen zwei Mustern wie bei der Sprungvorhersage kommen.

Nachteile. Je größer die Ordnung ist, desto länger ist der Anlernprozess der globalen zweistufigen Zustandsprädiktoren.

5.2.2 Lokale zweistufige Zustandsprädiktoren

Hier werden lokale Folgen von Kontexten betrachtet, d.h. es existiert für jeden Kontext ein lokaler zweistufiger Prädiktor, der die Sequenz der aufgetretenen Nachfolgekontexte für die Vorhersage verwendet. In ihrer Funktionsweise sind die lokalen zweistufigen Prädiktoren analog zu den globalen zweistufigen Prädiktoren. In der zweiten Stufe werden wieder Two-State-Prädiktoren eingesetzt, die auch durch beliebige k-State-Prädiktoren ersetzt werden können. Die lokalen zweistufigen Zustandsprädiktoren entsprechen den PAp-Prädiktoren der Sprungvorhersage, die pro Sprungbefehl ein Schieberegister sowie eine Musterverlaufstabelle verwenden.

Sei $\mathcal{C}$ die Menge der Kontexte mit $|\mathcal{C}| = n$ und sei r die Ordnung, dann gibt es $(n-1)^r$ Muster, da der Kontext, für den der lokale zweistufige Prädiktor betrachtet wird, nicht in einem der möglichen Muster auftreten kann.

Beispiel. *Location Prediction*

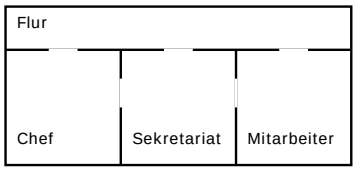


Abbildung 5.16: Flur, Sekretariat, Büro des Chefs und Büro des Mitarbeiters

Wir betrachten die Räume Flur F, Büro des Chefs C, Sekretariat S und Büro des Mitarbeiters M (siehe Abbildung 5.16). Sei

$$FCFSCFCMFSF$$

die abgelaufene Raumfolge einer Person. Die Person befindet sich im Flur, d.h. es wird der lokale zweistufige Zustandsprädiktor des Flurs zur Vorhersage benutzt. Die Folge der Nachfolgeräume des Flurs ist nun

$$CSCS$$

Sei die Ordnung $r = 3$, dann beinhaltet das Schieberegister das Muster $S\ C\ S$. Eine Vorhersage kann in diesem Fall noch nicht erfolgen, da dieses Muster das erste Mal aufgetreten ist.

Der lokale zweistufige Zustandsprädiktor mit Two-State-Prädiktor in der zweiten Stufe und Ordnung 0 entspricht dem einstufigen Two-State-Prädiktor. Statt eines Two-State-Prädiktors in der zweiten Stufe kann wie bei den globalen Prädiktoren auch ein beliebiger k-State-Prädiktor verwendet werden. Die lokalen zweistufigen Zustandsprädiktoren entsprechen den PAp-Prädiktoren der Sprungvorhersage, die pro Sprungbefehl ein Schieberegister und eine Musterverlaufstabelle verwenden.

Speicher- und Rechenaufwand. Sei r die Ordnung und sei n die Anzahl der möglichen Kontexte, d.h. $|\mathcal{C}| = n$, dann müssen für jeden Kontext $(n-1)^r$ Muster gespeichert werden. Der Rechenaufwand besteht zunächst aus der Selektion des lokalen zweistufigen Prädiktors entsprechend dem aktuellen Kontext. Dann muss das passende Muster in der Musterverlaufstabelle des selektierten Prädiktors gesucht sowie der Zustand des zugehörigen Two-State-Prädiktors gelesen und angepasst werden.

Vorteile. Mit einer kleineren Ordnung können die lokalen zweistufigen Prädiktoren auch längere globale Muster erkennen.

Nachteile. Der Anlernprozess ist sehr lang. Zusammenhängende Folgen von Kontexten werden unter Umständen nicht erkannt.

5.3 Markov-Prädiktoren

Die zweistufigen Zustandsprädiktoren sind den Markov-Prädiktoren aus dem Bereich der Datenkomprimierung ähnlich [CCM96] (siehe Kapitel **??**). Hier werden

in der zweiten Stufe anstatt des Two-State-Prädiktors die Häufigkeiten der aufgetretenen Nachfolgekontexte verwendet. Vorhergesagt wird dann der Kontext mit der größten Häufigkeit. Da der Unterschied zu den zweistufigen Zustandsprädiktoren die zweite Stufe ist, kann man die Markov-Prädiktoren wiederum mit lokalen und globalen Folgen von Kontexten verwenden.

Beispiel. *Location Prediction*

Als Beispiel soll ein Markov-Prädiktor betrachtet werden, der globale Folgen von Kontexten verwendet. Wir betrachten wieder das Beispiel des globalen zweistufigen Zustandsprädiktor (siehe Abbildung 5.14) mit der abgelaufenen Raumfolge

$$FSCFCSFSFCSFCS$$

und der Ordnung $r = 3$. Den entsprechenden Markov-Prädiktor mit Schieberegister und Musterverlaufstabelle zeigt Abbildung 5.17. Die Vorhersage ist, dass die Person als nächstes den Flur betreten wird.

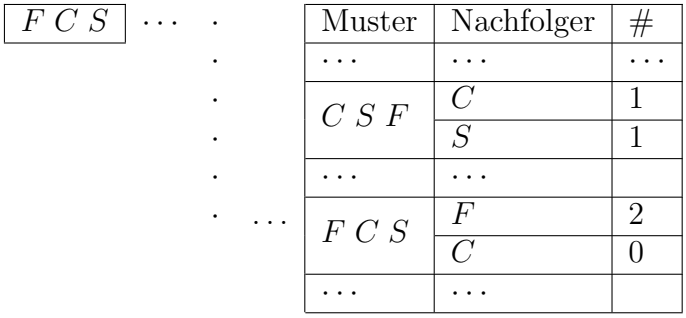


Abbildung 5.17: Markov-Prädiktor

Speicher- und Rechenaufwand. Der Speicheraufwand ist abhängig von der Anzahl der möglichen Kontexte und von der gewählten Ordnung. Sei die Anzahl der Kontexte n ($|\mathcal{C}| = n$) und die Ordnung r, dann müssen $n \cdot (n-1)^{r-1}$ Muster für den globalen Prädiktor gespeichert werden. In der zweiten Stufe müssen zu jedem Muster dann die vom letzten Kontext des Musters möglichen Nachfolgekontexte mit den zugehörigen Häufigkeiten gespeichert werden. Da der letzte Kontext im Muster selbst nicht als Nachfolgekontext auftreten kann, müssen für jedes Muster maximal $n - 1$ Häufigkeiten gespeichert werden. Insgesamt sind dann $n \cdot (n-1)^{r-1} \cdot (n-1) = n \cdot (n-1)^r$ Häufigkeiten für den globalen Prädiktor zu speichern. Für den lokalen Markov-Prädiktor entspricht die Anzahl der zu speichernden Muster dem lokalen zweistufigen Zustandsprädiktor. In diesem Fall müssen dann Häufigkeiten entsprechend der Anzahl der möglichen Nachfolgekontexte gespeichert werden. Der Rechenaufwand setzt sich aus der Suche nach dem

aktuellen Muster in der Musterverlaufstabelle sowie dem Auslesen und Anpassen der Häufigkeiten zusammen. Bei den lokalen Markov-Prädiktoren muss vorab entsprechend dem aktuellen Kontext noch der lokale Prädiktor selektiert werden.

Vorteile. Ein Vorteil ist auch hier das Erlernen komplizierter Muster. Durch die separate Behandlung der Muster können keine Interferenzen auftreten.

Nachteile. Nach sehr vielen Durchläufen eines Muster mit dem gleichen Nachfolgekontext benötigt der Markov-Prädiktor für das Umlernen sehr lange. Wenn zum Beispiel 1000 mal nach einem Muster der Kontext A eintritt, muss auch 1000 mal nach diesem Muster ein anderer Kontext eintreten, damit der andere Kontext vorhersagt wird.

5.4 Prediction by Partial Matching

Die zweistufigen Zustandsprädiktoren sowie die Markov-Prädiktoren können mittels *Prediction by Partial Matching* (PPM) erweitert werden (siehe Kapitel 4). Dabei wird statt der festen Ordnung in der ersten Stufe eine maximale Ordnung m gewählt. Dann wird mit dieser maximalen Ordnung m ein Muster entsprechend der letzten m Kontexte gesucht. Wird kein Muster der Länge m gefunden, wird das Muster der Länge $m-1$ gesucht, d.h. die letzten $m-1$ Kontexte. Dieser Prozess wird durchgeführt bis die Ordnung 1 erreicht ist. Wird mit Ordnung 1 kein Muster gefunden, wird im Fall der lokalen zweistufigen Prädiktoren eine Vorhersage mit Ordnung 0 gemacht. Ordnung 0 entspricht im Fall der lokalen zweistufigen Prädiktoren den einstufigen Prädiktoren, die nur lokal betrachtet wurden, da ohne das Wissen des aktuellen Kontexte keine sichere Vorhersage gemacht werden kann.

Im Fall der globalen zweistufigen Prädiktoren wird der PPM-Algorithmus mit Ordnung 1 gestoppt, d.h. wird kein Muster der Ordnung 1 gefunden, wird keine Vorhersage gemacht. Der Grund dafür ist der Zusammenhang zwischen den globalen zweistufigen Prädiktoren und den einstufigen Prädiktoren. Die globalen zweistufigen Prädiktoren mit Ordnung 1 entsprechen den lokalen einstufigen Prädiktoren.

Weiterhin wird eine vereinfachte Variante des PPM-Algorithmus vorgeschlagen, *Simple Prediction by Partial Matching* (SPPM). Hierbei wird eine maximale Ordnung m betrachtet. Wird kein Muster der Länge m gefunden, wird $m = 1$ gesetzt und ein Muster der Länge 1 gesucht. Wird ein solches Muster nicht gefunden, wird keine Vorhersage gemacht. Im Fall der lokalen zweistufigen Prädiktoren kann, nachdem kein passendes Muster der Länge m gefunden wurde, $m = 0$ gesetzt werden. Dies entspricht dann dem einstufigen Two-State-Prädiktor.

Beispiel. *Location Prediction*

Das Beispiel soll die Funktionsweise des PPM- bzw. SPPM-Algorithmus mit globalen zweistufigen Two-State-Prädiktoren verdeutlichen. Wir betrachten wieder die Raumfolge

$$FSCFCSFSFCSFCS$$

und beginnen mit einer maximalen Ordnung von 5. Die Vorgehensweise der Algorithmen zeigt Abbildung 5.18. Zunächst sucht der PPM-Algorithmus in der Musterverlaufstabelle des Two-State-Prädiktor der Ordnung 5 nach dem Muster, das den letzten fünf betretenen Räumen entspricht. Dieses tritt in der betrachteten Sequenz noch nicht auf, d.h. es ist noch nicht in der Tabelle vorhanden. Deshalb wird die Ordnung um eins verringert und der PPM-Algorithmus sucht das Muster der letzten vier betretenen Räume in der Musterverlaufstabelle des Two-State-Prädiktors der Ordnung 4. Auch dieses ist nicht vorhanden, so dass als nächstes in der Tabelle des Two-State-Prädiktors der Ordnung 3 gesucht wird. Die Musterverlaufstabelle entspricht der in Abbildung 5.15. Das Muster der letzten drei betretenen Räume ist in der Tabelle vorhanden. Somit kann die Vorhersage gemacht werden, dass die Person als nächstes den Flur betreten wird.

Im Fall des SPPM-Algorithmus wird zunächst auch nach dem Muster der letzten fünf Räume in der Musterverlaufstabelle des Two-State-Prädiktors der Ordnung 5 gesucht. Da dieses nicht vorhanden ist, setzt der Algorithmus die Ordnung auf 1. Der Prädiktor der Ordnung 1 muss nun in seiner Musterverlaufstabelle den zuletzt betretenen Raum S finden. Da S schon zuvor aufgetreten ist, kann die Vorhersage getroffen werden, dass der Flur der nächste Raum ist, den die Person betreten wird.

Speicher- und Rechenaufwand. Beim PPM-Algorithmus mit Two-State-Prädiktoren und maximaler Ordnung m sind die Speicherkosten aller Two-State-Prädiktoren mit Ordnung 1 bis Ordnung m nötig. Der SPPM-Algorithmus mit maximaler Ordnung m verbraucht nur den Speicher des Two-State-Prädiktors mit Ordnung 1 und Ordnung m. Bei Verwendung von Markov-Prädiktoren bauen sich die Kosten auch aus den einzelnen Speicherkosten der Markov-Prädiktoren auf. Der Rechenaufwand für die Vorhersage des PPM-Algorithmus besteht aus der Suche des Musters der Länge m in der Musterverlaufstabelle des Prädiktors der Ordnung m. Falls dieses nicht vorhanden ist, muss das Muster der Länge $m-1$ in der Musterverlaufstabelle des Prädiktors der Ordnung $m-1$ gesucht werden usw. Wird ein Muster gefunden, wird der Zustand des Two-State-Prädiktor bzw. die Häufigkeiten des Markov-Prädiktors für dieses Muster ausgelesen. Da alle verwendeten Prädiktoren angepasst werden müssen, besteht der Rechenaufwand für das Anpassen des PPM- bzw. SPPM-Algorithmus aus dem Aufwand für das Anpassen aller verwendeten Prädiktoren.

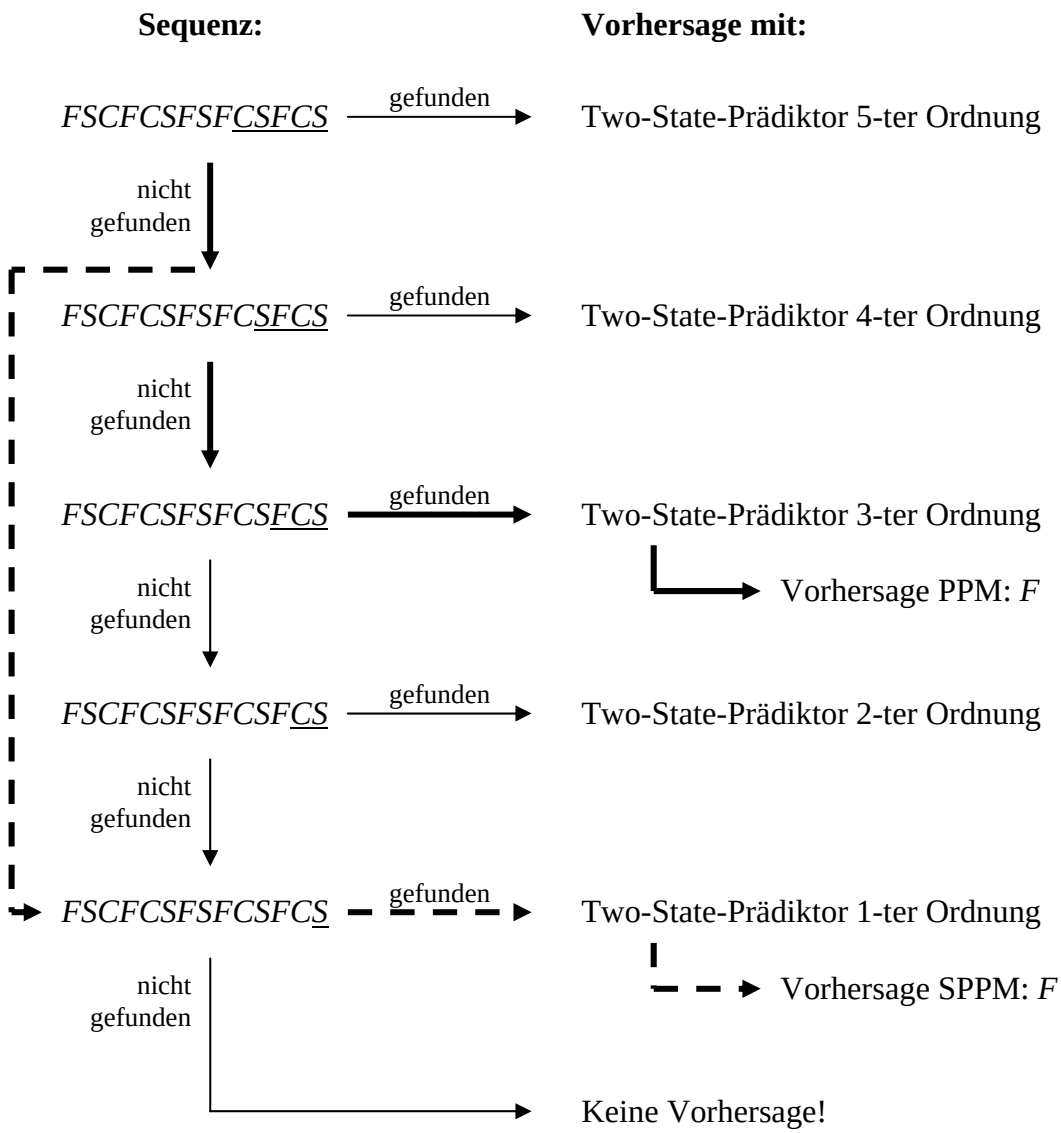


Abbildung 5.18: PPM- bzw. SPPM-Algorithmus 5-ter Ordnung mit zweistufigen Two-State-Prädiktoren

Vorteile. PPM- und SPPM-Algorithmus vereinen die Vorteile kurzer Ordnungen und die Vorteile langer Ordnungen, d.h. ein schnelles Anlernen sowie das Erlernen von komplizierten Mustern ist jetzt möglich.

Nachteile. Das Umlernen ist unter Umständen sehr langsam bei Verwendung der Markov-Prädiktoren. Der Speicher- und Rechenaufwand ist gegenüber den einzelnen Prädiktoren sehr hoch.

5.5 Evaluierung mit Augsburg Benchmarks

Im Folgenden werden die vorgestellten Zustandsprädiktoren und die entsprechenden Markov-Prädiktoren mit Hilfe der Augsburg Benchmarks bewertet und verglichen. Für beide Verfahren werden jeweils die Ordnungen 1, 2, 3, 4 und 5 sowie die PPM- und SPPM-Variante mit einer maximalen Ordnung von 5 betrachtet. Außerdem soll der One-State-Prädiktor mit dem Two-State-Prädiktor verglichen werden. Um auf einer möglichst großen Datenbasis aufzubauen, werden die Sommerdaten gefolgt von den Herbstdaten für jede der vier Personen verwendet. Da die meisten der Räume nur über den Korridor verlassen werden können, wird dieser in der Evaluierung nicht berücksichtigt, d.h. für die Simulation der Prädiktoren wurde der Kontext *corridor* aus den Benchmarks entfernt.

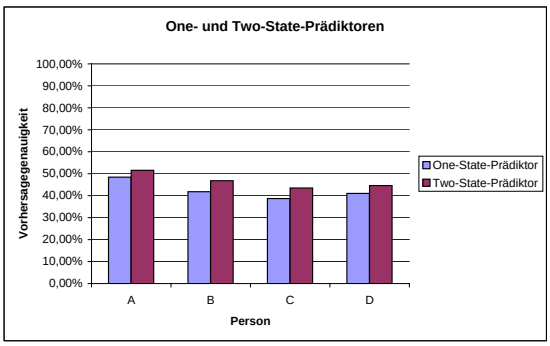


Abbildung 5.19: Vorhersagegenauigkeit der One- und Two-State-Prädiktoren - Augsburg Benchmarks

Abbildung 5.19 zeigt die Vorhersagegenauigkeit der One- und Two-State-Prädiktoren für alle vier Testpersonen. In allen Fällen ist der Two-State-Prädiktor besser als der One-State-Prädiktor, was auf die Umlerngeschwindigkeit zurückzuführen ist.

5.5.1 Genauigkeit

Die Diagramme in Abbildung 5.20 vergleichen die Vorhersagegenauigkeit der Zustands- und Markov-Prädiktoren für alle vier Testpersonen jeweils im lokalen und globalen Fall. Für die Ermittlung der Vorhersagegenauigkeit wurden nur gelernte Muster berücksichtigt. Es wird davon ausgegangen, dass bei jedem Kontextwechsel eine Vorhersage angefordert wird, d.h. die Anzahl der angeforderten Vorhersagen ist gleich der Anzahl der Kontextwechsel. Dabei wird die Anzahl der angeforderten Vorhersagen $v = v_l + v_n$ aufgeteilt in die Anzahl der lieferbaren Vorhersagen v_l und die nicht möglichen Vorhersagen v_n. Die lieferbaren Vorhersagen folgen auf Muster, die mindestens schon einmal aufgetreten und somit angelernt sind. Muster, die das erste Mal auftreten, erlauben keine sinnvolle Vorhersage.

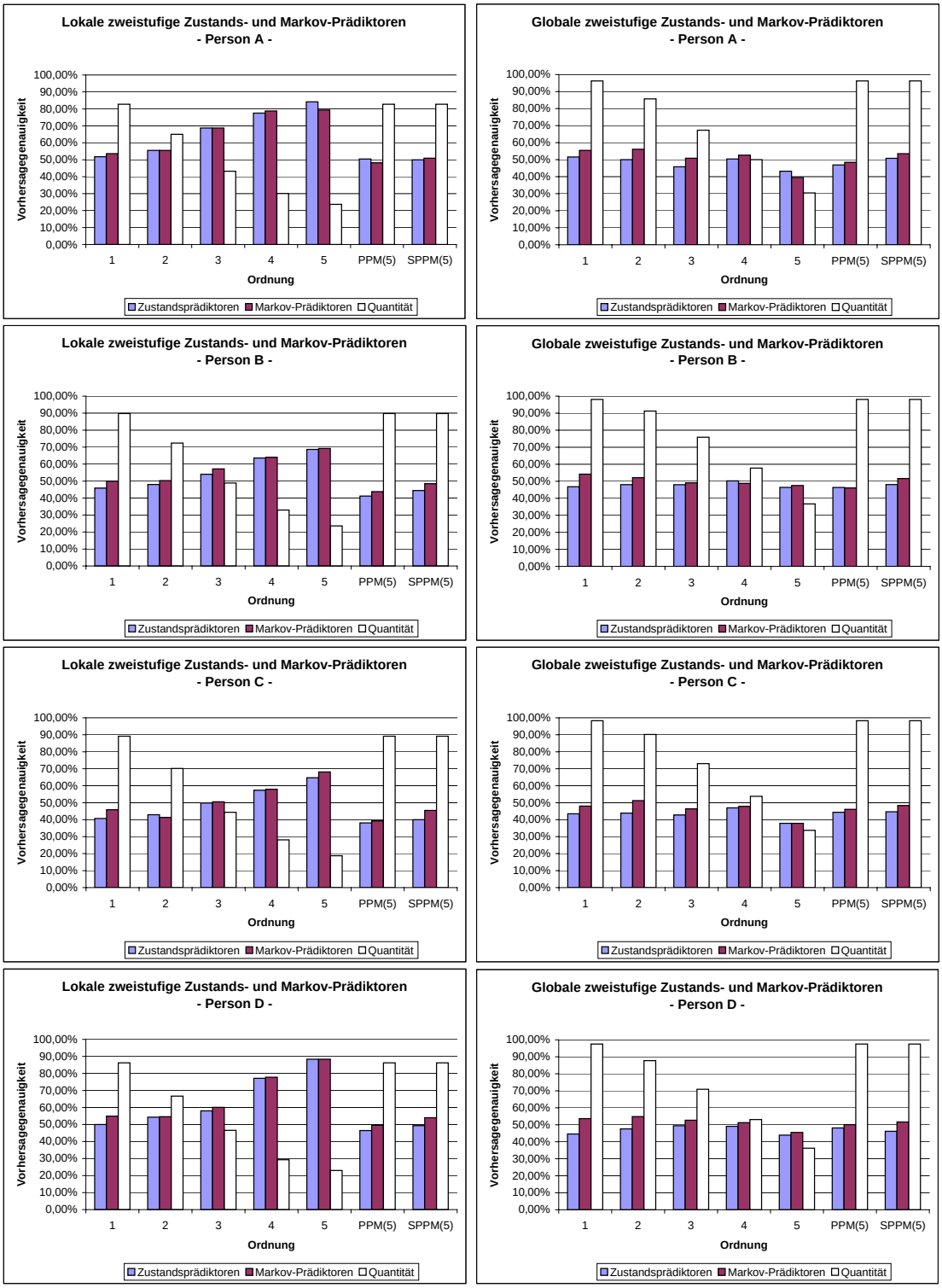


Abbildung 5.20: Vorhersagegenauigkeit der lokalen und globalen Prädiktoren - Augsburg Benchmarks

Hier ist somit keine Vorhersage möglich. Sei c die Anzahl der korrekten Vorhersagen, dann wird die Trefferwahrscheinlichkeit t der Prädiktoren mittels folgender Formel ermittelt:

$$t = \frac{c}{v_l}$$

Bei genauer Betrachtung der Messergebnisse ist festzustellen, dass im Allgemeinen weder die Zustandsprädiktoren besser als die Markov-Prädiktoren noch umgekehrt sind. Im globalen Fall stellt sich keine Ordnung als die optimale heraus. Interessant ist, dass in der Testumgebung die Prädiktoren mit den lokalen Mustern und Ordnung 5 bei allen vier Testpersonen sehr gute Ergebnisse liefern. Hier steigt mit höherer Ordnung auch die Trefferwahrscheinlichkeit.

Je höher die Ordnung der Prädiktoren ist, desto kleiner wird die Anzahl der lieferbaren Vorhersagen v_l, die auf Mustern beruhen, die mindestens schon einmal aufgetreten sind, also die angelernten Muster. Um dies zu verdeutlichen soll im Folgenden die Quantität q der lieferbaren Vorhersagen betrachtet werden:

$$q = \frac{v_l}{v}$$

Die Quantität der Prädiktoren ist in den Diagrammen in Abbildung 5.20 dargestellt. Hierbei gibt es keinen Unterschied zwischen Zustands- und Markov-Prädiktoren, da diese sich in der ersten Stufe nicht unterscheiden. Die Quantität der lokalen Prädiktoren ist immer schlechter als die der globalen Prädiktoren, da bei den lokalen Prädiktoren insgesamt mehr Muster möglich sind. Weiter hängt die Quantität auch von der Menge der Datenbasis ab. Die Quantität der Testperson A mit 533 Bewegungen ist schlechter als die der Testperson B mit 1430 Bewegungen.

5.5.2 Anlernverhalten

Die Anlernphase der Zustandsprädiktoren bzw. der Markov-Prädiktoren ist erst beendet, wenn alle Muster mindestens einmal aufgetreten sind. Da zum Beispiel bei einem Prädiktor der Ordnung $r = 5$ und $n = 15$ Kontexten bzw. Räumen

$$n \cdot (n-1)^{r-1} = 15 \cdot 14^4 = 576240$$

mögliche Muster existieren, wird die Anlernphase der Prädiktoren mindestens so viele Zyklen benötigen. Im Bereich der Kontextvorhersage werden solche Zahlen nie erreicht, d.h. die Zustandsprädiktoren bzw. Markov-Prädiktoren mit einer entsprechend großen Ordnung befinden sich immer in der Anlernphase. Deshalb

muss man vielmehr das Lernen der Muster einzeln betrachten. Sobald ein neues Muster auftritt, kann keine Aussage über den nächsten Kontext getroffen werden, der Prädiktor befindet sich in der Anlernphase für dieses Muster. Nach dem Eintreten des nächsten Kontextes ist der Prädiktor für dieses Muster angelernt. Beim nächsten Auftreten des Musters kann der Prädiktor eine Vorhersage über den nächsten Kontext treffen.

5.5.3 Umlernverhalten

In diesem Abschnitt soll das Umlernverhalten der Markov- und Zustandsprädiktoren verglichen werden. Dazu wurden wieder die lokalen und globalen Prädiktoren mit den Ordnungen 1, 2, 3, 4 und 5 sowie die PPM- und SPPM-Varianten mit der maximalen Ordnung 5 untersucht.

Um das Umlernverhalten zu analysieren, wird ein Umzug eines Angestellten in ein anderes Büro simuliert. Dazu werden die Sommer- und Herbstdaten von Testperson B gefolgt von den Sommer- und Herbstdaten von Testperson D verwendet. Der Korridor wird wiederum aus den Daten entfernt. Damit ergeben sich 717 Datensätze der Testperson B und 522 Datensätze der Testperson D plus ein Datensatz von B, der mit dem ersten Datensatz von D auf Korrektheit überprüft wird. Insgesamt sind dies also 1240 Datensätze, wobei die Änderung des Verhaltens nach 717 Bewegungen mit dem Umzug in das neue Büro beginnt.

Die Abbildungen 5.21 und 5.22 vergleichen die Vorhersagegenauigkeit der globalen Zustands- und globalen Markov-Prädiktoren mit Ordnung 1 sowie den SPPM-Varianten mit maximaler Ordnung 5. Die Messergebnisse der Prädiktoren mit anderen Ordnungen werden hier nicht gezeigt, da sich in diesen Fällen die Zustands- und Markov-Prädiktoren nicht wesentlich unterschieden. In einigen Fällen lag die Treffergenauigkeit der Markov-Prädiktoren über der der Zustandsprädiktoren und in einigen Fällen war es umgekehrt.

In Abbildung 5.21 ist die Vorhersagegenauigkeit über alle schon ausgewerteten Vorhersagen dargestellt. Dabei erkennt man, dass der Markov-Prädiktor nach 717 Vorhersagen einbricht und die Vorhersagegenauigkeit stetig abnimmt. Die Zustandsprädiktoren zeigen nur einen kurzen Einbruch. Da bei den Prädiktoren mit Ordnung 5, welche die SPPM-Variante verwenden, dieser Effekt nicht auftritt, ist der Einbruch der SPPM-Variante auch gegenüber den Prädiktoren mit Ordnung 1 auch nicht so stark.

Abbildung 5.22 zeigt die Vorhersagegenauigkeit der letzten 50 Vorhersagen der jeweiligen Prädiktoren. In den Diagrammen ist nach 717 Vorhersagen ein Einbruch der Zustands- sowie der Markov-Prädiktoren zu sehen. Die Zustandsprädiktoren erholen sich aber schneller als die Markov-Prädiktoren, die sogar bis auf 0% Vorhersagegenauigkeit fallen. Am Ende des Tests, d.h. nach 522 Vorhersagen, haben

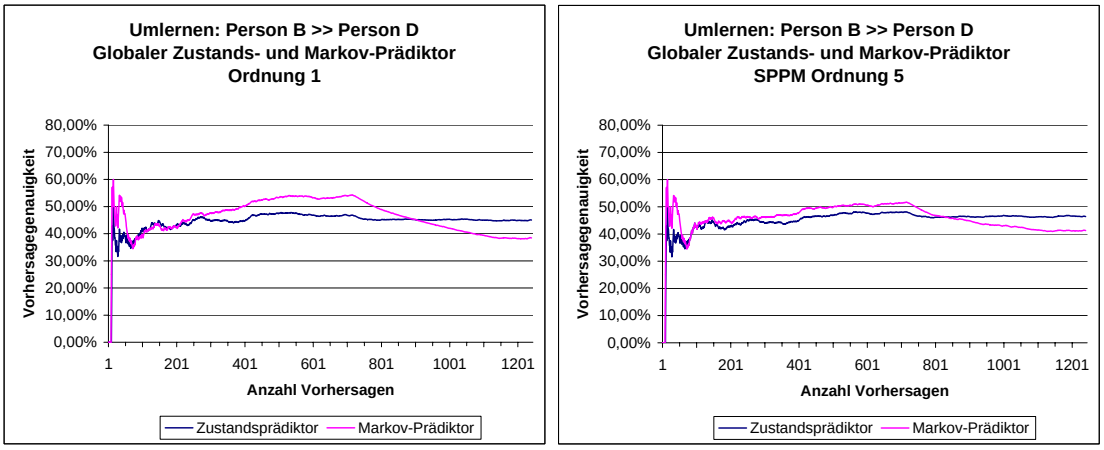


Abbildung 5.21: Umlernverhalten der Zustands- und Markov-Prädiktoren (Vorhersagegenauigkeit)

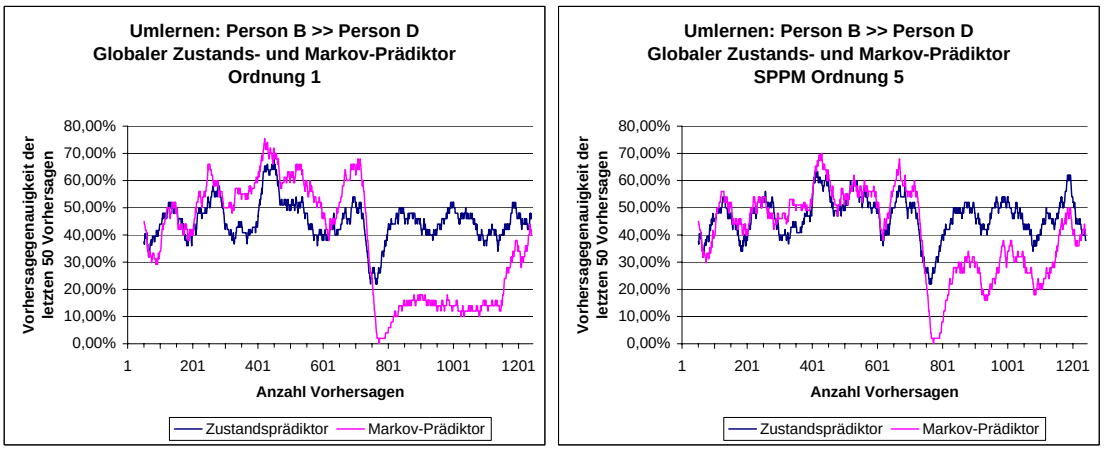


Abbildung 5.22: Umlernverhalten der Zustands- und Markov-Prädiktoren (Vorhersagegenauigkeit der letzten 50 Vorhersagen)

sich die Markov-Prädiktoren vollständig erholt und liefern annähernd die gleiche Vorhersagegenauigkeit wie die Zustandsprädiktoren.

5.5.4 Speicher- und Rechenaufwand

Für die Darstellung der Speicherkosten wird die minimale Anzahl an Bits berechnet, die für eine optimale Speicherung notwendig sind. Damit sind die Speicherkosten unter verschiedenen Verfahren besser vergleichbar, wenn für alle auf diese Art der Speicheraufwand ermittelt wird.

Für die Zustandsprädiktoren wachsen die Speicherkosten exponentiell mit der Ordnung. Sei n die Anzahl der verschiedenen Kontexte bzw. Räume und r die

Ordnung. Die Kosten C_{TS} für die Speicherung eines Zustandes des Two-State-Prädiktors bestehen aus den Kosten für die Speicherung der ID eines Kontextes und 1 Bit für die Unterscheidung zwischen schwachem und sicherem Zustand.

$$C_{TS} = C_{ID} + 1$$

Die Speicherkosten C_P für ein Muster bestehen aus den Kosten für das Muster selbst und den Kosten für den Two-State-Prädiktor.

$$C_P = r \cdot C_{ID} + C_{TS}$$

Es gibt n^r mögliche Muster, aber Muster wie $A - B - B$ können im globalen Fall nicht auftreten. Somit sind $n \cdot (n-1)^{r-1}$ verschiedene Muster möglich. Die gesamten Speicherkosten C_{state_global} eines globalen zweistufigen Two-State-Prädiktors können dann mit der folgenden Formel berechnet werden:

$$C_{state\_global} = n \cdot (n-1)^{r-1} \cdot (r \cdot C_{ID} + C_{TS})$$

Tabelle 5.1 zeigt die Speicherkosten eines globalen zweistufigen Two-State-Prädiktors für die Augsburg Benchmarks mit $n = 15$ verschiedenen Räumen. Somit werden für die Speicherung der ID eines Raumes 4 Bit benötigt ($C_{ID} = 4$). Die Tabelle zeigt wie die Speicherkosten exponentiell mit der Ordnung wachsen. Da aber das Maximum der Anzahl der Ortswechsel exklusive des Flurs bei Person A 266, bei Person B 717, bei Person C 634 und bei Person D 522 ist, kann es zum Beispiel bei Person B nur 713 Muster mit Ordnung 5 geben. Deshalb zeigt die letzte Spalte eine obere Grenze der Speicherkosten, wenn von einer Anzahl von maximal 500 Ortswechsel pro Person ausgegangen wird. Die obere Grenze der Anzahl der möglichen Muster ist dann $500 - (r-1)$.

Tabelle 5.1: Speicherkosten globaler zweistufiger Two-State-Prädiktor

Ordnung	# Muster	Speicher in Bit	obere Grenze # Muster	obere Grenze Speicher in Bit
$r = 1$	15	135		
$r = 2$	210	2.730		
$r = 3$	2940	49.980	498	8.466
$r = 4$	41160	864.360	497	10.437
$r = 5$	576240	14.406.000	496	12.400

Für einen lokalen zweistufigen Two-State-Prädiktor ändern sich gegenüber dem globalen Fall nur die Anzahl der Muster. Für jeden Kontext müssen $(n-1)^r$

Muster gespeichert werden. Insgesamt ergibt dies somit eine Anzahl von $n \cdot (n-1)^r$ Mustern. Die Speicherkosten eines lokalen zweistufigen Two-State-Prädiktor können dann mittels folgender Formel berechnet werden:

$$C_{state\_local} = n \cdot (n-1)^r \cdot (r \cdot C_{ID} + C_{TS})$$

Tabelle 5.2 zeigt die Speicherkosten eines lokalen zweistufigen Two-State-Prädiktors für die Augsburg Benchmarks mit $n = 15$ verschiedenen Räumen. Die Tabelle zeigt wiederum wie die Kosten exponentiell mit der Ordnung wachsen. Da hier auch Muster wie $A - B - B$ möglich sind, steigen die Kosten gegenüber dem globalen zweistufigen Two-State-Prädiktor an.

Tabelle 5.2: Speicherkosten lokaler zweistufiger Two-State-Prädiktor

Ordnung	# Muster	Speicher in Bit	obere Grenze # Muster	obere Grenze Speicher in Bit
$r = 1$	210	1.890		
$r = 2$	2.940	38.220	484	6.292
$r = 3$	41.160	782.040	469	7.973
$r = 4$	576.240	12.101.040	454	9.534
$r = 5$	8.067.360	201.684.000	439	10.975

Auch in diesem Fall ist die Anzahl der Muster durch die Anzahl an Ortswechsel begrenzt. Geht man von einer Gleichverteilung der Räume in wiederum 500 Ortswechsel aus, treten 11 Räume 33 mal und 4 Räume 34 mal in der Folge der betretenen Räume abzüglich des ersten Raumes auf ($11 \cdot 33 + 4 \cdot 34 = 499$). Somit können für jeden Raum nur Muster bestehend aus den 33 bzw. 34 folgenden Räumen auftreten, d.h. für Ordnung 5 können bei 33 Folgeräumen maximal 29 verschiedene Muster auftreten. Geht man davon aus, dass ein Raum die Folge dominiert, und somit maximal 250 mal auftritt, müssen für diesen Raum bei Ordnung 5 maximal 246 Muster gespeichert werden. In diesem Fall tritt nach allen anderen Räumen immer nur dieser Raum auf. Es muss somit für alle anderen Räume nur ein Muster gespeichert werden. Die gesamte Anzahl der zu speichernden Muster würde also in diesem Fall weit unter der maximalen Anzahl mit Gleichverteilung liegen. Tabelle 5.2 zeigt die Obergrenze der Muster und Speicherkosten mit angenommener Gleichverteilung der Räume innerhalb von 500 Ortswechsel.

Die Speicherkosten für einen Markov-Prädiktor können fast analog zu den Kosten eines Zustandsprädiktors ermittelt werden. Die Kosten eines globalen Markov-Prädiktors berechnen sich wie folgt:

$$C_{markov\_global} = n \cdot (n-1)^{r-1} \cdot (r \cdot C_{ID} + C_{freq})$$

Statt den Kosten für den Two-State-Prädiktor C_{2S} muss für jedes Muster eine Häufigkeitstabelle gespeichert werden, die aus der ID jedes möglichen nachfolgenden Ortes sowie der Häufigkeit jedes dieser Orte besteht. Somit ergibt sich für 8-Bit-Ganzzahlwerte als Häufigkeit

$$C_{freq} = (n-1) \cdot (C_{ID} + 8)$$

Tabelle 5.3 zeigt die maximalen Speicherkosten sowie eine obere Grenze der Speicherkosten bei 500 Ortswechsel eines globalen Markov-Prädiktor für die Augsburg Benchmarks mit $n = 15$ verschiedenen Räumen.

Tabelle 5.3: Speicherkosten globaler Markov-Prädiktor

Ordnung	# Muster	Speicher in Bit	obere Grenze # Muster	obere Grenze Speicher in Bit
$r = 1$	15	2.580		
$r = 2$	210	36.960		
$r = 3$	2940	529.200	498	89.640
$r = 4$	41160	7.573.440	497	91.448
$r = 5$	576240	108.333.120	496	93.248

Für den lokalen Markov-Prädiktor ändert sich gegenüber dem globalen nur die Anzahl der zu speichernden Muster analog zum Two-State-Prädiktor. Die Speicherkosten eines lokalen Markov-Prädiktor berechnen sich somit wie folgt:

$$C_{markov\_local} = n \cdot (n-1)^r \cdot (r \cdot C_{ID} + C_{freq})$$

Tabelle 5.4 zeigt die maximalen Speicherkosten sowie eine obere Grenze der Speicherkosten bei 500 Ortswechsel eines lokalen Markov-Prädiktors für die Augsburg Benchmarks mit $n = 15$ verschiedenen Räumen.

Tabelle 5.4: Speicherkosten lokaler Markov-Prädiktor

Ordnung	# Muster	Speicher in Bit	obere Grenze # Muster	obere Grenze Speicher in Bit
$r = 1$	210	36.120		
$r = 2$	2.940	517.440	484	85.184
$r = 3$	41.160	7.408.800	469	84.420
$r = 4$	576.240	106.028.160	454	82.536
$r = 5$	8.067.360	1.516.663.680	439	82.532

Wie in den vorherigen Abschnitten beschrieben besteht der Rechenaufwand aus der Suche nach dem Muster in der Musterverlaufstabelle sowie dem Auslesen und Anpassen der Zustände bzw. Häufigkeiten. Im lokalen Fall muss vorab noch der entsprechende Prädiktor selektiert werden. Die Prädiktoren wurden in Java und XML umgesetzt. Tabelle 5.5 zeigt die durchschnittliche Zeit pro Kontextwechsel für das gleichzeitige Anpassen aller untersuchten Prädiktoren (lokale und globale Zustandsprädiktoren mit Ordnung 1 bis 5 sowie lokale und globale Markov-Prädiktoren mit Ordnung 1 bis 5).

Tabelle 5.5: Rechenaufwand für das gleichzeitige Anpassen aller Prädiktoren

	Durchschnittliche Zeit pro Kontextwechsel
Person A	116 ms
Person B	260 ms
Person C	226 ms
Person D	193 ms

Die wenigsten Kontextwechsel erfolgen bei Person A, hier wird auch die geringste durchschnittliche Zeit für das Anpassen benötigt. Bei Person B treten die meisten Kontextwechsel auf und es wird die höchste durchschnittliche Zeit erzielt. Dies zeigt, mit mehr Kontextwechsel und somit mehr Einträgen in den Musterverlaufstabellen wird mehr Zeit für das Anpassen der Prädiktoren benötigt. Bei der Verwendung nur eines Prädiktors sind die Zeiten erheblich geringer. Die Zeit für eine Vorhersage, d.h. die Suche des Musters in der Musterverlaufstabelle und das Auslesen des Zustandes bzw. der Häufigkeit, ist dagegen sehr gering. Das Maximum der gemessenen Zeiten nur für die Vorhersage des nächsten Raumes betrug dabei 16 ms. Die Messungen wurden auf einem PC mit 3,4 GHz und einem Arbeitsspeicher von 2 GB durchgeführt.

5.6 Evaluierung mit Nokia Context Daten

In der Evaluierung mit den Nokia Context Daten soll die Zell-ID, der Gebiets-Code sowie die Benutzeraktivität vorhergesagt werden. Zunächst werden der One- und Two-State-Prädiktor betrachtet. Abbildung 5.23 zeigt die Vorhersagegenauigkeit dieser Prädiktoren.

Für die Ortsinformationen Zell-ID und Location Area Code zeigt sich, dass der Two-State-Prädiktor besser ist als der One-State-Prädiktor. Die Ursache ist wieder in der Umlerngeschwindigkeit zu finden. Im Fall der Aktivität des Benutzers verhält sich der One-State-Prädiktor besser als der Two-State-Prädiktor, was

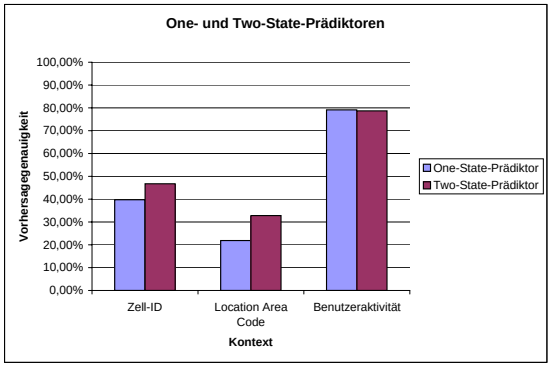


Abbildung 5.23: Vorhersagegenauigkeit der One- und Two-State-Prädiktoren - Nokia Context Daten

daran liegen könnte, dass auf eine Aktivität lange Sequenzen der gleichen Aktivitäten folgen. Zum Beispiel folgt auf die Aktivität A mehrmals hintereinander die Aktivität B. Jetzt ändert sich die auf A folgende Aktivität. Es folgt jetzt mehrmals die Aktivität C. Dies löst der One-State-Prädiktor besser. Dass auf eine Aktivität sehr oft die gleiche Aktivität folgt, erklärt auch die sehr guten Genauigkeiten von 79,1% und 78,7% bei der Vorhersage der nächsten Aktivität des Benutzers. Ein weiterer Grund ist die Verwendung der Bewegungsgeschwindigkeit als Aktivität. Um zum Beispiel von der Aktivität Stehen zur Aktivität Rennen zu gelangen, müssen zunächst die Zwischenstufen erreicht werden. Diese Regelmäßigkeiten sind bei der Zell-ID und dem Location Area Code nicht vorhanden, was zu den schlechten Vorhersagegenauigkeiten führt. Interessant ist, dass der grobkörnige Location Area Code wesentlich schlechter ist, als die feinkörnige Zell-ID.

5.6.1 Genauigkeit

Abbildung 5.24 zeigt die Vorhersagegenauigkeit der Zustands- und Markov-Prädiktoren für die Zell-ID, den Location Area Code und die Benutzeraktivität. Die Prädiktoren werden jeweils lokal und global betrachtet. Für die Berechnung der Genauigkeit wurden wiederum nur die schon bekannten Muster berücksichtigt. Desweiteren wurde auch die Quantität gemessen.

Die Diagramme zeigen, dass weder die Zustands- noch die Markov-Prädiktoren bessere Ergebnisse liefern. Bei Betrachtung der lokalen Prädiktoren ist festzustellen, dass bei langen Kontextsequenzen (Zell-ID 2317, Aktivität 2092) mit größerer Ordnung auch die Genauigkeit wächst. Bei kürzerer Sequenz (Location Area Code 207) ist dieses Verhalten nicht festzustellen. Im globalen Fall zeigen die Prädiktoren bei allen drei Kontexten, dass mit größerer Ordnung die Genauigkeit wächst. Im globalen Fall zeigen die Prädiktoren mit Ordnung 1 ein

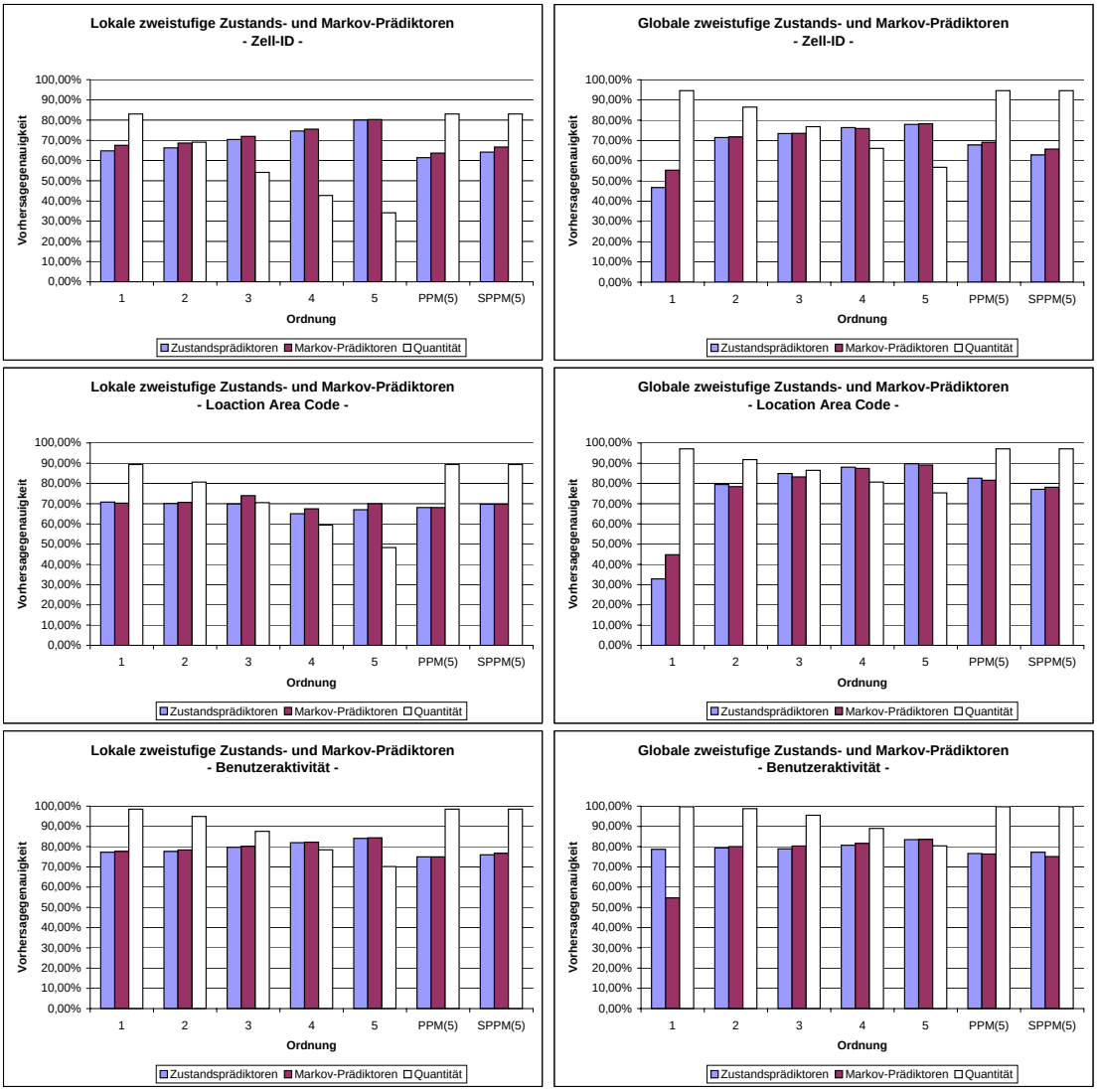


Abbildung 5.24: Vorhersagegenauigkeit der lokalen und globalen Prädiktoren - Nokia Context Daten

interessantes Verhalten. Für die Ortsinformationen Zell-ID und Location Area Code haben die Markov-Prädiktoren eine viel größere Vorhersagegenauigkeit als die State-Prädiktoren. Dies kann daran liegen, dass ein Mobiltelefon sich zwischen zwei Mobilfunkzellen oder Gebieten befindet und die gemessene Zell-ID zwischen diesen beiden Zellen pendelt, wie in der Einführung der Nokia Context Daten schon erwähnt wurde. Bei der Messung der Benutzeraktivität ist aber der State-Prädiktor mit Ordnung 1 wesentlich besser als der Markov-Prädiktor mit Ordnung 1. Ein Grund kann hier ein häufiges Durchlaufen der selben Sequenz von Aktivitäten am Anfang sein. Nach einer Änderung dieser Aktivitäten benötigt der Markov-Prädiktor sehr lange zum Umlernen.

Die Quantität zeigt das erwartete Verhalten. Sie fällt mit größerer Ordnung. Für die lokalen Prädiktoren ist sie kleiner als für die globalen Prädiktoren, da es im lokalen Fall mehr Muster zu erlernen gibt als im globalen Fall. Ein weiterer Zusammenhang der Quantität besteht mit der Anzahl der möglichen Kontexte. Bei ungefähr gleicher Sequenzlänge kann die Zell-ID 125 verschiedene Werte annehmen, die Benutzeraktivität aber nur 5 verschiedene Werte, d.h. für die Zell-ID gibt es mehr Muster zu erlernen als für die Benutzeraktivität. Somit ist die Quantität für die Zell-ID kleiner als für die Benutzeraktivität. Eine weitere offensichtliche Beziehung besteht zwischen Quantität und Sequenzlänge. Betrachtet man die Sequenzlänge des Location Area Code (207) und die Sequenzlänge der Benutzeraktivität (2092), so ist bei annähernd gleicher Anzahl der möglichen Werte (Location Area Code 6, Aktivität 5) die Quantität beim Location Area Code kleiner als bei der Benutzeraktivität. Der Grund liegt darin, dass beim Location Area Code mit einer Sequenzlänge von 207 der Anteil der Muster, die das erste Mal auftreten, größer ist als bei der Benutzeraktivität mit einer Sequenzlänge von 2092.

5.6.2 Anlern- und Umlernverhalten

Erst wenn alle Muster einmal aufgetreten sind, ist die Anlernphase beendet. Die Anzahl der möglichen Muster steigt mit der Ordnung und mit der Größe des Wertebereiches des betrachteten Kontextes. Im Folgenden ist die maximale Anzahl der Muster für die drei Kontexte Zell-ID mit $n = 125$ möglichen Werten, Location Area Code mit $n = 6$ möglichen Werten sowie Aktivität mit $n = 5$ Werten bei Betrachtung der globalen Prädiktoren mit Ordnung $r = 5$ berechnet:

Zell-ID: $n \cdot (n-1)^{r-1} = 125 \cdot 124^4 = 29\,552\,672\,000$

Location Area Code: $n \cdot (n-1)^{r-1} = 6 \cdot 5^4 = 3\,750$

Aktivität: $n \cdot (n-1)^{r-1} = 5 \cdot 4^4 = 1\,280$

Diese Zahlen werden natürlich im Bereich der Kontextvorhersage nie erreicht. Desweiteren sind in diesen Zahlen natürlich auch Muster enthalten, die nie auftreten können. Zum Beispiel hat eine Zelle nicht alle anderen Zellen als Nachbarn, was aber in der Berechnung nicht beachtet wurde. Deshalb muss das Anlernen jedes einzelnen Musters betrachtet werden. Jedes Muster muss nur einmal auftreten. Beim nächsten Auftreten dieses Musters ist die Anlernphase beendet, da nun aufgrund des Kontextes, der beim ersten Auftreten folgte, eine Vorhersage gemacht werden kann.

Das Umlernen wurde mit den Nokia Context Daten nicht simuliert. Aber im vorigen Abschnitt wurde schon der State- und Markov-Prädiktor mit Ordnung 1 bei der Messung der Vorhersage der Benutzeraktivität erwähnt. Hier spielt vermutlich das schlechte Umlernverhalten der Markov-Prädiktoren eine Rolle.

5.6.3 Speicher- und Rechenaufwand

Für die Berechnung der Speicherkosten des globalen zweistufigen Two-State-Prädiktors wurde die Formel schon bei der Evaluierung der Augsburg Benchmarks ermittelt. Für die Zell-ID wären die theoretischen Speicherkosten extrem hoch. Bei Betrachtung der maximalen Anzahl an Zellenwechsel verringern sich die Speicherkosten. Der Rechenaufwand für die Anpassung der Prädiktoren ist für den Kontext Zell-ID am größten, da hier die Musterverlaufstabellen aufgrund der theoretisch möglichen Muster am größten werden können. Die Zeit für eine Vorhersage ist wie bei den Augsburg Benchmarks sehr gering.

6 Zuverlässigkeitsschätzung

Ubiquitäre Systeme sollen dem Menschen alltägliche Aufgaben abnehmen, um so den Komfort für ihn zu erhöhen. Durch die Kontextvorhersage soll es dem System möglich sein, sich vorausschauend auf den Menschen einzustellen und proaktiv zu handeln. Da Vorhersagen nie zu 100 Prozent korrekt sind, ist es in vielen Fällen besser keine Voraussage anstatt einer falschen Voraussage anzubieten. Aus diesem Grund ist eine Zuverlässigkeitsschätzung des Vorhersageergebnisses unerlässlich.

Da die Zustandsprädiktoren von der Sprungvorhersage aus dem Bereich der Prozessorarchitektur motiviert ist, sind die Zuverlässigkeitsverfahren von Sprungvorhersageergebnissen für diese Prädiktoren von großem Interesse. Jacobson et al. [JRS96] verwenden zusätzlich zur Sprungvorhersage eine Tabelle mit Schieberegistern, den so genannten n-Bit *Correct/Incorrect Register*. Der Zugriff auf die Tabelle kann zum Beispiel auch wie bei der Sprungvorhersage erfolgen. Für jede korrekte Vorhersage wird eine 0 eingetragen, für eine falsche eine 1. Mittels einer so genannten *Reduction Function* wird ermittelt, ob die Zuverlässigkeit hoch oder niedrig einzustufen ist. Eine entsprechende Funktion ist zum Beispiel, die Anzahl der 1 mit einem Schwellenwert zu vergleichen. Liegt diese Anzahl über dem Schwellenwert, d.h. der Anteil der Fehlvorhersagen ist hoch, wird die Zuverlässigkeit als niedrig eingestuft, andernfalls als hoch. Jacobsen et al. schlagen weiterhin ein zweistufiges Verfahren vor, welches auf zwei der vorab erwähnten Tabellen aufbaut.

Tyson et al. [TLF97] stellten bei einer Untersuchung der Verteilung der Fehlvorhersagen unter den Sprüngen fest, dass eine kleine Zahl von Sprungbefehlen den Hauptteil der Fehlvorhersagen ausmacht. Sie schlugen einen Zuverlässigkeitsschätzer vor, der einer bestimmten Anzahl von Sprungbefehlen eine hohe Zuverlässigkeit zuweist, und den übrigen eine niedrige Zuverlässigkeit. Dieser Zuverlässigkeitsschätzer ist ohne zusätzlichen Hardwareaufwand realisierbar.

Smith [Smi81] schlug einen Zuverlässigkeitsschätzer vor, der sich die Konstruktion eines Sättigungszähler zu Nutzen macht. Grunwald et al. [GKMP98] untersuchten zusätzlich zu den schon genannten Techniken ein Verfahren, welches die korrekten Vorhersagen für jeden Sprungbefehl zählt und mit einem Schwellenwert zuverlässige Sprünge bestimmt. Liegt der Anteil der korrekten Vorhersagen für einen Sprungbefehl oberhalb des Schwellenwertes, hat der Sprungbefehl eine hohe Zuverlässigkeit, andernfalls eine niedrige.

6.1 Verfahren der Zuverlässigkeitsschätzung

Im Folgenden werden ein statisches und drei dynamische Verfahren der Zuverlässigkeitsschätzung der Kontextvorhersage beschreiben [PBTU04].

6.1.1 Statische Zuverlässigkeit

Die Idee von Tyson et al. [TLF97], dass es eine bestimmte Menge von unzuverlässigen Sprungbefehlen gibt, soll hier aufgegriffen werden. Dabei wird angenommen, dass es Kontexte bzw. Sequenzen von Kontexten gibt, für die es unmöglich oder nicht sinnvoll ist eine Vorhersage über den nächsten Kontext zu machen.

Das statische Zuverlässigkeitsverfahren ist unabhängig vom verwendeten Vorhersagealgorithmus. Sei $\mathcal{C}$ die Menge der Kontexte. Wir betrachten die Menge der zuverlässigen Kontexte $\mathcal{C}_{con}$ und die Menge der unzuverlässigen Kontexte $\mathcal{C}_{unc}$. Wird ein Kontext als unzuverlässig eingestuft, wird dieser zur Menge $\mathcal{C}_{unc}$ hinzugefügt. Für die Menge der zuverlässigen Kontexte gilt dann $\mathcal{C}_{con} = \mathcal{C} \setminus \mathcal{C}_{unc}$. Diese Einteilung findet vor der Initialisierungsphase des Prädiktors statt. Sei der aktuelle Kontext c, dann gilt für die Vorhersage:

$c \in \mathcal{C}_{con}$: Anbieten des Vorhersageergebnisses
$c \in \mathcal{C}_{unc}$: Zurückhalten des Vorhersageergebnisses

Ist der aktuelle Kontext in der Menge der zuverlässigen Kontexte, wird die vom verwendeten Algorithmus gelieferte Vorhersage angeboten. Gehört der aktuelle Kontext aber zur Menge der unzuverlässigen Kontexte, wird das Vorhersageergebnis zurückgehalten.

Für dieses Verfahren ist kein zusätzlicher Speicher- und Rechenaufwand nötig. Vorab muss lediglich eine Analyse durchgeführt werden, welche Kontexte unzuverlässige Vorhersagen liefern.

6.1.2 Sicherer Zustand

Das Verfahren *Sicherer Zustand* kann nur bei Prädiktoren eingesetzt werden, die auf dem Two-State-Prädiktor aufbauen. Der Two-State-Prädiktor hat für jeden Kontext zwei Zustände, einen schwachen und einen sicheren Zustand, von denen dieser als nächster Kontext vorhergesagt wird. Der Prädiktor befindet sich schon nach einem einmaligen Auftreten eines Kontextes im schwachen Zustand, in dem dieser Kontext vorhergesagt wird. In einen sicheren Zustand wechselt

der Prädiktor erst nach einem wiederholten Auftreten dieses Kontextes. Die Idee dieses Verfahren besteht nun lediglich darin, eine Vorhersage nur in einem sicheren Zustand anzubieten. Abbildung 6.1 zeigt den Two-State-Prädiktor für die drei Kontexte A, B und C. Bei Verwendung des Verfahrens *Sicherer Zustand* wird eine Vorhersage des entsprechenden Kontextes nur in den sicheren Zuständen (doppelt umrahmt) angeboten.

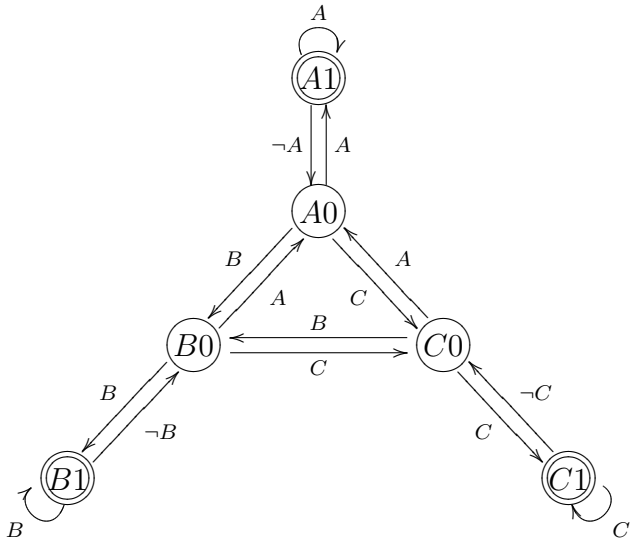


Abbildung 6.1: Verfahren *Sicherer Zustand* - Vorhersage nur in den sicheren Zuständen $A1$, $B1$ und $C1$

Diese Verfahren kann für den k-State-Prädiktor erweitert werden. Dabei muss ein b mit $1 \leq b \leq k$ festgelegt werden. Der Wert b gibt dann die Grenze zwischen schwachen und sicheren Zuständen an. Für den Two-State-Prädiktor ($k = 2$) gilt $b = 1$. Sei $1 \leq s \leq k$ der Wert des aktuellen Zustandes des k-State-Prädiktors, dann gilt:

$s \geq b$: Anbieten des Vorhersageergebnisses
$s < b$: Zurückhalten des Vorhersageergebnisses

Die Arbeitsweise des Verfahrens *Sicherer Zustand* wird im Folgenden am Beispiel eines zweistufigen Two-State-Prädiktors mit Ordnung n beschrieben. Der Prädiktor startet mit einer leeren Musterverlaufstabelle. Tritt ein Muster der Länge n das erste Mal auf, kann keine Vorhersage gemacht werden. Folgt jetzt der Kontext C auf das Muster, wird der Two-State-Prädiktor für dieses Muster mit dem schwachen Zustand für den Kontext C initialisiert. Beim nächsten Auftreten dieses Musters könnte nun eine Vorhersage angeboten werden. Da der Two-State-Prädiktor für dieses Muster sich aber in einem schwachen Zustand

befindet, wird diese Vorhersage nicht angeboten. Wenn jetzt auf dieses Muster wieder der Kontext C folgt, wechselt der Prädiktor in den sicheren Zustand für den Kontext C. Tritt jetzt dieses Muster das nächste Mal auf, wird die Vorhersage angeboten, dass der Kontext C als nächstes eintritt.

Folgt auf dieses Muster irgendwann nicht mehr der Kontext C, wechselt der Two-State-Prädiktor in den schwachen Zustand für den Kontext C. Nun wird beim nächsten Auftreten wieder keine Vorhersage angeboten, da der Prädiktor sich in einem schwachen Zustand befindet.

Auch für dieses Verfahren ist kein zusätzlicher Speicheraufwand nötig. Der zusätzliche Rechenaufwand besteht darin, festzustellen, ob sich der Prädiktor in einem sicheren Zustand befindet.

6.1.3 Schwellenwert-Verfahren

Die Zuverlässigkeitsanalyse mittels Schwellenwert-Verfahren greift die Idee von Grunwald et al. [GKMP98] auf und ist unabhängig vom eingesetzten Vorhersagealgorithmus. Bei diesem Verfahren wird die bisherige Genauigkeit der Vorhersagen mit einem vorgegebenen Schwellenwert verglichen. Liegt die Genauigkeit über dem Schwellenwert wird die Vorhersage als zuverlässig angesehen und somit angeboten. Hierbei ist zu beachten, dass bei den zweistufigen Prädiktoren jeder Two-State-Prädiktor für sich betrachtet wird, d.h. für jedes Muster wird der entsprechende Two-State-Prädiktor einzeln auf seine Genauigkeit untersucht. Die Genauigkeit jedes einzelnen Two-State-Prädiktors wird nun mittels der Anzahl der korrekten und der falschen Vorhersagen dieses Prädiktors bestimmt, indem der Anteil der korrekten an der Gesamtzahl der Vorhersagen berechnet wird. Sei c die Anzahl der korrekten Vorhersagen (*correct*), i die Anzahl der falschen Vorhersagen (*incorrect*), sowie α der Schwellenwert, dann kann das Schwellenwert-Verfahren wie folgt beschrieben werden:

$$\frac{c}{c+i} \geq \alpha \quad : \quad \text{Anbieten des Vorhersageergebnisses}$$
$$\frac{c}{c+i} < \alpha \quad : \quad \text{Zurückhalten des Vorhersageergebnisses}$$

Bei den globalen zweistufigen Prädiktoren muss die Musterverlaufstabelle um die beiden Werte für korrekte und falsche Vorhersagen erweitert werden (siehe Abbildung 6.2). Dadurch ergibt sich eine leicht Erhöhung des Speicher- und Rechenaufwandes.

Der globale zweistufige Two-State-Prädiktor wird mit einer leeren Musterverlaufstabelle gestartet. Beim ersten Auftreten eines Musters kann noch keine Vorhersage gemacht werden. Folgt der Kontext c auf dieses Muster, wird der Two-

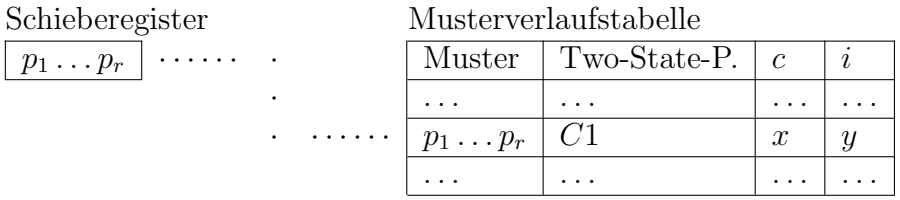


Abbildung 6.2: Globaler zweistufiger Two-State-Prädiktor mit Schwellenwert

State-Prädiktor entsprechend initialisiert, außerdem wird für die Anzahl der korrekten und die Anzahl der falschen Vorhersagen jeweils der Wert 0 eingetragen. Beim zweiten Auftreten des Musters kann noch keine Aussage über die Zuverlässigkeit gemacht werden, da noch keine Vorhersage verifiziert werden konnte. Auch können die Werte noch nicht in die Formel zur Bestimmung der Zuverlässigkeit eingesetzt werden, da der Nenner gleich 0 ist. Aus diesen Gründen wird nun angenommen, dass die Vorhersage zuverlässig ist. Diese wird somit angeboten. Erweist sich die Vorhersage als korrekt wird der Wert für die Anzahl der korrekten Vorhersagen um 1 erhöht. Andernfalls wird die Anzahl der falschen Vorhersagen inkrementiert. Weiterhin wird natürlich auch der Two-State-Prädiktor entsprechend angepasst. Beim nächsten Auftreten des Musters können nun die Werte in die Formel eingesetzt werden und mit dem Schwellenwert verglichen werden.

Ein Nachteil dieses Verfahren ist unter Umständen ein „ Festfahren“ oberhalb des Schwellenwertes. Das heißt wenn die vergangenen Vorhersagen alle korrekt waren, liegt die Genauigkeit weit oberhalb des Schwellenwertes. Folgen aber nun falsche Vorhersagen, kann es zu lange dauern bis der Schwellenwert wieder unterschritten wird. Somit wird der nicht zuverlässige Prädiktor noch als zuverlässig angesehen. Eine Abhilfe kann hier das Verfahren mit Zuverlässigkeitszähler schaffen.

6.1.4 Zuverlässigkeitszähler

Das Verfahren mit Zuverlässigkeitszähler entspricht dem Ansatz von Smith [Smi81] und ist auch unabhängig vom verwendeten Vorhersagealgorithmus. Hierbei wird nun die Genauigkeit vergangener Vorhersagen anhand eines Zählers mitprotokolliert. Der Zähler besteht aus $n + 1$ Zuständen (siehe Abbildung 6.3).

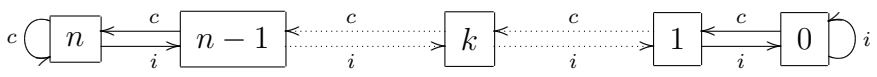


Abbildung 6.3: Zuverlässigkeitszähler - Zustandsgraph

Der Initialzustand des Zählers kann beliebig gewählt werden. Erweist sich eine Vorhersage als richtig (c = *correct*), wird der Zähler erhöht, d.h. der Zustands-

graph wechselt von Zustand s in den Zustand $s+1$. Ist die Vorhersage falsch (i = *incorrect*), wird in den Zustand $s-1$ gewechselt. Weiterhin gibt es einen Zustand k mit $0 \leq k \leq n+1$, für den gilt: Ist s größer oder gleich k, wird der Prädiktor als zuverlässig angenommen, andernfalls als nicht zuverlässig und es wird keine Vorhersage angeboten. Das Verfahren kann formell wie folgt beschrieben werden.

$s \geq k$: Anbieten des Vorhersageergebnisses
$s < k$: Zurückhalten des Vorhersageergebnisses

Auch bei diesem Verfahren wird für jeden Two-State-Prädiktor die Genauigkeit mittels des Zuverlässigkeitszählers separat bestimmt. Dazu muss die Musterverlaufstabelle um eine Spalte, die den Zustand dieses Zählers (cc = *confidence counter*) speichert, erweitert werden (siehe Abbildung 6.4). Dadurch ergibt sich eine leichte Erhöhung des Speicher- und Rechenaufwandes.

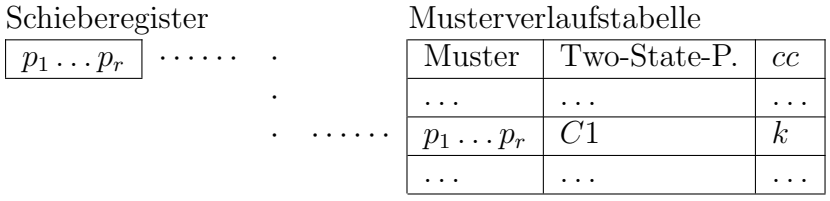


Abbildung 6.4: Globaler zweistufiger Two-State-Prädiktor mit Zuverlässigkeitszähler

Die Initialisierungsphase soll im Folgenden anhand des globalen zweistufigen Two-State-Prädiktor beschrieben werden. Beim Start des Prädiktors ist die Musterverlaufstabelle leer. Beim ersten Auftreten eines Musters kann noch keine Vorhersage gemacht werden. Folgt auf das Muster der Kontext c wird ein Two-State-Prädiktor für dieses Muster mit dem Kontext c initialisiert. Desweiteren wird für dieses Muster dem Zuverlässigkeitszähler ein Wert zwischen 0 und n zugewiesen, zum Beispiel der Wert k. Beim zweiten Auftreten dieses Muster kann nun schon anhand des Initialwertes des Zuverlässigkeitszählers über die Zuverlässigkeit des Two-State-Prädiktors für dieses Muster entschieden werden. Wurde als Initialwert k gewählt, wird die Vorhersage angeboten.

6.2 Evaluierung mit Augsburg Benchmarks

Bei der Evaluierung mit den Augsburg Benchmarks wurde wie in Kapitel 5 der Flur nicht berücksichtigt, da alle Räume nur über den Flur verlassen werden können. Desweiteren wurden nur die globalen zweistufigen Two-State-Prädiktoren von Ordnung 1 bis 5, sowie mit *Prediction by Partial Matching* (PPM)

und *Simple Prediction by Martial Matching* (SPPM) untersucht. Alle Prädiktoren wurden für jede Testperson mit den entsprechenden Sommerdaten gefolgt von den Herbstdaten evaluiert.

Zur Bewertung der Verfahren wird zunächst wieder die Treffergenauigkeit berechnet. Es wird dabei wie im letzten Kapitel angenommen, dass bei jedem Kontextwechsel eine Vorhersage angefordert wird, d.h. die Anzahl der angeforderten Vorhersagen v entspricht der Anzahl der Kontextwechsel. Diese teilen sich wieder in lieferbare Vorhersagen v_l und nicht mögliche Vorhersagen v_n aufgrund leerer Einträge in der Musterverlaufstabelle auf (siehe Kapitel 5). Die lieferbaren Vorhersagen werden nun unterteilt in die zuverlässigen Vorhersage v_z und die unzuverlässigen Vorhersagen v_u, d.h. $v = v_z + v_u + v_n$. Die Treffergenauigkeit t wird nun mit den zuverlässigen Vorhersage berechnet. Sei c die Anzahl der korrekten Vorhersagen, dann gilt

$$t = \frac{c}{v_z}$$

Damit wird die Quantität q mittels der zuverlässigen Vorhersagen berechnet:

$$q = \frac{v_z}{v}$$

Da die zuverlässigen Vorhersage nur einen Teil der lieferbaren Vorhersagen ausmachen, wird die Quantität unter Verwendung eines Zuverlässigkeitsverfahren gegenüber der Quantität ohne Zuverlässigkeit sinken.

Ein weiteres Bewertungkriterium der Zuverlässigkeitsverfahren stellt die Steigerung dar. Die Steigerung g der Treffergenauigkeit mit Zuverlässigkeitsanalyse t_{con} gegenüber der Treffergenauigkeit ohne Zuverlässigkeitsanalyse $t_{w/o}$ wird wie folgt verwendet:

$$g = \frac{t_{con} - t_{w/o}}{1 - t_{w/o}} \cdot 100$$

Die Steigerung gibt damit an, welcher prozentuale Anteil einer maximal möglichen Steigerung erreicht wurde. Dabei ist die maximal mögliche Steigerung bei einer Vorhersagegenauigkeit von 100% erreicht.

6.2.1 Statische Zuverlässigkeit

Bei den Augsburg Benchmarks sind die Kontexte nur die Räume, die eine Person betritt, d.h. für die statische Zuverlässigkeit muss untersucht werden, von welchen Räumen aus keine zuverlässige Vorhersage getroffen werden kann. Bei

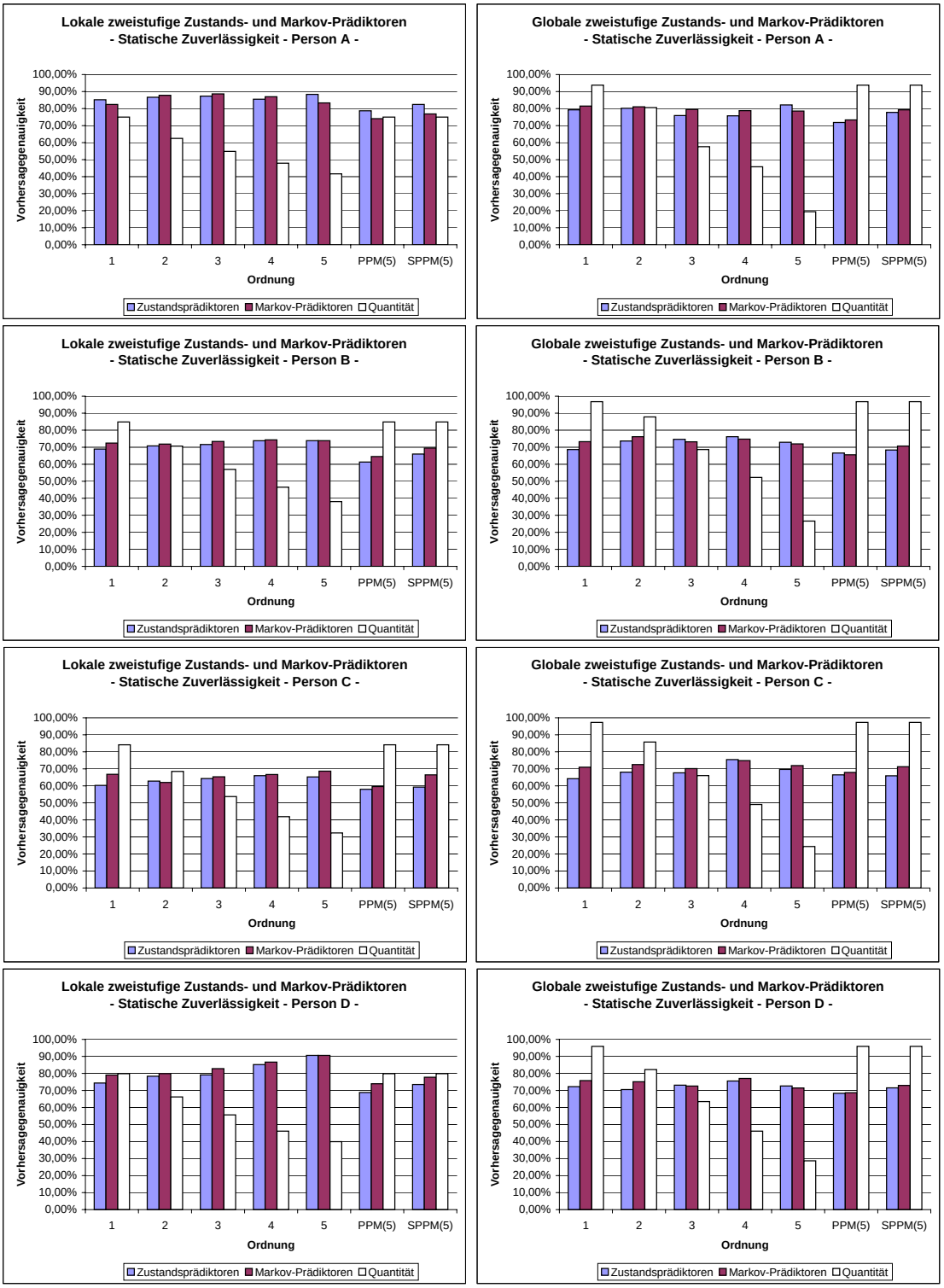


Abbildung 6.5: Lokale und globale Prädiktoren mit statischer Zuverlässigkeit - Augsburg Benchmarks

genauer Betrachtung der Benchmarks wurde festgestellt, dass eine Vorhersage aus dem eigenen Büro nur schwer möglich ist, da im eigenen Büro die meisten wiederkehrenden Bewegungsabläufe beginnen. Zusätzlich ist in der Anwendung der Smart Doorplates eine Vorhersage aus dem eigenen Büro nicht nötig. Aus diesen Gründen wurde das eigene Büro des jeweils untersuchten Angestellten in die Menge der unzuverlässigen Kontexte eingefügt. Dies bedeutet für die globalen Prädiktoren, dass für alle Muster, die als letzten Kontext das eigene Büro haben, keine Vorhersage angeboten wird. Für die lokalen Prädiktoren wird einfach die Tabelle mit den Nachfolgekontexten des eigenen Büros nicht geführt.

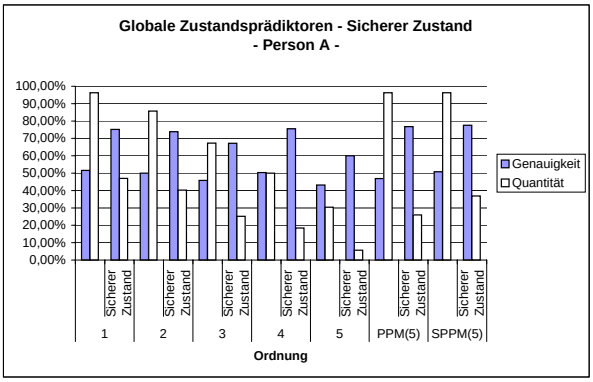
Für einen besseren Vergleich mit den Ergebnissen aus Kapitel 5 werden hier auch die Markov-Prädiktoren betrachtet. Abbildung 6.5 zeigt die Vorhersagegenauigkeiten der lokalen und globalen Zustands- sowie Markov-Prädiktoren mit den Ordnungen 1, 2, 3, 4 und 5 sowie die PPM- und SPPM-Varianten mit maximaler Ordnung 5 unter Verwendung der statischen Zuverlässigkeit. Im Vergleich mit Abbildung 5.20 in Abschnitt 5.5 ist eine deutliche Steigerung der Vorhersagegenauigkeiten festzustellen, was eine Wahl des eigenen Büros in die Menge der unzuverlässigen Kontexte bestärkt. Auffallend ist, dass die Quantität im lokalen Fall bei Verwendung der statischen Zuverlässigkeit mit kleiner Ordnung schlechter und mit großer Ordnung besser wird als ohne Verwendung der statischen Zuverlässigkeit. Der Grund dafür sind die jeweils zur Berechnung der Quantität verwendeten Mengen der angeforderten Vorhersagen v, die in beiden Fällen unterschiedlich sind. Bei Verwendung der statischen Zuverlässigkeit werden im eigenen Büro keine Vorhersagen angefordert.

6.2.2 Sicherer Zustand

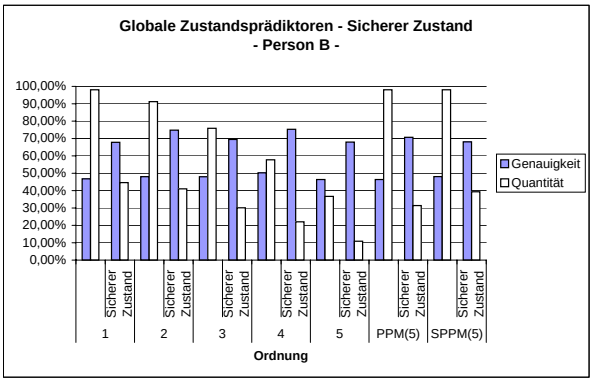
In Abbildung 6.6 sind die Messergebnisse der Evaluierung des Verfahren *Sicherer Zustand* für die Augsburg Benchmarks dargestellt. Für alle vier Personen wurde die Vorhersagegenauigkeit, die Quantität sowie die Steigerung gemessen. Es wurden nur die globalen zweistufigen Two-State-Prädiktoren untersucht.

Die Steigerung ist bei allen Prädiktoren immer positiv, d.h. die Vorhersagegenauigkeit verbessert sich immer unter Verwendung des Zuverlässigkeitsverfahrens *Sicherer Zustand*. Die größte Steigerung (59,59%) erreicht der globale zweistufige Two-State-Prädiktor mit Ordnung 1 für Person D. Der Grund für diese große Steigerung liegt darin, dass der Prädiktor oft zwischen schwachen Zuständen wechselt und somit ohne Zuverlässigkeitsverfahren oft Fehlvorhersagen liefert. Die geringste Steigerung wird für die Person C mit dem Ordnung 5 erreicht. Der Grund für die geringe Steigerung sind viele Wechsel zwischen den vorherzusagenden Kontexten. Deswegen wird oft in sicheren Zuständen eine falsche Vorhersage gemacht, was auch die Genauigkeiten um 40% zeigen.

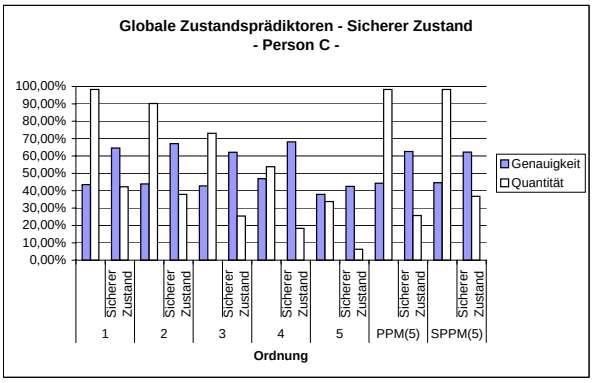
Wenn man die Quantität betrachtet, ist festzustellen, dass diese wie erwartet



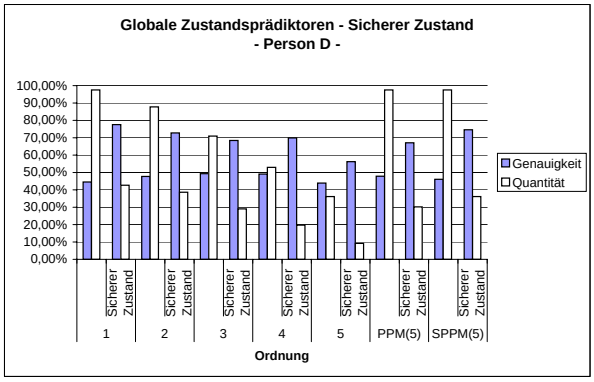
Person A	
Ordnung	Steigerung
1	48,80%
2	47,66%
3	39,41%
4	50,65%
5	29,57%
PPM(5)	56,35%
SPPM(5)	54,39%



Person B	
Ordnung	Steigerung
1	39,50%
2	51,58%
3	41,26%
4	50,39%
5	40,22%
PPM(5)	45,30%
SPPM(5)	38,53%



Person C	
Ordnung	Steigerung
1	37,26%
2	41,34%
3	33,80%
4	39,91%
5	7,48%
PPM(5)	32,81%
SPPM(5)	31,80%



Person D	
Ordnung	Steigerung
1	59,59%
2	47,93%
3	37,68%
4	40,87%
5	21,99%
PPM(5)	36,90%
SPPM(5)	52,90%

Abbildung 6.6: Globale Zustandsprädiktoren mit Verfahren *Sicherer Zustand* - Augsburg Benchmarks

in allen Fällen sinkt. Im Fall des globalen zweistufigen Two-State-Prädiktor mit Ordnung 5 sinkt die Quantität bis auf Person C sogar unter 10%.

6.2.3 Schwellenwert-Verfahren

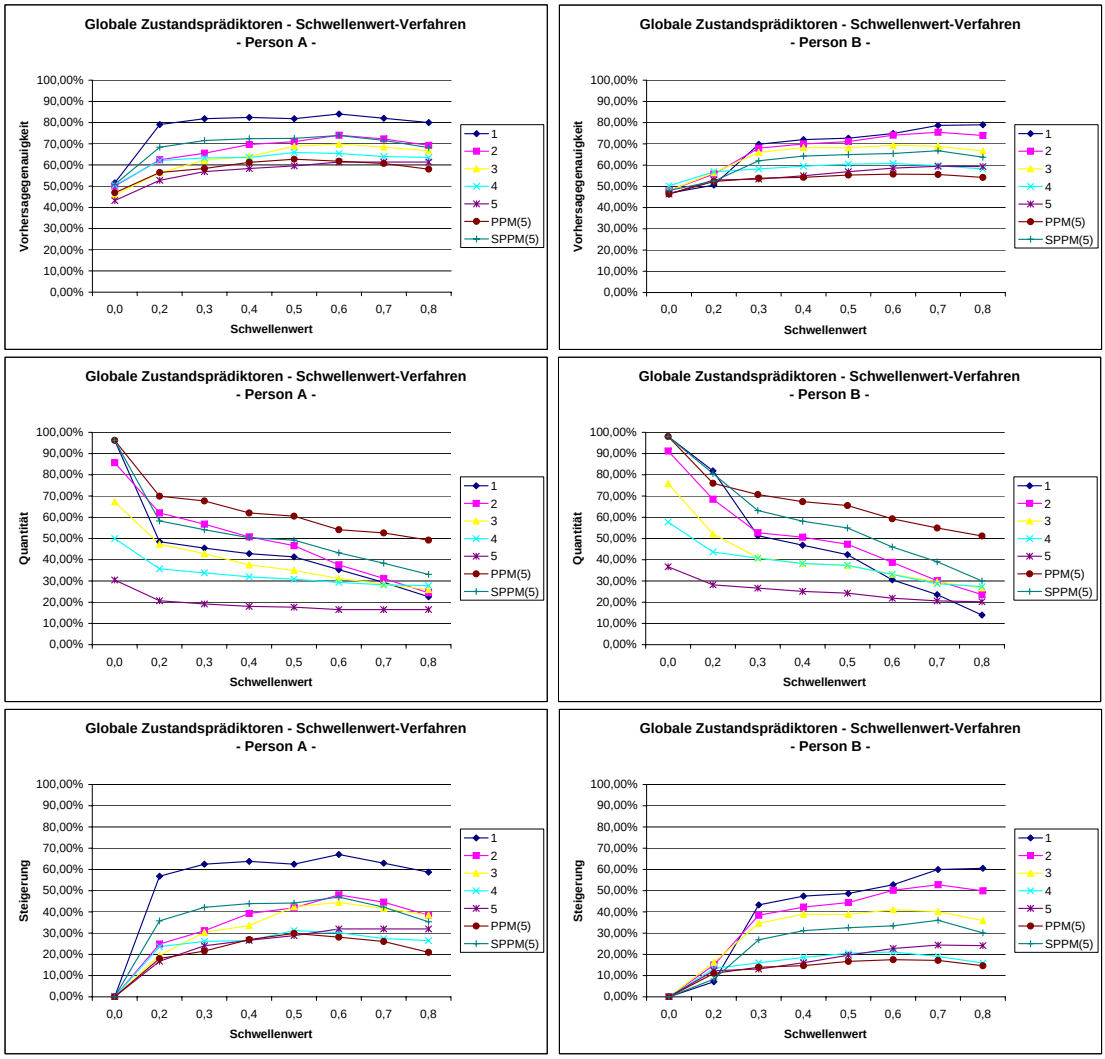


Abbildung 6.7: Globale Zustandsprädiktoren mit Schwellenwert-Verfahren - Person A und B

In den Abbildungen 6.7 und 6.8 sind die Ergebnisse der Messungen mit dem Schwellenwert-Verfahren dargestellt. Für alle vier Personen wurde die Vorhersagegenauigkeit, die Quantität sowie die Steigerung gemessen. Hierbei wurden nur die globalen Two-State-Prädiktoren bewertet. Der Schwellenwert wurde von $0,2$ bis $0,8$ variiert.

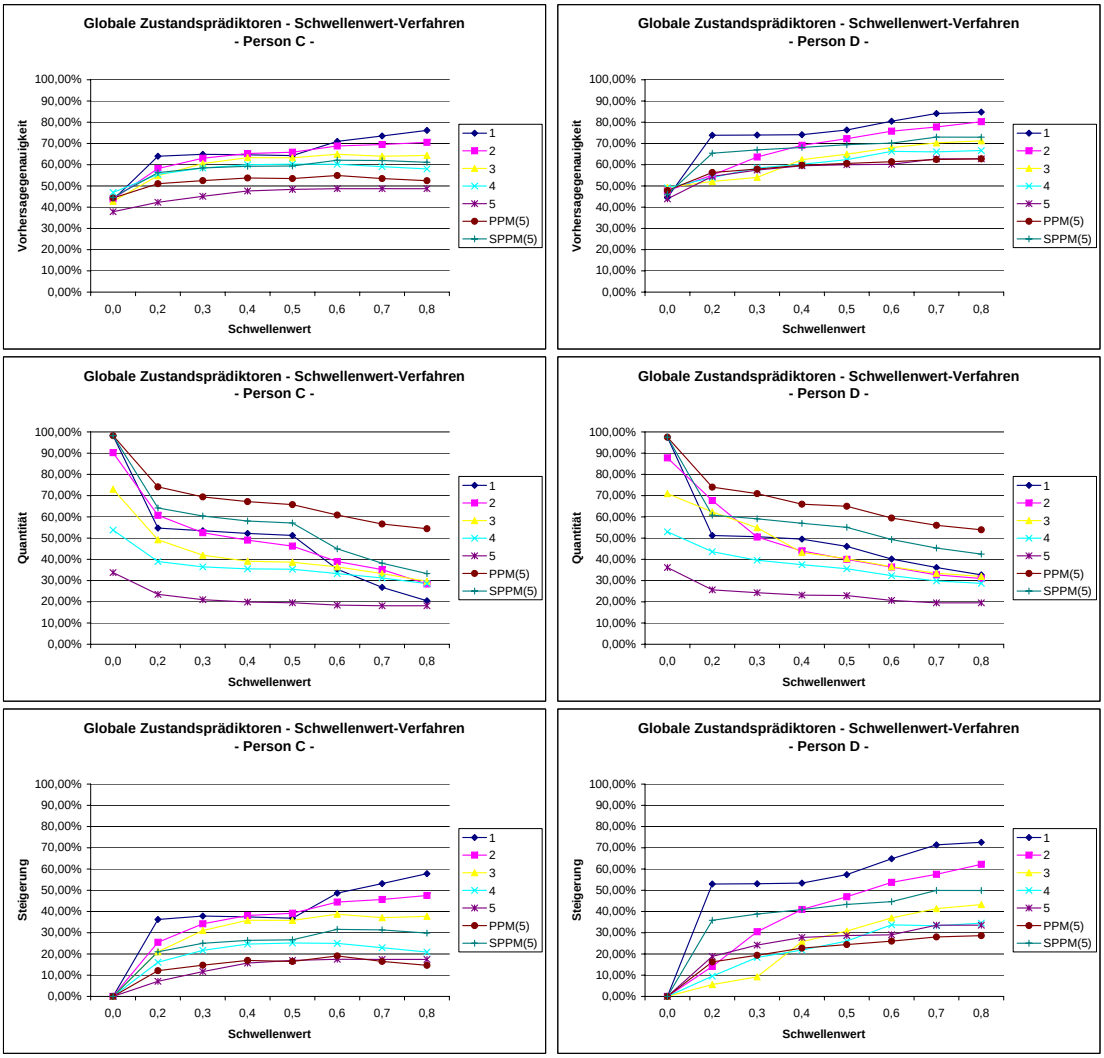


Abbildung 6.8: Globale Zustandsprädiktoren mit Schwellenwert-Verfahren - Person C und D

In der Regel steigt die Vorhersagegenauigkeit mit zunehmenden Schwellenwert. Bei Person A ist zu beobachten, dass die Genauigkeit ab einen Schwellenwert von $0,7$ abnimmt. Dies liegt daran, dass der globale zweistufige Two-State-Prädiktor für viele Muster eine Vorhersagegenauigkeit zwischen 60 und 70 Prozent hat. Die höchste Vorhersagegenauigkeit mit Schwellenwert-Verfahren erreicht bei allen vier Personen der globale Two-State-Prädiktor mit Ordnung 1.

Wie zu erwarten sinkt die Quantität bei zunehmenden Schwellenwert. Die höchste Quantität erreicht der globale zweistufige Two-State-Prädiktor mit Prediction by Partial Matching. Die Steigerung verhält sich wie die Vorhersagegenauigkeit – mit einem höheren Schwellenwert ist auch eine höhere Steigerung auszumachen. Bei Person A ist wieder ein Abfallen ab Schwellenwert $0,7$ zu beobachten. Die größte

Steigerung erreicht bei allen Personen der Prädiktor mit Ordnung 1.

6.2.4 Zuverlässigkeitszähler

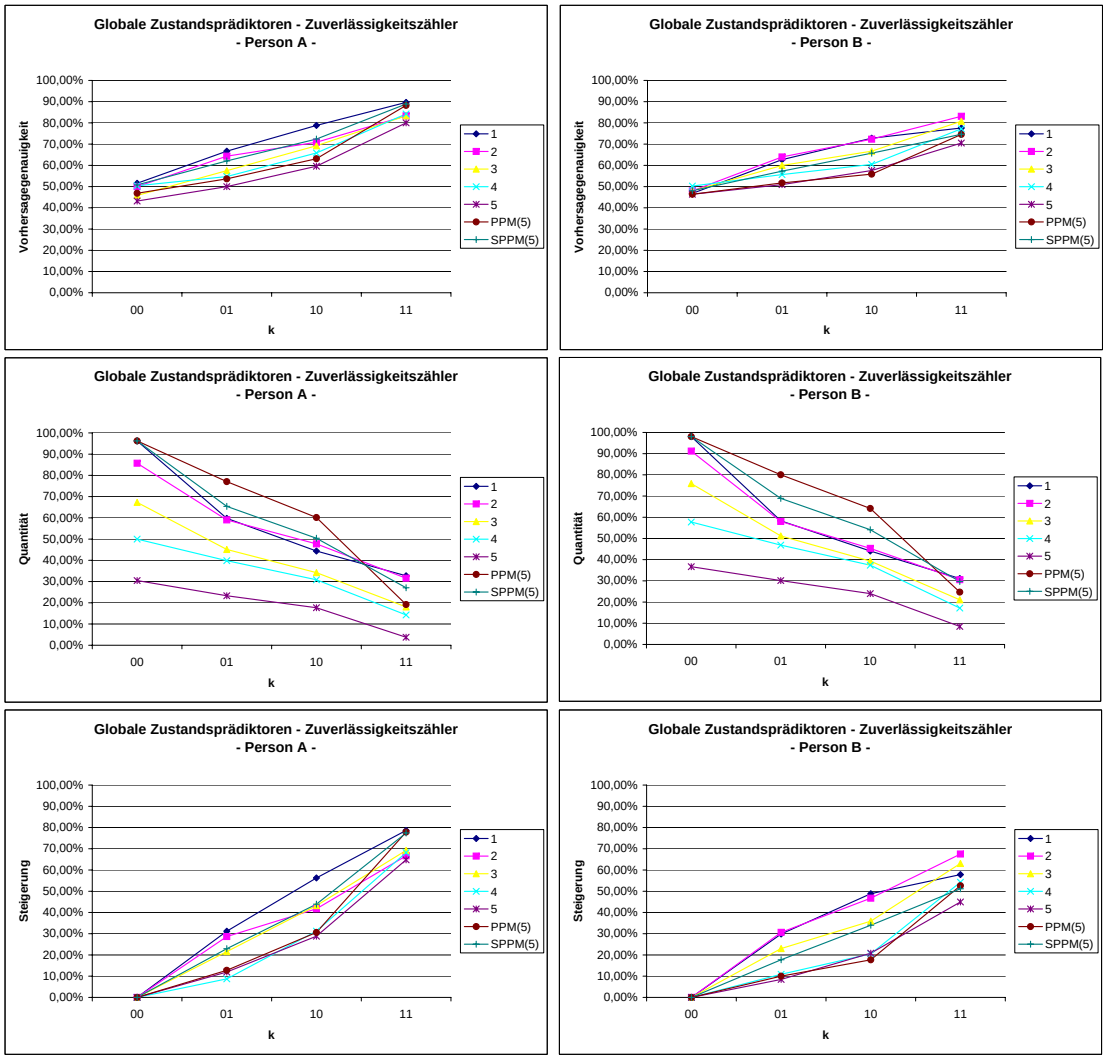
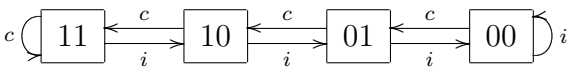


Abbildung 6.9: Globale Zustandsprädiktoren mit Zuverlässigkeitszähler - Person A und B

Die Ergebnisse der Messungen mit Zuverlässigkeitszähler sind in den Abbildungen 6.9 und 6.10 dargestellt. Für die Messungen wurde $n = 3$ gewählt, d.h. der Zähler hat vier Zustände, welche mit zwei Bit wie folgt dargestellt werden können:



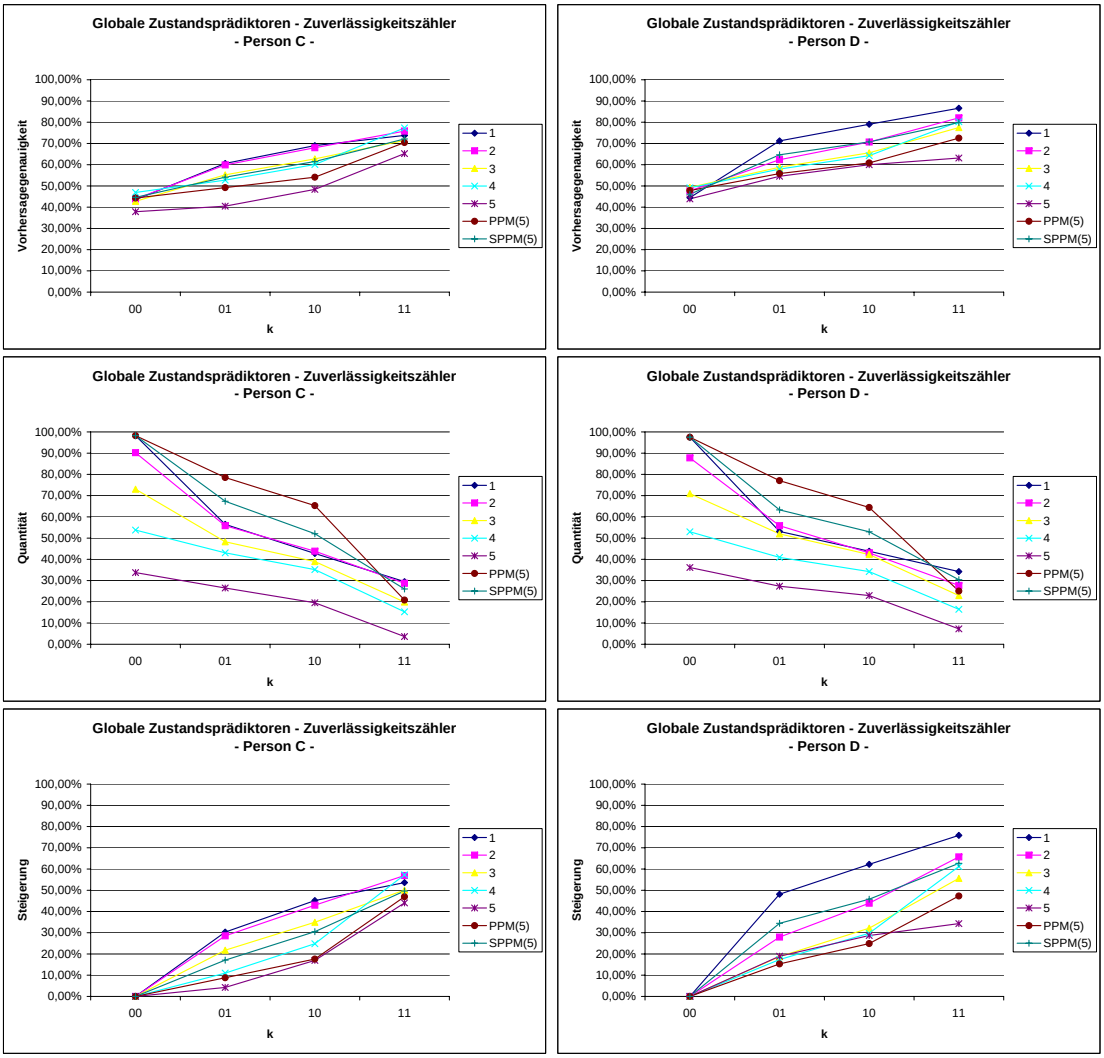


Abbildung 6.10: Globale Zustandsprädiktoren mit Zuverlässigkeitszähler - Person C und D

Als Initialzustand wurde der Zustand 10 gewählt. Der Wert k wurde variiert, d.h. es wurden Messungen mit $k \in \{00, 01, 10, 11\}$ durchgeführt, wobei sich für $k = 00$ der Prädiktor analog zu dem entsprechenden Prädiktor ohne Zuverlässigkeitszähler verhält. Wiederum werden nur die globalen zweistufigen Two-State-Prädiktoren für die vier Personen betrachtet. Desweiteren wurde wieder die Vorhersagegenauigkeit, die Quantität sowie die Steigerung gemessen.

Wie zu erwarten steigt die Vorhersagegenauigkeit bei allen Personen mit größerem k und die Quantität sinkt mit größerem k. Die Steigerung verhält sich analog zur Vorhersagegenauigkeit. Die höchsten Vorhersagegenauigkeiten sowie die größte Steigerung erreichen in fast allen Fällen die globalen Two-State-Prädiktoren mit Ordnung 1 und 2. Für $k = 01$ und $k = 10$ liefert der Prädiktor mit Prediction by

Partial Matching die beste Quantität. Die zweitbesten Werte für die Quantität werden vom Prädiktor mit Simple Prediction by Partial Matching erreicht. Für $k = 11$ brechen die Werte dieser beiden Prädiktoren ein. Die geringste Quantität hat in allen Fällen der globale Two-State-Prädiktor mit Ordnung 5.

6.3 Evaluierung mit Nokia Context Daten

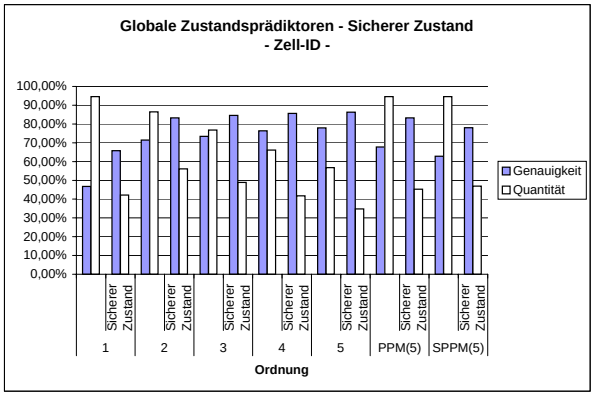
Für die Nokia Context Daten sollen wieder die drei Kontexte Zell-ID, Location Area Code und Benutzeraktivität betrachtet werden. Auch hier werden nur die globalen zweistufigen Two-State-Prädiktoren mit Ordnung 1 bis 5 sowie mit *Prediction by partial Matching* (PPM) und *Simple Prediction by Partial Matching* (SPPM) bezüglich der Vorhersagegenauigkeit, Quantität und Steigerung untersucht. Da für die statische Zuverlässigkeitsanalyse das notwendige Wissen über die Benchmarks fehlt, wird diese hier nicht betrachtet.

6.3.1 Sicherer Zustand

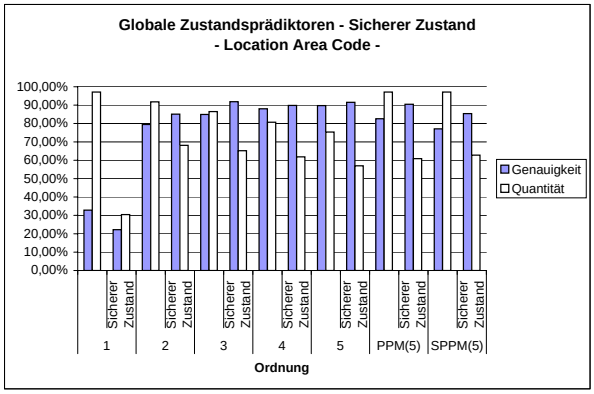
Abbildung 6.11 zeigt die Vorhersagegenauigkeit, die Quantität sowie die Steigerung der Vorhersagegenauigkeit mit dem Zuverlässigkeitsverfahren *Sicherer Zustand* für die drei Kontexte Zell-ID, Location Area Code und Aktivität der Nokia Context Daten. Bis auf den Prädiktor mit Ordnung 1 beim Location Area Code steigt die Vorhersagegenauigkeit bei Verwendung des Zuverlässigkeitsverfahrens *Sicherer Zustand*. Desweiteren fällt wie erwartet die Quantität in allen betrachteten Fällen. Für die Zell-ID wird mit diesem Zuverlässigkeitsverfahren eine Vorhersagegenauigkeit von über 86% bei Ordnung 5 erreicht. Die Vorhersagegenauigkeit beim Location Area Code wird sogar auf fast 92% für Ordnung 3 gesteigert. Ähnlich verhält es sich bei der Aktivität, wo auch eine Genauigkeit von fast 92% für Ordnung 5 erreicht wird.

6.3.2 Schwellenwert-Verfahren

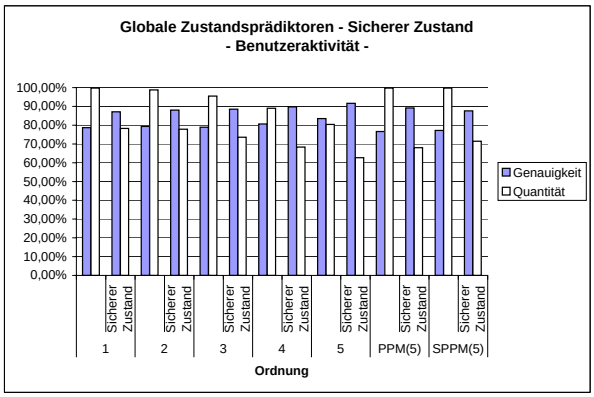
Die Abbildungen 6.12, 6.13 und 6.14 zeigen jeweils in der linken Spalte die Vorhersagegenauigkeit, die Quantität und die Steigerung bei Verwendung des Schwellenwert-Verfahrens für die Nokia Context Daten. Es wurden die Schwellenwerte $0,2$ bis $0,8$ wie in der Evaluierung mit den Augsburg Benchmarks untersucht. Der Schwellenwert $0,0$ entspricht der Vorhersagegenauigkeit ohne Verwendung des Zuverlässigkeitsverfahrens. Bei Betrachtung der Zell-ID ist festzustellen, dass mit einem größeren Schwellenwert auch eine höhere Vorhersagegenauigkeit erreicht wird. Die Quantität fällt wie erwartet mit größerem Schwellenwert. Mit Ordnung 2 und Schwellenwert $0,8$ wird hier eine Vorhersagegenauigkeit von fast



Zell-ID	
Ordnung	Steigerung
1	35,79%
2	41,29%
3	41,82%
4	39,23%
5	38,04%
PPM(5)	47,96%
SPPM(5)	40,85%



Location Area Code	
Ordnung	Steigerung
1	-15,80%
2	27,44%
3	45,98%
4	15,20%
5	17,37%
PPM(5)	45,31%
SPPM(5)	36,14%



Benutzeraktivität	
Ordnung	Steigerung
1	39,58%
2	42,05%
3	45,45%
4	46,83%
5	49,27%
PPM(5)	53,74%
SPPM(5)	45,66%

Abbildung 6.11: Globale Zustandsprädiktoren mit Verfahren *Sicherer Zustand* - Nokia Context Daten

86% erreicht. Beim Location Area Code steigt die Vorhersagegenauigkeit mit größerem Schwellenwert nur für die Ordnungen 2 und 3 sowie für die PPM- und SPPM-Variante (bis auf einzelne Ausreißer). Mit Ordnung 1 steigt die Genauigkeit zunächst bis Schwellenwert $0,5$, bei Schwellenwert $0,6$ fällt diese sogar unter den Wert ohne Zuverlässigkeitsverfahren. Für Schwellenwert $0,7$ und $0,8$ steigt die Genauigkeit wieder leicht an. Mit Ordnung 4 und 5 ist die Vorhersagegenauigkeit sogar bis Schwellenwert $0,6$ schlechter als ohne Zuverlässigkeit. Erst ab Schwellenwert $0,7$ ist eine positive Steigerung auszumachen. Da mit Ordnung

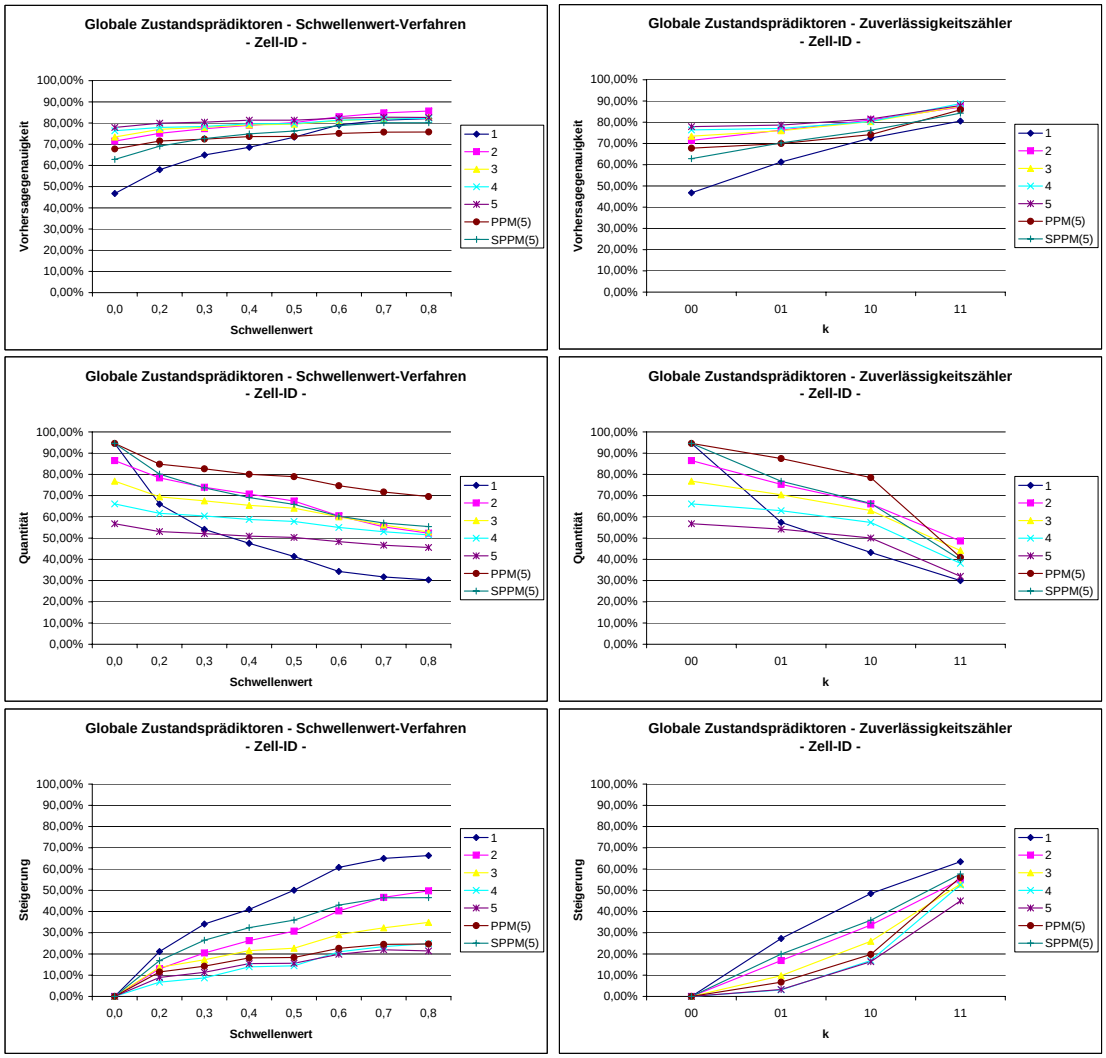


Abbildung 6.12: Globale Zustandsprädiktoren mit Schwellenwert-Verfahren bzw. Zuverlässigkeitszähler - Zell-ID

5 ohne Zuverlässigkeitsverfahren die höchste Vorhersagegenauigkeit unter allen Ordnungen erreicht wurde, wird auch bei Verwendung eines Schwellenwert von $0,8$ die höchste Genauigkeit von über 92% mit Ordnung 5 erreicht. Die Quantität verhält sich wie erwartet und sinkt mit größerem Schwellenwert. Bei der Benutzeraktivität steigt die Vorhersagegenauigkeit wieder mit größerem Schwellenwert, die Quantität sinkt mit größerem Schwellenwert. Für die Benutzeraktivität wird hierbei eine Vorhersagegenauigkeit von fast 88% mit Ordnung 2 und 3 erreicht.

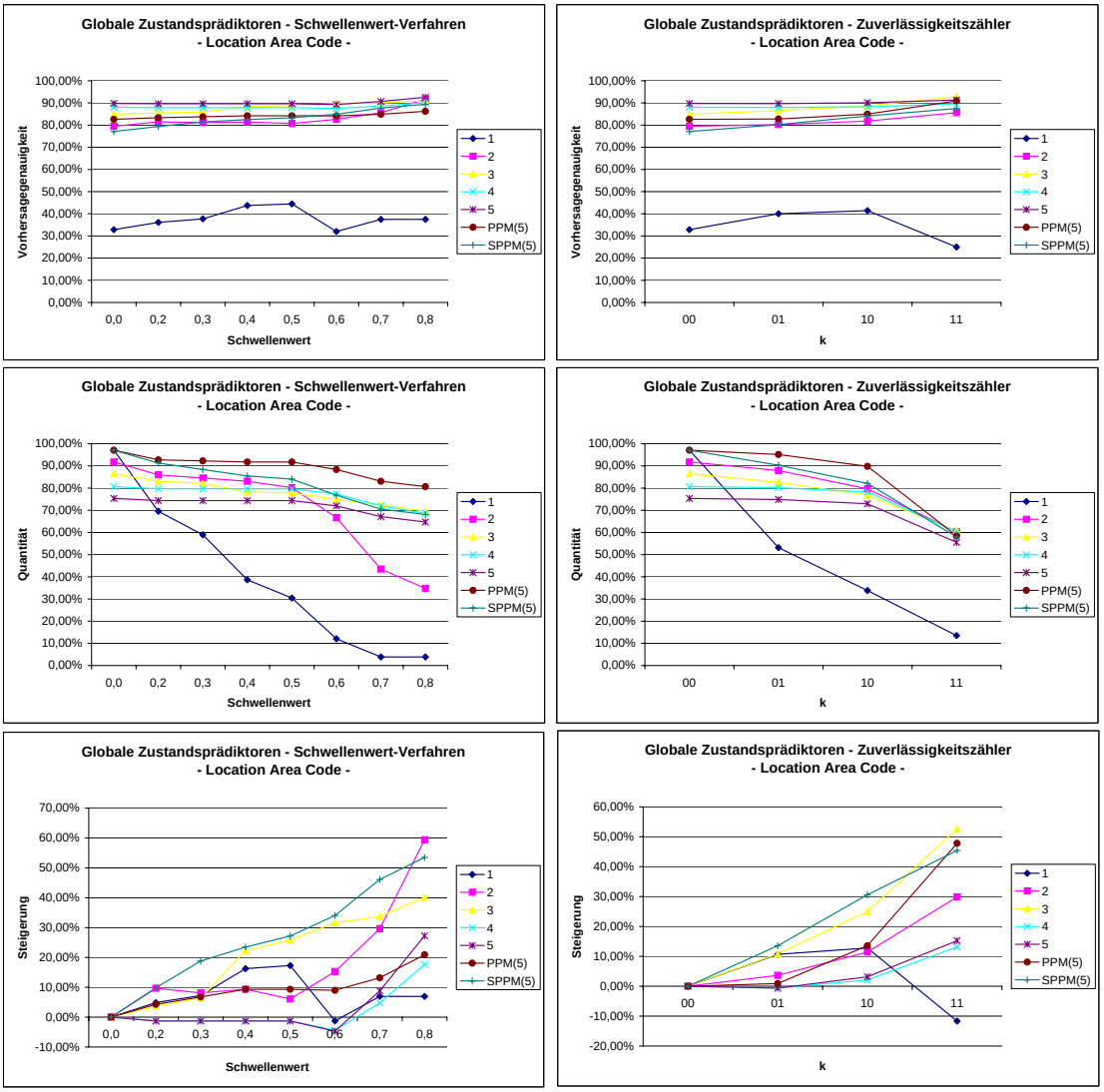


Abbildung 6.13: Globale Zustandsprädiktoren mit Schwellenwert-Verfahren bzw. Zuverlässigkeitszähler - Location Area Code

6.3.3 Zuverlässigkeitszähler

Die Abbildungen 6.12, 6.13 und 6.14 zeigen jeweils in der rechten Spalte die Vorhersagegenauigkeit, die Quantität und die Steigerung bei Verwendung eines Zuverlässigkeitszählers für die Nokia Context Daten. Es wurde der Zuverlässigkeitszähler mit vier Zuständen wie bei der Evaluierung mit den Augsburg Benchmarks verwendet. Alle vier Zustände des Zähler wurden als Barriere untersucht ($k \in \{00, 01, 10, 11\}$), wobei $k = 00$ der Vorhersage ohne Zuverlässigkeitszähler entspricht. Bei der Zell-ID sowie der Benutzeraktivität wächst die Vorhersagegenauigkeit mit größerem k, die Quantität sinkt dementsprechend. Für die Zell-ID

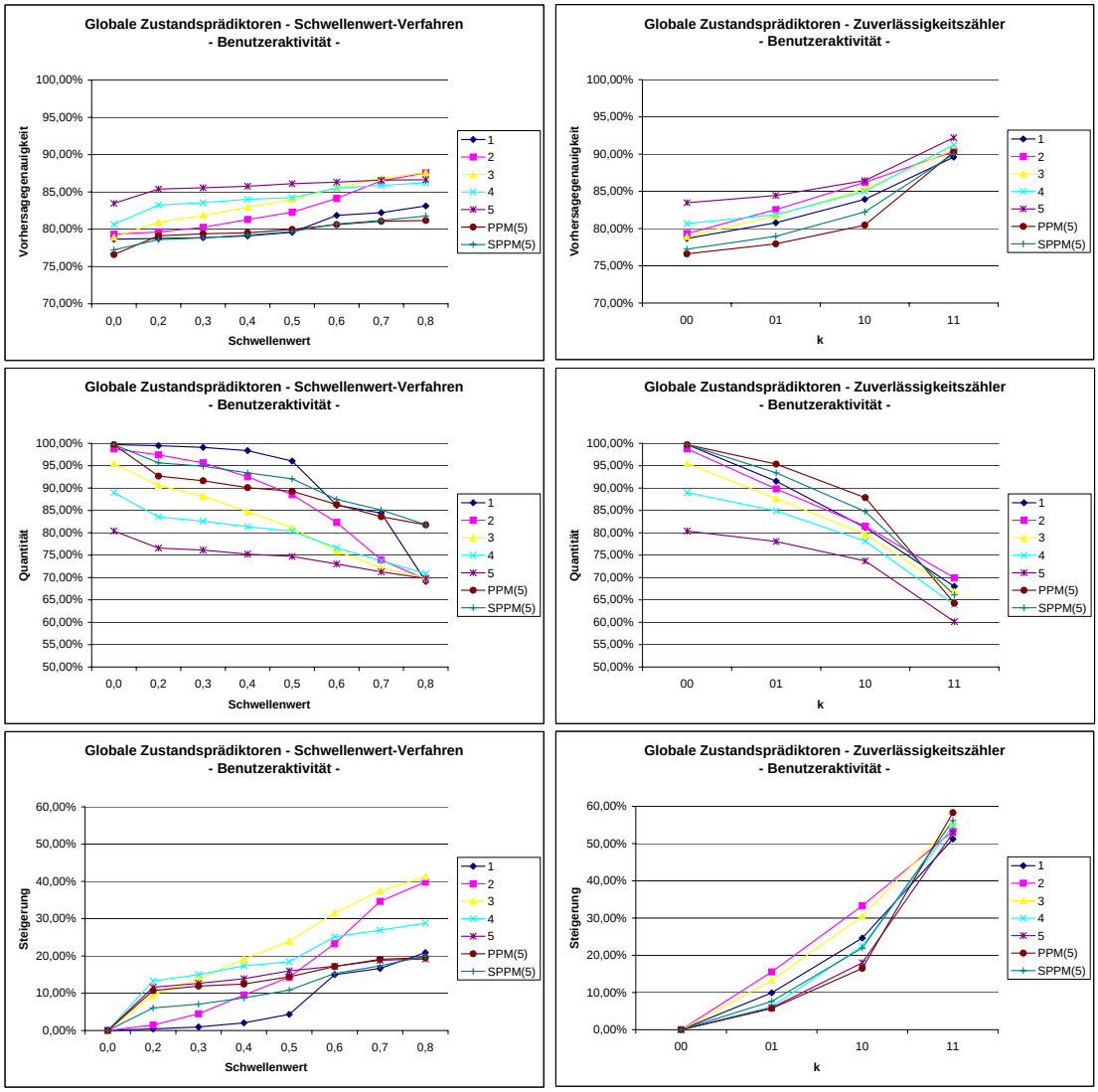


Abbildung 6.14: Globale Zustandsprädiktoren mit Schwellenwert-Verfahren bzw. Zuverlässigkeitszähler - Benutzeraktivität

wird dabei mit Ordnung 4 und $k = 11$ eine Genauigkeit von fast 88% erreicht. Für die Benutzeraktivität werden über 92% Vorhersagegenauigkeit mit Ordnung 5 erreicht. Beim Location Area Code fällt die Vorhersagegenauigkeit mit Ordnung 1 für $k = 11$ unter den erreichten Wert ohne Zuverlässigkeitszähler. Für die Ordnungen 4 und 5 fällt die Genauigkeit zunächst mit $k = 01$, steigt aber dann mit größerem k an. Für alle anderen Ordnungen verhält sich die Vorhersagegenauigkeit wie für die anderen Kontexte und steigt mit größerem k. Die höchste Vorhersagegenauigkeit von fast 93% wird mit Ordnung 3 und $k = 11$ erzielt. Die Quantität fällt wie erwartet mit größerem k auch für den Location Area Code.

6.4 Fazit Zuverlässigkeitsschätzung

Bei allen untersuchten Kontexten stellt sich die Verwendung des Zuverlässigkeitszählers mit Barriere $k = 11$ als das Verfahren mit den höchsten Vorhersagegenauigkeiten heraus. Auch ist bei diesem Verfahren die Quantität noch akzeptabel, bis auf Person C der Augsburg Benchmarks liegt diese über 30%. Eine höhere Quantität wird mit einer kleineren Barriere (z.B. $k = 10$) erreicht, wobei die Vorhersagegenauigkeit dabei bis zu 10%-Punkte kleiner ist.

Die mit dem Schwellenwert-Verfahren erreichten Vorhersagegenauigkeiten liegen knapp unter den besten Werten mit Zuverlässigkeitszähler und wurden fast alle mit dem Schwellenwert von $0,8$ erreicht. Bei der Quantität unterscheiden sich die beiden Benchmarks. Mit den Augsburg Benchmarks liegt die Quantität mit Schwellenwert-Verfahren fast immer unter der Quantität mit Zuverlässigkeitszähler. Mit den Nokia Context Daten verhält sich dies umgekehrt. Ein Vorteil des Schwellenwert-Verfahrens ist, dass mit kleineren Schwellenwerten zwar auch die Vorhersagegenauigkeiten sinken, aber dies deutlich geringer ausfällt als mit Zuverlässigkeitszähler.

Mit den Augsburg Benchmarks lieferte das Verfahren *Sicherer Zustand* die geringsten Steigerungen unter allen Verfahren. Sogar die statische Zuverlässigkeit erzielt hier höhere Vorhersagegenauigkeiten. Dies zeigt, wenn Wissen über die Kontexte vorhanden ist, kann dies wie im Fall der Augsburg Benchmarks einen großen Vorteil bringen. Bei den Nokia Context Daten liefert das Verfahren *Sicherer Zustand* höhere Vorhersagegenauigkeiten als das Schwellenwert-Verfahren (Zell-ID, Benutzeraktivität) bzw. die Vorhersagegenauigkeit liegt nur knapp unter der mit dem Schwellenwert-Verfahren erzielten (Location Area Code).

Dies zeigt, dass es unter den Verfahren kein bestes gibt. Das statische Verfahren sowie das Verfahren *Sicherer Zustand* verbrauchen bezüglich Speicher keine weiteren Kosten. Beim Schwellenwert-Verfahren und Zuverlässigkeitszähler muss vorab eine geeignete Barriere bestimmt werden. Für den Zuverlässigkeitszähler mit vier Zuständen könnte diese $k = 10$ als Kompromiss zwischen hoher Vorhersagegenauigkeit und noch akzeptabler Quantität sein.

7 Hybridprädiktoren

In diesem Kapitel soll untersucht werden, ob durch die gleichzeitige Verwendung mehrerer Prädiktoren Vorteile entstehen. Diese Vorteile können eine bessere Vorhersagegenauigkeit oder aber bei gleicher Genauigkeit eine höhere Quantität sein. Die Hybridprädiktoren bestehen aus einer Menge von Prädiktoren, die alle zur Vorhersage verwendet werden. Aus der Menge der gelieferten Vorhersageergebnissen wird dann über ein bestimmtes Auswahlverfahren das Vorhersageergebnis des Hybridprädiktors bestimmt. Der Speicher- und Rechenaufwand der Hybridprädiktoren setzt sich aus dem Aufwand der verwendeten Prädiktoren zusammen. Zusätzlich wird noch der Rechenaufwand für das Auswahlverfahren benötigt.

Für das Auswahlverfahren gibt es verschiedene Möglichkeiten. Im Folgenden werden drei Hybridprädiktoren – ein Warm-up-Prädiktor, ein Mehrheitsprädiktor und ein Zuverlässigkeitsprädiktor – vorgestellt, die sich durch die verwendeten Auswahlverfahren unterscheiden.

7.1 Verfahren

7.1.1 Warm-up-Prädiktor

Die Idee des Warm-up-Prädiktors wurde schon in Kapitel 5 angesprochen. Zugrunde lag der Vorteil der Prädiktoren mit einer kleinen Ordnung schnell anzulernen und der Nachteil keine komplexen Muster erlernen zu können. Bei Prädiktoren mit großer Ordnung verhält es sich umgekehrt, sie benötigen eine längere Anlernphase, können aber komplexere Muster erlernen. Die Kombination beider Vorteile führt zum *Prediction by Partial Matching* (PPM) bzw. *Simple Prediction by Partial Matching* (SPPM).

Das SPPM-Verfahren spiegelt die Idee eines Prädiktor mit niedriger Ordnung ($r = 1$) kombiniert mit einem Prädiktor höherer Ordnung ($r =$ maximale Ordnung) wider. Wenn der Prädiktor mit maximaler Ordnung keine Vorhersage machen kann, da das entsprechende Muster noch nie aufgetreten ist, wird der Prädiktor mit Ordnung $r = 1$ zur Vorhersage verwendet. Es wird somit das schnelle Anlernverhalten des Prädiktors mit Ordnung $r = 1$ ausgenutzt. Kennt der Prädiktor mit maximaler Ordnung das aktuelle Muster, d.h. dieses Muster ist vorher schon

mindestens einmal aufgetreten, wird die Vorhersage dieses Prädiktors verwendet. Es wird somit der Vorteil ausgenutzt, dass der Prädiktor mit maximaler Ordnung komplexere Muster erlernen kann. Abbildung 7.1 zeigt dieses Verfahren für den SPPM-Prädiktor mit Two-State-Prädiktor und maximaler Ordnung 5. Die farbigen Flächen geben dabei an, wenn eine Vorhersage durch den jeweiligen Prädiktor gemacht werden kann. Durch Verwendung des Prädiktors mit Ordnung 1 werden die Leerräume (keine Vorhersage möglich) des Prädiktors mit Ordnung 5 reduziert.

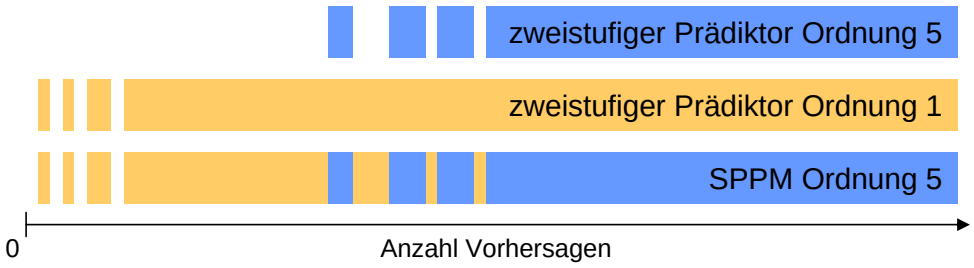


Abbildung 7.1: Warm-up-Prädiktor

Auch das PPM-Verfahren gehört in die Klasse der Warm-up-Prädiktoren. Betrachtet man einen PPM-Prädiktor mit maximaler Ordnung 5, besteht die Menge der für den Hybridprädiktor verwendeten Prädiktoren aus den Prädiktoren mit Ordnung 1, 2, 3, 4 und 5. Ein Nachteil des SPPM-Verfahren ist, dass nur die Extrema verwendet werden, d.h. der Prädiktor mit Ordnung 1, welcher am schnellsten angelernt ist, sowie der Prädiktor mit maximaler Ordnung, welcher komplexe Muster erlernen kann. Tritt ein Muster maximaler Länge das erste Mal auf, wird sofort der Prädiktor mit Ordnung 1 verwendet. Das PPM-Verfahren füllt diese Lücke, da hier, nachdem der Prädiktor mit maximaler Ordnung m keine Vorhersage machen kann, der Prädiktor mit Ordnung $m-1$ für die Vorhersage verwendet wird. Kann dieser eine Vorhersage anbieten, beruht diese auf einem komplexeren Muster als die Vorhersage des Prädiktors mit Ordnung 1. Das PPM-Verfahren bietet somit eine Abstufung zwischen dem einfachsten und komplexesten Prädiktor.

Die hier verwendete Idee wurde auch in der Prozessorarchitektur von Young und Smith [YS94] vorgeschlagen. Während der Warmlaufphase eines zweifach adaptiven Prädiktors soll ein statischer oder ein einfacher dynamischer Prädiktor eingesetzt werden. Hier fehlt aber die vorab beschriebene Dynamik, dass in der Übergangsphase (siehe Abbildung 7.1) zwischen den beiden Prädiktoren hin- und hergeschaltet wird. In der Prozessorarchitektur wird für die ersten n Millionen Sprungvorhersagen ein einfacher dynamischer bzw. ein statischer Prädiktor verwendet. Dann wird umgeschaltet auf einen zweifach adaptiven Prädiktor und nur dieser wird von da an für die Sprungvorhersagen verwendet.

7.1.2 Mehrheitsprädiktor

Für die Auswahl des Vorhersageergebnisses dieses Hybridprädiktors werden nur die Prädiktoren aus der Prädiktormenge betrachtet, die auch ein Vorhersageergebnis liefern können. Die Vorhersage des Hybridprädiktors ist dann das Vorhersageergebnis, das unter den gelieferten Ergebnissen die Mehrheit hat. Es werden zwei verschiedene Mehrheiten betrachtet, die ihren Ursprung im demokratischen Mehrheitsbegriff haben. Zum Einen wird die *relative* Mehrheit verwendet. Die relative Mehrheit gibt das Ergebnis an, welches in der Menge der Ergebnisse am meisten vorhanden ist. Für den Fall, dass zwei Ergebnisse gleich oft in der Menge auftreten, wird das erste Ergebnis entsprechend der Sortierung der Prädiktoren in der Prädiktormenge gewählt. Zum Anderen wird die *einfache* Mehrheit verwendet. Die einfache Mehrheit ist dann erreicht, wenn mehr als die Hälfte der Vorhersageergebnisse gleich sind. Bei der einfachen Mehrheit kann es keine zwei Möglichkeiten geben. Wenn kein Ergebnis die einfache Mehrheit erreicht, gibt der Hybridprädiktor keine Vorhersage an.

Formal kann dieser Prädiktor wie folgt beschrieben werden. Sei P die Menge der Prädiktoren, die der Hybridprädiktor verwendet. Sei P_r mit $|P_r| \leq |P|$ die Menge der verwendeten Prädiktoren, die ein Vorhersageergebnis liefern und R sei die Menge der Vorhersageergebnisse der Prädiktoren aus P_r, wobei R eine Multimenge ist. Sei weiterhin die Anzahl eines Wertes x in einer Menge M definiert als $|M|_x$. Dann liefert der Mehrheitsprädiktor mit *relativer* Mehrheit a als Vorhersageergebnis, falls

$$|R|_a \geq |R|_b \quad \forall\, b \in R$$

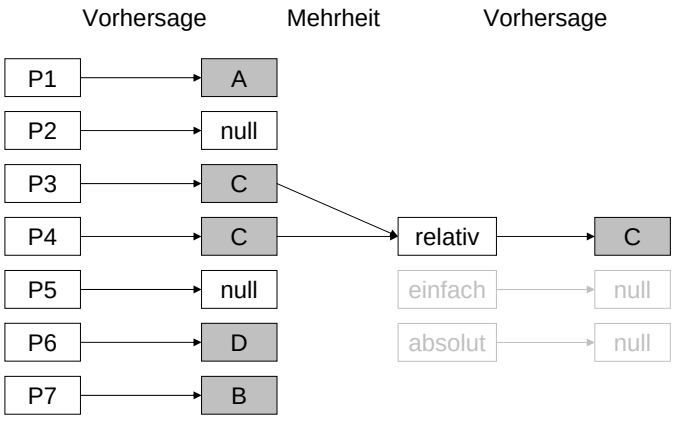


Abbildung 7.2: Mehrheitsprädiktor mit relativer Mehrheit

Abbildung 7.2 zeigt einen Mehrheitsprädiktor mit relativer Mehrheit, dessen Prädiktormenge P sieben Prädiktoren enthält. In diesem Beispiel liefern 5 Prädiktoren ein Ergebnis, d.h. $P_r = \{P_1, P_3, P_4, P_6, P_7\}$. Die Vorhersageergebnisse

dieser 5 Prädiktoren bilden die Multimenge $R = \{A, C, C, D, B\}$. Damit gilt $|R|_C = 2 > 1 = |R|_x \quad \forall\, x \in R \setminus \{C\}$. Somit ist die Vorhersage C.

Der Mehrheitsprädiktor mit *einfacher* Mehrheit liefert a als Vorhersageergebnis, falls

$$|R|_a > \frac{|R|}{2}$$

Wenn es kein solches a gibt, liefert der Hybridprädiktor kein Ergebnis, d.h. der Mehrheitsprädiktor mit einfacher Mehrheit wird in weniger Fällen als der Mehrheitsprädiktor mit relativer Mehrheit eine Vorhersage machen. Das Beispiel aus Abbildung 7.2 zeigt dieses Verhalten, der Mehrheitsprädiktor mit einfacher Mehrheit kann kein Ergebnis liefern. Abbildung 7.3 zeigt ein weiteres Beispiel, in dem ein Mehrheitsprädiktor mit einfacher Mehrheit ein Ergebnis liefert.

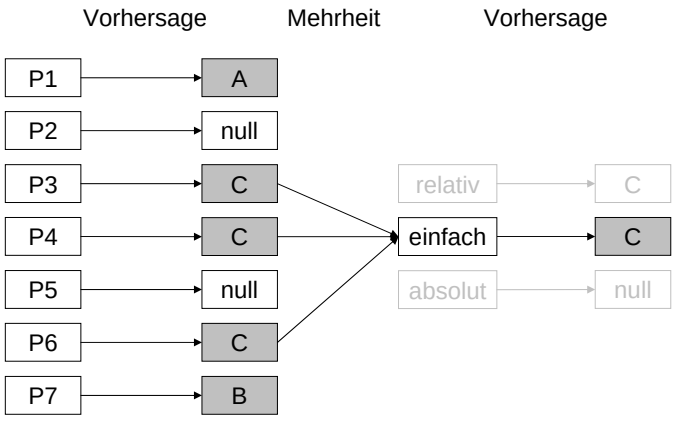


Abbildung 7.3: Mehrheitsprädiktor mit einfacher Mehrheit

In diesem Beispiel können wiederum die 5 Prädiktoren eine Vorhersage anbieten, d.h. $P_r = \{P_1, P_3, P_4, P_6, P_7\}$. Die Vorhersageergebnisse dieser 5 Prädiktoren bilden die Multimenge $R = \{A, C, C, C, B\}$. Weiter gilt $|R|_C = 3 > 2,5 = \frac{|R|}{2}$, so dass C die Vorhersage des Hybridprädiktors ist. Natürlich würde auch ein Mehrheitsprädiktor mit relativer Mehrheit dieses Ergebnis liefern.

Man kann die Überlegungen auch fortführen und einen Prädiktor mit *absoluter* Mehrheit betrachten. Dabei wird das Ergebnis als Vorhersage des Hybridprädiktors verwendet, welches die Mehrheit bezogen auf alle Prädiktoren in der Prädiktormenge erreicht. Ein Mehrheitsprädiktor mit *absoluter* Mehrheit liefert a als Vorhersage, falls

$$|R|_a > \frac{|P|}{2}$$

Da hier in noch weniger Fällen eine Vorhersage gemacht werden würde, wird diese Variante in der Evaluierung nicht weiter betrachtet. Die vorherigen Beispiele (Abbildungen 7.2 und 7.3) zeigen dieses Verhalten, in beiden Fällen kann ein Mehrheitsprädiktor mit absoluter Mehrheit kein Ergebnis liefern. Abbildung 7.4 zeigt ein Beispiel, in dem auch der Mehrheitsprädiktor mit absoluter Mehrheit eine Vorhersage macht, die auch von den Prädiktoren mit relativer und einfacher Mehrheit angeboten wird.

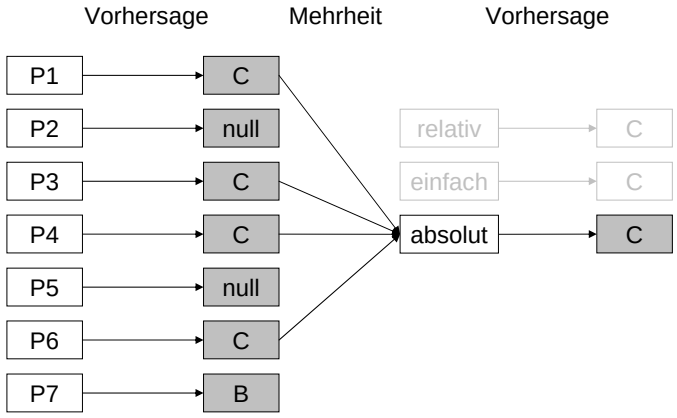


Abbildung 7.4: Mehrheitsprädiktor mit absoluter Mehrheit

Wie schon erwähnt besteht zwischen den Prädiktoren mit den verschiedenen Mehrheiten ein Zusammenhang. Sei V_{rel} die Menge aller gelieferten Vorhersagen einer Versuchsreihe eines Mehrheitsprädiktors mit relativer Mehrheit, V_{ein} die eines Mehrheitsprädiktors mit einfacher Mehrheit sowie V_{abs} die eines Mehrheitsprädiktors mit absoluter Mehrheit. Dann gilt

$$V_{abs} \subseteq V_{ein} \subseteq V_{rel}$$

Die Ergebnisse, die von einem Mehrheitsprädiktor mit absoluter Mehrheit geliefert werden, werden auch von den beiden anderen Mehrheitsprädiktoren mit einfacher und relativer Mehrheit geliefert. Die Ergebnisse, die von einem Mehrheitsprädiktor mit einfacher Mehrheit geliefert werden, werden auch von einem Mehrheitsprädiktor mit relativer Mehrheit geliefert.

7.1.3 Zuverlässigkeitsprädiktor

Im Bereich der Prozessorarchitektur wurde von Grunwald et al. [GKMP98] ein Zuverlässigkeitsprädiktor als Hybridprädiktor für die Sprungvorhersage vorgeschlagen. Hierbei wird der Prädiktor mit der größten Zuverlässigkeit für die Vorhersage ausgewählt. Dabei hat der Prädiktor die größte Zuverlässigkeit, der

die größte Trefferwahrscheinlichkeiten erreicht. Da in Kapitel 6 verschiedene Zuverlässigkeitsschätzer betrachtet wurde, bietet sich auch hier eine Untersuchung verschiedener Zuverlässigkeitsprädiktoren an. Abbildung 7.5 zeigt die Arbeitsweise eines Zuverlässigkeitsprädiktors.

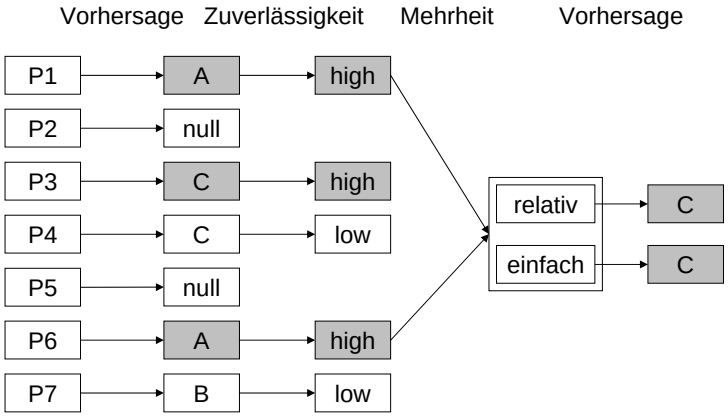


Abbildung 7.5: Zuverlässigkeitsprädiktor

Zunächst macht jeder Prädiktor aus der Prädiktormenge eine Vorhersage. Dann werden die Ergebnisse der Prädiktoren, die ein Vorhersageergebnis geliefert haben, mittels eines Zuverlässigkeitsverfahrens in eine Zuverlässigkeitsstufe eingeordnet. Die Prädiktoren mit der höchsten Zuverlässigkeitsstufe werden zur Vorhersage herangezogen. Aus der Menge der Vorhersageergebnisse dieser Prädiktoren wird dann wieder mittels eines Mehrheitsverfahrens das Ergebnis des Hybridprädiktors gewählt.

Bei der Verwendung eines Zuverlässigkeitsprädiktors gilt es zwischen den Vorhersagen der einzelnen verwendeten Prädiktoren und der Vorhersage des Hybridprädiktors mehrere Entscheidungen zu treffen. Primär muss entschieden werden, welches Zuverlässigkeitsverfahren verwendet wird. Sekundär muss eine Mehrheitsverfahren für den Fall von mehreren möglichen Vorhersageergebnissen gewählt werden. Im Folgenden soll zunächst das primäre Auswahlkriterium näher betrachtet werden. Anschließend wird das sekundäre sowie eine mögliche Erweiterung der Zuverlässigkeitsprädiktoren, das tertiäre Auswahlkriterium, beschrieben.

Primäres Auswahlkriterium

Die Zuverlässigkeit der Prädiktoren kann mittels eines der vorgestellten Verfahren aus Kapitel 6 bestimmt werden. Die statische Zuverlässigkeit wird nicht betrachtet, da sich dieses Verfahren nicht direkt auf die einzelnen Prädiktoren bezieht, sondern auf den aktuellen Kontext, in dem sich der Benutzer befindet. Hier sollen das Verfahren *Sicherer Zustand*, das Schwellenwert-Verfahren und

der Zuverlässigkeitszähler betrachtet werden. Dabei muss die Zuverlässigkeit aller Prädiktoren mittels des selben Verfahren bestimmt werden, da ein Vergleich der Zuverlässigkeitsstufen verschiedener Zuverlässigkeitsverfahren untereinander nicht möglich ist.

Ein Zuverlässigkeitsprädiktor, der das Verfahren *Sicherer Zustand* verwendet, kann nur mit einer Menge von Zustandsprädiktoren arbeiten. Bei diesem Verfahren gibt es nur zwei Zuverlässigkeitsstufen, ein Prädiktor befindet sich im schwachen Zustand, was einer niedrigen Zuverlässigkeit entspricht, oder der Prädiktor befindet sich in einem sicheren Zustand, was einer hoher Zuverlässigkeit entspricht.

Bei der Verwendung des Schwellenwert-Verfahrens können alle möglichen Prädiktoren in der Prädiktormenge verwendet werden. Hier gibt es theoretisch eine unendliche Anzahl von Zuverlässigkeitsstufen, da die Stufen den Vorhersagegenauigkeiten der Prädiktoren entsprechen.

Auch ein Zuverlässigkeitsprädiktor, der mit einem Zuverlässigkeitszähler arbeitet, kann alle Prädiktoren in der Prädiktormenge verwenden. Mit diesem Verfahren existiert eine diskrete Anzahl von Zuverlässigkeitsstufen. Jeder Zustand des Zuverlässigkeitszählers bildet eine Zuverlässigkeitsstufe.

Sekundäres Auswahlkriterium

Nach der Einordnung der Vorhersageergebnisse in die Zuverlässigkeitsstufen, sollte nun der Prädiktor gewählt werden, der die höchste Zuverlässigkeitsstufe unter allen Prädiktoren erreicht hat. In den meisten Fällen wird dies aber nicht nur ein Prädiktor sein, sondern mehrere Prädiktoren werden die höchste Zuverlässigkeitsstufe erreichen. Deshalb muss an dieser Stelle mittels eines Mehrheitsverfahrens entschieden werden, welches Ergebnis aus der Menge der zuverlässigsten Prädiktoren für die Vorhersage verwendet wird. Im vorherigen Abschnitt wurden drei Verfahren für den Mehrheitsprädiktor vorgeschlagen. Wobei im Folgenden die relative und die einfache Mehrheit verwendet werden. Die absolute Mehrheit wird wie auch bei den Mehrheitsprädiktoren nicht untersucht.

Vor allem bei Verwendung des Verfahrens *Sicherer Zustand* und des Zuverlässigkeitszählers (zum Beispiel mit vier Zuständen) wird das sekundäre Auswahlkriterium entscheidend sein.

Tertiäres Auswahlkriterium

Wenn die höchste erreichte Zuverlässigkeitsstufe sehr niedrig ist, tritt das Problem auf, dass zur Vorhersage ein unzuverlässiger Prädiktor bzw. eine Menge von unzuverlässigen Prädiktoren für die Vorhersage des Hybridprädiktors verwendet

wird. Dies kann zum Beispiel auftreten, wenn bei der Verwendung des Verfahrens *Sicherer Zustand* alle Prädiktoren in einem schwachen Zustand sind und somit als unzuverlässig eingeordnet werden.

Um dieses Problem zu umgehen, kann ein tertiäres Auswahlkriterium verwendet werden, welches über die Verwendung einer Barriere entscheidet. Bei Verwendung der Barriere werden analog zu den Zuverlässigkeitsschätzern nur Prädiktoren betrachtet, deren Zuverlässigkeit über der vorgegebenen Barriere liegt. Zum Beispiel werden für den Zuverlässigkeitsprädiktor mit dem Verfahren *Sicherer Zustand* bei der Verwendung der Barriere nur noch Prädiktoren betrachtet, die sich in einem sicheren Zustand befinden.

Liegt die höchste erreichte Zuverlässigkeitsstufe über der vorgegebenen Barriere ändert sich gegenüber der Arbeitsweise ohne Barriere nichts. Andernfalls, wenn kein Prädiktor einer Zuverlässigkeitsstufe über der Barriere zugeordnet ist, macht der Hybridprädiktor keine Vorhersage. Der Hybridprädiktor kann somit als unzuverlässig angesehen werden.

Zusammenfassung Auswahlkriterien

In Tabelle 7.1 sind die Auswahlmöglichkeiten für die Verwendung eines Zuverlässigkeitsprädiktors aufgelistet. Insgesamt gibt es 12 Varianten eines Zuverlässigkeitsprädiktors, die in der Evaluierung untersucht werden sollen.

Tabelle 7.1: Auswahlkriterien Zuverlässigkeitsprädiktor

primär	sekundär	tertiär
Sicherer Zustand	relativ	ohne Barriere
		mit Barriere
	einfach	ohne Barriere
		mit Barriere
Schwellenwert-Verfahren	relativ	ohne Barriere
		mit Barriere
	einfach	ohne Barriere
		mit Barriere
Zuverlässigkeitszähler	relativ	ohne Barriere
		mit Barriere
	einfach	ohne Barriere
		mit Barriere

7.2 Evaluierung

Nach der Evaluierung der Warm-up-Prädiktoren, die immer einen Prädiktortyp mit verschiedenen Ordnungen verwenden, wird für die Evaluierung der Mehrheitsprädiktoren und Zuverlässigkeitsprädiktoren eine Menge P der verwendeten Prädiktoren benötigt. Dabei sollen vier verschiedene Mengen betrachtet werden, die in Tabelle 7.2 aufgelistet sind.

Tabelle 7.2: Prädiktormengen für die Mehrheitsprädiktoren und Zuverlässigkeitsprädiktoren

	P_1	P_2	P_3	P_4
lokaler einstufiger One-State-Prädiktor	•			
lokaler einstufiger Two-State-Prädiktor	•			
lokaler zweistufiger Two-State-Prädiktor Ordnung 1 - 5	•	•	•	
lokaler zweistufiger Two-State-Prädiktor PPM(5)	•			
lokaler zweistufiger Two-State-Prädiktor SPPM(5)	•			
lokaler zweistufiger Markov-Prädiktor Ordnung 1 - 5	•	•		•
lokaler zweistufiger Markov-Prädiktor PPM(5)	•			
lokaler zweistufiger Markov-Prädiktor SPPM(5)	•			
globaler zweistufiger Two-State-Prädiktor Ordnung 1 - 5	•	•	•	
globaler zweistufiger Two-State-Prädiktor PPM(5)	•			
globaler zweistufiger Two-State-Prädiktor SPPM(5)	•			
globaler zweistufiger Markov-Prädiktor Ordnung 1 - 5	•	•		•
globaler zweistufiger Markov-Prädiktor PPM(5)	•			
globaler zweistufiger Markov-Prädiktor SPPM(5)	•			

Auffallend an diesen Mengen ist die Aufnahme der PPM- und SPPM-Prädiktoren in P_1, welche ja selbst schon Hybridprädiktoren sind. Diese wurden zum Einen schon in den vorangegangenen Kapiteln evaluiert und können wegen ihrer Arbeitsweise nicht mit diesen Mengen bewertet werden. Zum Anderen soll die Verwendung in Menge P_1 zeigen, dass Hybridprädiktoren auch in der Menge der verwendeten Prädiktoren bereits Hybridprädiktoren enthalten können.

Evaluiert werden die Hybridprädiktoren wiederum für die Vorhersage des nächsten Raumes mit den Augsburg Benchmarks sowie für die Vorhersage der Zell-ID, des Location Area Code sowie der Aktivität mit den Nokia Context Daten.

7.2.1 Warm-up-Prädiktor

Die Warm-up-Prädiktoren PPM und SPPM wurden schon in den vorangegangenen Kapiteln evaluiert. Bei Betrachtung der Augsburg Benchmarks (siehe Abbildung 5.20) und der Nokia Context Daten (siehe Abbildung 5.24) ist keine Verbesserung der Vorhersagegenauigkeit bei den lokalen Prädiktoren zu verzeichnen. Die PPM- und SPPM-Varianten zeigen hier sogar das schlechteste Verhalten im Vergleich mit den Prädiktoren mit Ordnungen 1 bis 5. Der Grund für dieses Phänomen liegt darin, dass im lokalen Fall die Muster sehr oft aus den gleichen Kontexten bestehen, die durch wenige andere Kontexte unterbrochen werden. Folgende einfache Beispiele (siehe Tabellen 7.3 und 7.4) verdeutlichen dieses Verhalten.

Tabelle 7.3: SPPM-Prädiktor mit maximaler Ordnung 3 - Beispiel 1

Sequenz_1	(1)	(3)	SPPM(3)
a	–	–	–
a	a✓	–	a✓
a	a	–	a
b	–	–	–
a	a✓	–	a✓
a	a	–	a
b	a✓	a✓	a✓
a	a✓	a✓	a✓
a	a✓	b	b
a			
Treffergenauigkeit t	$\frac{15}{21} \sim 71,4\%$	$\frac{14}{21} \sim 66,7\%$	$\frac{12}{21} \sim 57,1\%$

Beide Beispiele zeigen je eine Sequenz von Kontexten, auf der basierend ein SPPM-Prädiktor der Ordnung 3 Vorhersagen macht. Zustands- und Markov-Prädiktoren liefern in diesen Beispielen die selben Vorhersagen. Spalte 1 zeigt die Sequenz, die mit jeder Zeile um einen Kontext erweitert wird. In Spalte 2 und 3 sind die Vorhersagen der Prädiktoren mit Ordnung 1 und 3 nach der bis dahin aufgetretenen Sequenz aufgelistet. Die vierte Spalte zeigt die Vorhersage des SPPM-Prädiktors mit maximaler Ordnung 3. Beide Beispiele zeigen, dass in einem ungünstigen Fall die Treffergenauigkeit eines Warm-up-Prädiktors schlechter sein kann, als die Treffergenauigkeiten der verwendeten Prädiktoren, und zwar tritt dies genau dann auf, wenn der Prädiktor mit maximaler Ordnung eine falsche Vorhersage und der Prädiktor mit Ordnung 1 eine richtige Vorhersage liefert.

In die Werte aus den Abbildungen 5.20 und 5.24 sind nur Vorhersagen aus angelernten Mustern eingegangen, d.h. nach Mustern, welche das erste Mal auftreten,

Tabelle 7.4: SPPM-Prädiktor mit maximaler Ordnung 3 - Beispiel 2

Sequenz_2	(1)	(3)	SPPM(3)
a	–	–	–
a	a✓	–	a✓
a	a	–	a
b	–	–	–
a	a✓	–	a✓
a	a	–	a
b	a✓	a✓	a✓
a	a✓	a✓	a✓
a	a	b	b
c			
Treffergenauigkeit t	$\frac{12}{21} \sim 57,1\%$	$\frac{14}{21} \sim 66,7\%$	$\frac{12}{21} \sim 57,1\%$

wird keine Vorhersage angeboten. Um diese Größe zu messen und zu vergleichen wurde die Quantität eingeführt, welche bei den PPM- und SPPM-Varianten die gleichen sehr guten Werte wie die Prädiktoren mit Ordnung 1 erreicht. Betrachtet man jetzt die absolute Vorhersagegenauigkeit, welche definiert wird als

$$t_{abs} = \frac{c}{v}$$

wobei c die Anzahl der korrekten Vorhersagen und v die Anzahl der gesamten Vorhersagen ist, d.h. v entspricht auch der Länge der Sequenz minus eins, die je Versuch aufgezeichnet wurden. Die absolute Treffergenauigkeit kann auch berechnet werden als Produkt aus Trefferwahrscheinlichkeit t ohne ungelernte Muster und Quantität q, d.h. $t_{abs} = t \cdot q$. Abbildung 7.6 zeigt die absoluten Trefferwahrscheinlichkeiten für den Kontext Zell-ID der Nokia Context Daten im lokalen und globalen Fall.

Bei Betrachtung der absoluten Treffergenauigkeit im lokalen Fall relativiert sich das Phänomen, da hier eine nicht gemachte Vorhersage aufgrund des ersten Auftretens eines Musters für den Prädiktor mit maximaler Ordnung als falsch gewertet wird. In der Anlernphase der einzelnen Muster wird beim Warm-up-Prädiktor auf einen Prädiktor mit kleinerer Ordnung zurückgegriffen. Im schlechtesten Fall macht dieser immer eine falsche Vorhersage und der Warm-up-Prädiktor hat die selbe absolute Treffergenauigkeit wie der Prädiktor mit maximaler Ordnung. Andernfalls verbessert sich die absolute Treffergenauigkeit gegenüber dem Prädiktor mit maximaler Ordnung, was die Diagramme verdeutlichen.

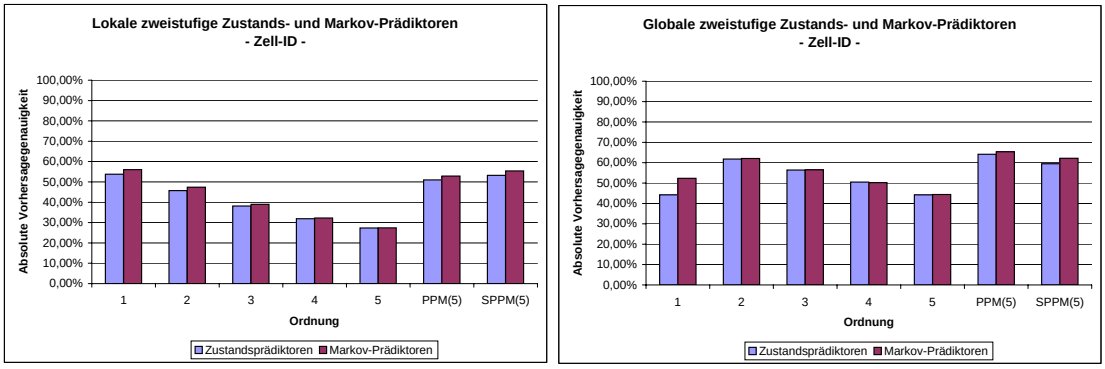


Abbildung 7.6: Absolute Vorhersagegenauigkeit der lokalen und globalen Zustands- und Markov-Prädiktoren - Zell-ID

Betrachtet man die globalen Prädiktoren (siehe Abbildungen 5.20 und 5.24), existieren nicht mehr so viele mögliche Muster, d.h. das Phänomen, dass die PPM- und SPPM-Varianten die niedrigste Trefferwahrscheinlichkeit liefern, trifft hier nicht mehr zu. In allen Messungen ist festzustellen, dass die Warm-up-Prädiktoren nie das schlechteste aber auch nie das beste Ergebnis liefern. Im Fall der Augsburg Benchmarks arbeiten die Warm-up-Prädiktoren immer besser als der Prädiktor mit maximaler Ordnung 5. Mit den Nokia Context Daten speziell der Vorhersage der Zell-ID und des Location Area Code sowie der Vorhersage der Aktivität arbeiten sie schlechter als die Prädiktoren mit maximaler Ordnung, da hier jeweils die Prädiktoren mit Ordnung 1 sehr schlechte Ergebnisse liefern. Bei Betrachtung der absoluten Trefferwahrscheinlichkeit bei Vorhersage der Zell-ID (Abbildung 7.6) arbeitet die PPM-Variante am besten gegenüber allen anderen Prädiktoren.

Bei einem Vergleich von PPM und SPPM ist festzustellen, dass bei den Augsburg Benchmarks die SPPM-Variante immer bessere Vorhersagegenauigkeiten als die PPM-Variante liefert. Bei den Nokia Context Daten ist es genau umgekehrt, d.h. die PPM-Variante ist besser als die SPPM-Variante. Dies liegt darin begründet, dass hier die Prädiktoren mit Ordnung 1 sehr niedrige Genauigkeiten liefern. Dieses Verhalten ist auch bei den absoluten Trefferwahrscheinlichkeiten zu beobachten.

Wie die vorangegangene Analyse gezeigt hat, bringen die Warm-up-Prädiktoren in den verwendeten Szenarios keine Verbesserungen. Nach Betrachtung der absoluten Treffergenauigkeit kann man feststellen, dass die Warm-up-Prädiktoren eingesetzt werden können, wenn der Speicheraufwand keine wesentliche Rolle spielt und eine falsche Vorhersage gegenüber keiner Vorhersage keine zu hohen Strafkosten verursacht.

7.2.2 Mehrheitsprädiktor

Tabelle 7.5 zeigt die Vorhersagegenauigkeiten der Mehrheitsprädiktoren mit den Augsburg Benchmarks sowie den Nokia Context Daten. Bei den Augsburg Benchmarks wurden auch die Trefferwahrscheinlichkeiten mit statischer Zuverlässigkeit gemessen. Dabei wurde wiederum das eigene Büro der jeweiligen Person in die Menge der unzuverlässigen Kontexte aufgenommen ($C_{unc} = \{\text{own office}\}$). Es wurde weiterhin die Quantität gemessen, da hier nicht die absolute Trefferwahrscheinlichkeit berechnet wurde. Als zugrunde liegende Mengen von Prädiktoren, die die Hybridprädiktoren verwenden, sind die zu Beginn dieses Abschnitts eingeführten Mengen P_1 bis P_4 eingesetzt worden. Die beste Treffergenauigkeit jedes Kontextes ist in der Tabelle hervorgehoben.

Es sollen zunächst die Augsburg Benchmarks ohne statische Zuverlässigkeit betrachtet werden. Die besten Vorhersageergebnisse für alle Personen liefert hier der Mehrheitsprädiktor mit einfacher Mehrheit und Prädiktormenge P_2. Für die Person B liefert dieser gegenüber den Vorhersagegenauigkeiten der einzelnen Prädiktoren aus der Prädiktormenge P_2 das beste Resultat. Bei Person A ist nur der lokale Zustandsprädiktor mit Ordnung 5 mit 84,13% besser (Quantität 23,68%). Bei Person C liefert nur der lokale Markov-Prädiktor mit Ordnung 5 eine bessere Genauigkeit mit 68,07% (Quantität 18,77%). Bei Person D erzielen die lokalen Zustands- und Markov-Prädiktoren mit Ordnungen 4 und 5 größere Trefferwahrscheinlichkeiten.

Betrachtet man die Quantität zeigt sich, dass die Mehrheitsprädiktoren mit relativer Mehrheit wie erwartet eine sehr hohe Quantität erzielen, da diese, sobald ein Prädiktor aus der Prädiktormenge ein Ergebnis liefern kann, eine Vorhersage machen. Die Prädiktoren mit einfacher Mehrheit weisen dagegen eine geringe Quantität zwischen 32% und 50% auf. Diese Werte liegen aber für Person A und C über der Quantität der einzelnen Prädiktoren, die bessere Treffergenauigkeiten lieferten als der Hybridprädiktor.

Bei der Einbeziehung der statische Zuverlässigkeit werden die besten Mehrheitsprädiktoren nicht mehr mit derselben Prädiktormenge erreicht. Wiederum sind aber die Prädiktoren mit einfacher Mehrheit besser als die Prädiktoren mit relativer Mehrheit. Bei Person A liefert der Prädiktor mit Prädiktormenge P_1 das beste Resultat, welcher aber nur gering besser ist als das Resultat mit Menge P_2 bzw. P_4. Alle drei Werte sind besser als die Genauigkeiten der einzelnen Prädiktoren. Die Quantität mit Menge P_1 ist aber mit 67% viel höher als mit den Mengen P_2 bzw. P_4 mit 54%. Ein Grund hierfür ist die Verwendung der PPM- und SPPM-Varianten in der Prädiktormenge P_1.

Tabelle 7.5: Mehrheitsprädiktor mit Augsburg Benchmarks und Nokia Context Daten

			P_1, rel.	P_1, einf.	P_2, rel.	P_2, einf.	P_3, rel.	P_3, einf.	P_4, rel.	P_4, einf.
Person A	$\mathcal{C}_{unc} = \emptyset$	Genauigkeit	51,17%	76,86%	53,13%	80,22%	48,83%	74,23%	55,47%	77,55%
		Quantität	96,24%	45,49%	96,24%	34,21%	96,24%	36,47%	96,24%	36,84%
	$\mathcal{C}_{unc} = \{$own office$\}$	Genauigkeit	77,78%	88,66%	80,00%	88,61%	77,78%	87,50%	80,00%	88,61%
		Quantität	93,75%	67,36%	93,75%	54,86%	93,75%	55,56%	93,75%	54,86%
Person B	$\mathcal{C}_{unc} = \emptyset$	Genauigkeit	50,78%	69,51%	51,07%	71,67%	48,65%	68,98%	50,92%	70,10%
		Quantität	98,05%	50,77%	98,05%	40,86%	98,05%	38,21%	98,05%	43,38%
	$\mathcal{C}_{unc} = \{$own office$\}$	Genauigkeit	71,98%	79,71%	72,75%	81,25%	71,47%	82,30%	72,49%	81,98%
		Quantität	96,77%	68,66%	96,77%	55,72%	96,77%	51,99%	96,77%	55,22%
Person C	$\mathcal{C}_{unc} = \emptyset$	Genauigkeit	46,23%	63,54%	46,87%	65,24%	44,62%	63,73%	48,96%	62,13%
		Quantität	98,26%	45,43%	98,26%	33,12%	98,26%	32,18%	98,26%	37,07%
	$\mathcal{C}_{unc} = \{$own office$\}$	Genauigkeit	69,01%	72,18%	69,01%	72,58%	66,20%	71,93%	69,86%	72,77%
		Quantität	97,26%	67,95%	97,26%	50,96%	97,26%	46,85%	97,26%	52,33%
Person D	$\mathcal{C}_{unc} = \emptyset$	Genauigkeit	52,65%	71,71%	54,62%	75,63%	50,88%	72,68%	54,03%	71,62%
		Quantität	97,51%	48,08%	97,51%	37,74%	97,51%	37,16%	97,51%	42,53%
	$\mathcal{C}_{unc} = \{$own office$\}$	Genauigkeit	74,02%	81,00%	75,09%	84,91%	72,95%	85,16%	74,02%	84,15%
		Quantität	95,90%	68,26%	95,90%	54,27%	95,90%	52,90%	95,90%	55,97%
Zell-ID		Genauigkeit	67,11%	81,21%	67,75%	82,86%	66,33%	83,33%	67,11%	82,37%
		Quantität	94,61%	60,64%	94,61%	51,88%	94,61%	48,94%	94,61%	52,14%
Location Area Code		Genauigkeit	77,11%	82,14%	73,63%	81,70%	73,63%	82,07%	74,13%	81,46%
		Quantität	97,10%	81,16%	97,10%	73,91%	97,10%	70,05%	97,10%	72,95%
Benutzeraktivität		Genauigkeit	78,91%	84,32%	78,95%	85,79%	78,67%	85,62%	78,33%	86,65%
		Quantität	99,71%	86,90%	99,71%	82,41%	99,71%	82,79%	99,71%	80,93%

Für Person B liefert der Prädiktor mit der Prädiktormenge P_3 das beste Resultat, welches sogar 6%-Punkte über dem besten Einzelresultat liegt. Bei Person C ist die höchste Trefferwahrscheinlichkeit mit P_4 zu finden. Dieses Ergebnis wird nur vom globalen Zustands- und Markov-Prädiktor mit Ordnung 4 übertroffen (75,42% und 74,86%). Bei Person D wird wie im Fall ohne statische Zuverlässigkeit das beste Resultat der Mehrheitsprädiktoren von lokalen Zustands- und Markov-Prädiktoren mit Ordnung 4 und 5 übertroffen. Das beste Resultat der Mehrheitsprädiktoren für Person D wird mit der Prädiktormenge P_3 erzielt.

Bei den Nokia Context Daten sind wiederum die Mehrheitsprädiktoren mit einfacher Mehrheit wesentlich besser als mit relativer Mehrheit. Bei der Vorhersage der Zell-ID liefert der Prädiktor mit der Menge P_3, also den Zustandsprädiktoren das beste Ergebnis, welches auch besser ist als alle Ergebnissen der einzelnen Prädiktoren. Der Prädiktor mit Menge P_1 erreicht für den Location Area Code die höchste Vorhersagegenauigkeit, die aber von den Treffergenauigkeiten der globalen Zustands- und Markov-Prädiktoren mit Ordnung 3, 4 und 5 übertroffen wird. Ein Grund könnte die geringe Anzahl von 207 Datensätzen sein. Das beste Resultat der Mehrheitsprädiktoren für die Vorhersage der Benutzeraktivität wird mit der Menge P_4 (also der Menge der Markov-Prädiktoren) erzielt. Dieses kann von keinem Ergebnis eines einzelnen Prädiktors übertroffen werden.

Die vorab beschriebenen Resultate führen zu der Erkenntnis, dass die Mehrheitsprädiktoren mit relativer Mehrheit keinen Vorteil bringen, sondern wesentlich schlechtere Ergebnisse als die einzelnen Prädiktoren liefern. Ein Einsatz des Mehrheitsprädiktor mit einfacher Mehrheit bietet auf jeden Fall eine über dem Durchschnitt liegende Trefferwahrscheinlichkeit. Natürlich steigt auch hier der Speicheraufwand enorm, was gegenüber den Einzelprädiktoren abzuwägen ist. Eine sinnvolle Prädiktormenge scheint die Menge P_2 zu sein. Sie beinhaltet die Zustands- sowie Markov-Prädiktoren im lokalen und globalen Fall mit den Ordnungen 1 bis 5 und kombiniert somit die Vorteile der einzelnen Prädiktortypen. Der Hybridprädiktor mit dieser Menge lieferte zwar nicht immer das beste Resultat, lag aber in solch einem Fall knapp unter dem besten Resultat.

7.2.3 Zuverlässigkeitsprädiktor

Für die Zuverlässigkeitsprädiktoren wurden wiederum die Prädiktormengen P_1 bis P_4 verwendet. Eine Ausnahme ist das Verfahren *Sicherer Zustand*, hier wurde nur die Menge P_3 eingesetzt, da dieses Verfahren nur mit Zustandsprädiktoren arbeiten kann. Bei der Evaluierung wurden alle Varianten aus Tabelle 7.1 untersucht.

Im Vergleich der vier Prädiktormengen unter Verwendung des Zuverlässigkeitszählers und des Schwellenwert-Verfahrens ist festzustellen, dass sich die Hybridprädiktoren mit den Mengen P_2, P_3 und P_4 annähernd gleich verhalten. Un-

ter diesen Mengen sind nur Abweichungen bis zu maximal einem Prozentpunkt auszumachen. Aus diesem Grund soll für die Zuverlässigkeitsprädiktoren mit Zuverlässigkeitszähler und mit Schwellenwert-Verfahren im Folgenden die Prädiktormenge P_2 betrachtet werden, welche die bei den Mehrheitsprädiktoren erwähnten Vorteile der lokalen und globalen sowie der Zustands- und Markov-Prädiktoren kombiniert.

Abbildung 7.7 zeigt die erreichten Vorhersagegenauigkeiten mit Prädiktormenge P_2 für den Zuverlässigkeitszähler und das Schwellenwert-Verfahren sowie mit Prädiktormenge P_3 für das Verfahren *Sicherer Zustand* am Beispiel der Person D der Augsburg Benchmarks. Abbildung 7.8 zeigt die gleichen Messungen für den Kontext Zell-ID der Nokia Context Daten.

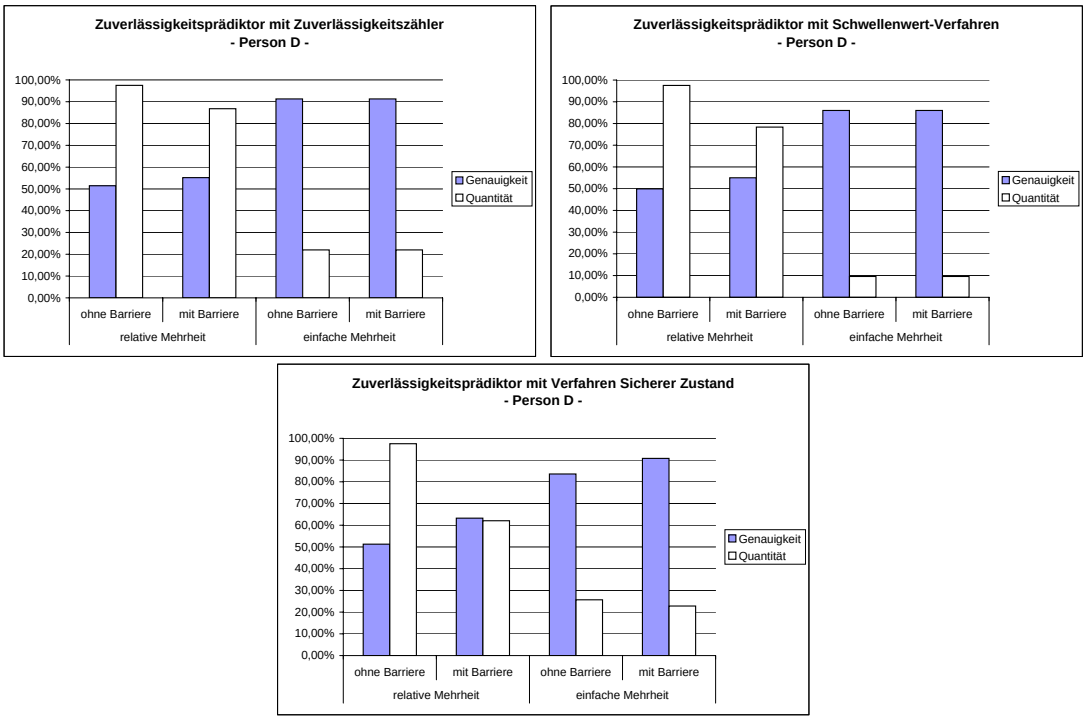


Abbildung 7.7: Zuverlässigkeitsprädiktoren - Person D

Tertiäres Auswahlkriterium

Beim Schwellenwert-Verfahren wurde als Barriere 60% und beim Zuverlässigkeitszähler wurden vier Zustände und $k = 10$ gewählt.

Für das Schwellenwert-Verfahren und den Zuverlässigkeitszähler wurde festgestellt, dass sich jeweils die einfache Mehrheit mit und ohne Barriere in den meisten Fällen identisch verhalten. Geringe Unterschiede waren nur in den kleineren

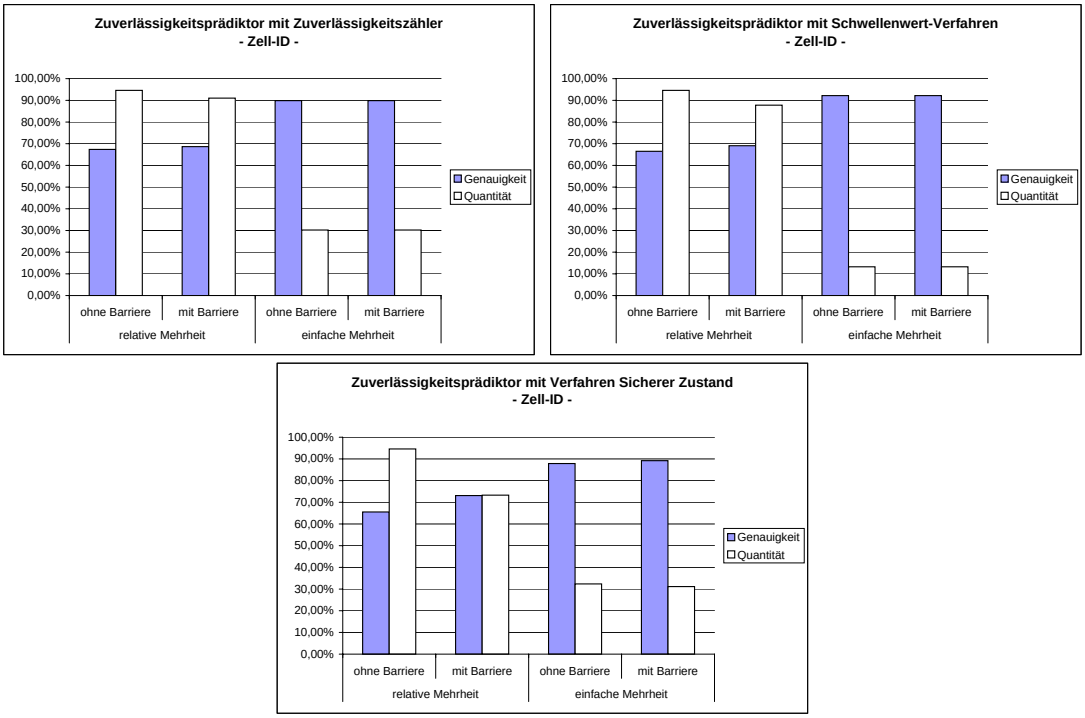


Abbildung 7.8: Zuverlässigkeitsprädiktoren - Zell-ID

Prädiktormengen P_3 und P_4 festzustellen. In diesen Fällen war der Einsatz der Barriere und der einfachen Mehrheit wie erwartet besser.

Die identischen Ergebnisse bei einfacher Mehrheit mit und ohne Barriere zeigen, dass der Einsatz der Barriere als Kontrolle gesehen werden kann, da alle Prädiktoren, die als zuverlässig eingestuft werden, dieselbe Zuverlässigkeitsstufe haben. Das bedeutet, wenn mehr als die Hälfte der Prädiktoren auf der höchsten Zuverlässigkeitsstufe dasselbe Vorhersageergebnis liefern, dann ist diese Zuverlässigkeitsstufe auch größer oder gleich der verwendeten Barriere.

Bei Verwendung des Verfahrens *Sicherer Zustand* ist eine deutliche Steigerung der Variante mit einfacher Mehrheit mit Barriere gegenüber einfacher Mehrheit ohne Barriere zu sehen. Der Grund hierfür liegt darin, dass es bei diesem Verfahren nur zwei Zuverlässigkeitsstufen gibt. Dabei kann es oft vorkommen, dass alle Prädiktoren sich in der niedrigen Stufe befinden.

Der Einsatz der Barriere bei Verwendung der relativen Mehrheit brachte bei allen Messungen zwar eine Verbesserung, aber gegenüber der einfachen Mehrheit sind diese Ergebnisse sehr schlecht.

Sekundäres Auswahlkriterium

Wie aus der Betrachtung der Mehrheitsprädiktoren vermutet, arbeiten auch hier alle Zuverlässigkeitsprädiktoren, die die relative Mehrheit verwenden, schlechter als die einzelnen Prädiktoren mit der entsprechenden Zuverlässigkeitsanalyse (siehe Kapitel 6). Die Zuverlässigkeitsprädiktoren, welche die einfache Mehrheit verwenden, sind aber immer besser als der beste Einzelprädiktor mit dem gleichen Zuverlässigkeitsverfahren.

Primäres Auswahlkriterium

Im Vergleich der drei Zuverlässigkeitsverfahren untereinander ist kein bestes Verfahren auszumachen. Betrachtet man die Person D, liefert der Prädiktor mit Schwellenwert-Verfahren das schlechteste Ergebnis, und die Verwendung des Zuverlässigkeitszählers erzielt die höchste Genauigkeit. Im Fall der Zell-ID liefert jedoch das Schwellenwert-Verfahren das beste Resultat, und das Verfahren *Sicherer Zustand* zeigt das schlechteste Verhalten. Auch bei den hier nicht gezeigten Treffergenauigkeiten bei Vorhersage der anderen Kontexte ist kein Gewinner unter den drei Verfahren auszumachen. Bezieht man die Quantität mit in die Beurteilung ein, welches Verfahren die besten Ergebnisse liefert, ist das Schwellenwert-Verfahren im Nachteil gegenüber den anderen beiden Verfahren. Die schlechte Quantität des Schwellenwert-Verfahrens, die hier am Beispiel der Person D und der Zell-ID gezeigt wurde, war bei allen anderen Kontexten meist noch schlechter. Dabei wurde sogar mehrmals eine Quantität von nur 2% gemessen.

Bei allen Verfahren ist zu beobachten, dass eine höhere Genauigkeit nur mit einer geringeren Quantität zu erreichen ist. Einen sehr großen Unterschied gibt es zwischen der Quantität mit relativer und einfacher Mehrheit. Dabei schneidet das Schwellenwert-Verfahren am schlechtesten ab, was am kontinuierlichen Verlauf der Zuverlässigkeitsstufen liegen könnte, so dass immer nur sehr wenige Prädiktoren aus der Prädiktormenge auf der höchsten Zuverlässigkeitsstufe liegen.

Zusammenfassend ist für den Hybridprädiktor mit Zuverlässigkeitsverfahren festzustellen, dass die Verwendung der Prädiktormenge P_2 meist die besten Ergebnisse lieferte oder die gelieferten Ergebnisse nur knapp unterhalb der besten Ergebnisse der einzelnen Prädiktoren der Prädiktormenge zu finden waren. Mit der einfachen Mehrheit unter Einsatz der Barriere wurden in allen Messungen die höchsten Treffergenauigkeiten erzielt. Das Schwellenwert-Verfahren zeigte immer die niedrigste Quantität, welche nicht mehr akzeptabel erscheint.

7.3 Fazit Hybridprädiktoren

Beim Vergleich aller Hybridverfahren untereinander, fällt zunächst auf, dass die Warm-up-Prädiktoren nicht die gewünschte Verbesserung brachten, wogegen mit den Mehrheitsprädiktoren sowie mit den Zuverlässigkeitsprädiktoren eine Steigerung der Vorhersagegenauigkeit bzw. eine Vorhersagegenauigkeit über dem Durchschnitt der Vorhersagegenauigkeiten der einzelnen Prädiktoren der Prädiktormenge festzustellen war. Die Entscheidung, welches Mehrheitsverfahren zu wählen ist, fällt sowohl bei den Mehrheitsprädiktoren als auch beim sekundären Auswahlkriterium der Zuverlässigkeitsprädiktoren auf die einfache Mehrheit. Bei den Zuverlässigkeitsprädiktoren ist wie im vorherigen Abschnitt schon beschrieben kein Favorit unter den verwendeten Zuverlässigkeitsverfahren zu finden. Eine Verwendung der Barriere bringt keine bzw. für das Verfahren *Sicherer Zustand* eine kleine Verbesserung. Beim Vergleich der Mehrheitsprädiktoren mit den Zuverlässigkeitsprädiktoren fällt auf, dass die Zuverlässigkeitsprädiktoren immer eine höhere Treffergenauigkeit erreichen, wogegen aber die Mehrheitsprädiktoren in allen Tests eine deutlich bessere Quantität gegenüber der sehr niedrigen Quantität der Zuverlässigkeitsprädiktoren zeigen. P_2 scheint eine gute Wahl für die Prädiktormenge zu sein. P_2 kombiniert die Vorteile der Zustands- und Markov-Prädiktoren sowie die Vorteile der globalen und lokalen Prädiktoren.

8 Zeitvorhersage

Die Vorhersage des nächsten Kontextes bietet einem Benutzer schon einen gewissen Komfort. Eine Steigerung des Komforts kann durch die Angabe erfolgen, zu welchem Zeitpunkt der nächste Kontext eintreten wird. Diese zusätzliche Angabe der erwarteten Zeit ist besonders für die in den vergangenen Kapiteln beschriebenen Szenarios (z. B. ein Besucher wartet auf einen Angestellten) unerlässlich. Um den Zeitpunkt des nächsten Kontextwechsels zu bestimmen, kann die Verweildauer im aktuellen Kontext des Benutzers vorhergesagt werden. Mit dem Zeitpunkt des letzten Kontextwechsels kann dann die Zeit berechnet werden, zu der der nächste Kontext eintreten wird.

Für die Vorhersage der Verweildauer im aktuellen Kontext gibt es zwei Möglichkeiten. Zum Einen kann eine Vorhersage parallel zur Kontextvorhersage vorgenommen werden. Zum Anderen kann eine Zeitvorhersage in die verwendeten Prädiktoren integriert werden.

8.1 Parallele Modellierung

Bei der parallelen Modellierung der Vorhersage der Zeitdauer wird in den Ablauf des verwendeten Prädiktors nicht eingegriffen. Deshalb kann diese Modellierung auch zusammen mit allen möglichen Verfahren verwendet werden. Das Ziel ist hierbei, die Dauer vorherzusagen, die sich ein Benutzer in einem bestimmten Kontext befindet. Mit Hilfe dieser prognostizierten Dauer und dem Wissen, wie lange sich der Benutzer schon im aktuellen Kontext befindet, kann dann der Zeitpunkt vorhergesagt werden, zu dem der Benutzer in den vom eingesetzten Prädiktor vorhergesagten Kontext wechselt.

Die Frage bei der Vorhersage der Zeit ist nun, von welchen Parametern die Aufenthaltsdauer eines Benutzers in einem Kontext abhängt. Betrachtet man den Ort als Kontext, zum Beispiel die Räume in einem Bürogebäude, ist festzustellen, dass die Aufenthaltsdauer in einem Raum zunächst davon abhängt, wie lange sich ein Angestellter in der Vergangenheit in diesem Raum aufgehalten hat. Zum Beispiel wird der Angestellte sich immer nur sehr kurz im Druckerraum aufhalten. Gleiches gilt wahrscheinlich für die Küche. Hier kommt der Angestellte nur vorbei, um sich zum Beispiel eine Tasse Kaffee zu holen. Ein anderes Verhalten ist

vermutlich bei den Verweilzeiten im eigenen Büro des Angestellten vorzufinden. Die Aufenthaltsdauer im eigenen Büro wird sehr unterschiedlich sein.

Für diese aufgezeigte Abhängigkeit bieten die statistischen Parameter Mittelwert und Median der vergangenen Aufenthaltsdauer einen einfachen Ansatz. Es wird somit für jeden Kontext separat der Mittelwert bzw. Medians berechnet. Der Mittelwert ist das arithmetische Mittel aller Werte. Er wird berechnet, indem die Summe aller Werte durch die Anzahl aller Werte dividiert wird. Seien $x_1, ..., x_n$ die Verweilzeiten des zu untersuchenden Kontextes. Der Mittelwert $\bar{x}$ ist dann definiert als

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

Der mittlere Wert einer geordneten Liste aller Werte wird als Median bezeichnet. Der Median ist also der Wert, der von gleich vielen kleineren und größeren Werten eingerahmt wird. Seien $x_{(1)} \leq ... \leq x_{(n)}$ die der Größe nach geordneten Verweilzeiten des zu untersuchenden Kontextes. Der Median $\tilde{x}$ ist dann definiert als

$$\tilde{x} = \begin{cases} x_{((n+1)/2)} & \text{, falls n ungerade} \\ \frac{1}{2}(x_{(n/2)} + x_{((n+2)/2)} & \text{, falls n gerade} \end{cases}$$

Eine weitere Abhängigkeit für die Verweildauer in einem Kontext kann auch der aktuelle Zeitpunkt wie die Tageszeit, der Wochentag oder der Monat sein. Ein Angestellter verlässt das Bürogebäude zum Beispiel immer mittags für eine halbe Stunde, um zur Kantine zu gehen. Wenn er aber am Nachmittag das Gebäude verlässt, findet wahrscheinlich eine Besprechung außerhalb statt. Betrachtet man zum Beispiel die Abwesenheit am Abend, kann mit großer Sicherheit davon ausgegangen werden, dass der Angestellte erst am nächsten Morgen wieder im Büro sein wird. Wenn es dann noch Freitag ist, kann weiterhin davon ausgegangen werden, dass der Angestellte über das ganze Wochenende abwesend sein wird. Natürlich wird auch anderes Wissen über die Person bzw. das Umfeld eine Verbesserung der Vorhersage der Verweildauer bringen. Diese Abhängigkeit soll aber nicht weiter untersucht werden. Weiterhin könnten auch Abhängigkeiten zur Aufenthaltsdauer in vorherigen Kontexten existieren. Auch diese sowie andere Abhängigkeiten, die sicherlich vorhanden sind, sollen hier nicht näher untersucht werden.

Eine weitere Überlegung betrifft die Zuverlässigkeit der Vorhersage der Verweildauer. Man könnte wiederum die Vorhersage als zuverlässig bzw. nicht zuverlässig einstufen. Dabei kann aber bei der parallelen Modellierung das Phänomen auftreten, dass ein Prädiktor aufgrund hoher Zuverlässigkeit eine Vorhersage macht und die Vorhersage der Verweildauer wegen niedriger Zuverlässigkeit nicht vorgenommen wird. Da aber die Vorhersage der Verweildauer bei der Vorhersage

des nächsten Kontextes unerlässlich ist, muss die Zeitvorhersage vorgenommen werden, wenn die Kontextvorhersage als zuverlässig eingestuft wird. Dieser Zusammenhang gilt für die dynamischen Verfahren der Zuverlässigkeitsanalyse. Die statische Zuverlässigkeit lässt sich im Gegensatz dazu sehr gut nutzen, da es Kontexte gibt, für die es unmöglich ist, eine Verweildauer anzugeben. Dies wären zum Beispiel in einem Bürogebäude das eigene Büro eines Angestellten. Betrachtet man die Abwesenheit eines Angestellten, ist festzustellen, dass diese sehr unregelmäßig ist. Zum Mittag verlässt der Angestellte das Gebäude für eine halbe Stunde, abends geht er für 14 Stunden nach Hause und am Wochenende ist er schließlich 65 Stunden abwesend. Dieses Beispiel zeigt, dass bei der Verwendung des Mittelwertes oder Medians auch die Abwesenheit vom Bürogebäude einen unzuverlässigen Kontext darstellt.

8.2 Integrierte Modellierung

Für eine integrierte Modellierung der Zeitvorhersage muss ein verwendetes Vorhersageverfahren verändert werden bzw. neu modelliert werde. Wird ein Bayessches Netz zur Vorhersage des Kontextes verwendet, ist eine Erweiterung um die Vorhersage der Verweildauer durch eine neue Modellierung des Bayesschen Netzes möglich [Pie04, PPB+05].

Die Zustandsprädiktoren bieten diese Möglichkeit nicht. Hier könnte eine Erweiterung um Zeit durch eine Änderung der Anpassung des Prädiktors vorgenommen werden. Dabei wird die Anpassung des Zustandsgraphen nicht mehr durch einen Kontextwechsel ausgelöst, sondern durch einen erreichten Zeitpunkt. Zum Beispiel wird aller fünf Minuten der Zustandsgraph angepasst. Dadurch können jetzt auch im globalen Fall Muster mit zwei gleichen aufeinander folgenden Kontexten auftreten. Auch wird in diesem Fall die bisher evaluierte Länge der Muster von fünf nicht mehr ausreichen. Die Folge dieser beiden Änderungen ist, dass hierbei wesentlich mehr Speicherplatz benötigt wird, was wiederum für den Einsatz in ressourcenschwachen Geräten wenig Sinn macht. Um die Ressource Speicher zu schonen, könnte die Zeitspanne zur Anpassung vergrößert werden. Hierbei tritt das Problem auf, dass Übergänge zwischen zwei Zeitpunkten nicht registriert werden. Wenn zum Beispiel als Zeitspanne eine Stunde gewählt wird, können Bewegungen wie ein Gang zum Drucker zwischen zwei Messungen nicht wahrgenommen werden. Eine weitere Idee wäre eine Mischform, d.h. eine Anpassung in bestimmten Zeitabständen sowie eine Anpassung bei einem Kontextwechsel.

Eine Evaluierung dieser Ideen wurde aufgrund der genannten Probleme nicht vorgenommen. Als Ausblick bezüglich der Zeitvorhersage soll vorweggenommen werden, dass eine integrierte Modellierung der Zeit in einem nächsten Schritt untersucht werden sollte. Dabei ist es interessant einen Mittelweg zwischen den

beiden aufgezeigten Extremen zu finden, d.h. zwischen einer kurzen Zeitspanne, die die Ressource Speicher enorm beansprucht, und einer langen Zeitspanne, die kurzzeitige Bewegungen nicht registriert. Eine weitere Voraussetzung für solch eine Untersuchung ist auch eine größere Datenbasis als die hier verwendete.

8.3 Evaluierung

In der Evaluierung wird nur die parallele Modellierung untersucht, d.h. die Vorhersage der Verweildauer mittels Mittelwert und Median. Zur Bewertung wurden die Augsburg Benchmarks und die Nokia Context Daten verwendet.

In ubiquitären Systemen, vor allem in den beschriebenen Szenarios, ist eine minuten- oder gar sekundengenaue Vorhersage der Aufenthaltsdauer in einem Kontext nicht nötig und auch in vielen Fällen nicht sinnvoll. Desweiteren wäre eine solche Vorhersage sehr schwer zu bewerten, da ein Vergleich mit der tatsächlichen Zeit so gut wie nie zu einer Übereinstimmung führen würde. Deshalb soll zunächst überlegt werden, für welche Zeitstufen eine Vorhersage sinnvoll ist. Für viele Anwendungen, zum Beispiel ein Besucher wartet auf einen Angestellten, erscheint eine Unterteilung in 5-Minuten-Schritte sinnvoll:

$$5, 10, 15, 20, \ldots$$

Dies bedeutet, wenn die Vorhersage 7 Minuten ist, wird dem Benutzer 10 Minuten als Vorhersage angezeigt. Stellt sich nun als tatsächliche Zeit 6 Minuten heraus, wird die Vorhersage als korrekt eingestuft, da 6 größer ist als die nächstkleinere Stufe 5.

Wenn als Ortsinformation die Zell-ID verwendet wird, ist eine bessere Unterscheidung im unteren Bereich der Zeitstufen sinnvoll. Deshalb soll eine weitere Unterteilung betrachtet werden, in der eine weitere Stufe verwendet wird:

$$1, 5, 10, 15, 20, \ldots$$

Aus Sicht eines Benutzers ist es weiterhin weniger sinnvoll bei längeren Zeitangaben eine Unterteilung in 5-Minuten-Schritte zu machen. Zum Beispiel wird die Angabe, dass der Mitarbeiter in 40 Minuten zurückkommt oder in 45 Minuten, vom Benutzer vermutlich als gleich lang angesehen. Deshalb wird eine weitere Unterteilung betrachtet, bei der höhere Stufen zusammengefasst werden:

$$5, 10, 15, 20, 30, 45, 60, 120, > 120$$

Es sollen somit in der Evaluierung die drei Unterteilungen betrachtet werden, die im Folgenden noch einmal sortiert von fein bis grob aufgelistet sind.

$$\downarrow \begin{array}{ll} \text{Unterteilung 1:} & 1, 5, 10, 15, 20, \ldots \\ \text{Unterteilung 2:} & 5, 10, 15, 20, \ldots \\ \text{Unterteilung 3:} & 5, 10, 15, 20, 30, 45, 60, 120, > 120 \end{array}$$

Die Abbildungen 8.1 und 8.2 zeigen die Ergebnisse der Evaluierung mit den Augsburg Benchmarks und den Nokia Context Daten. Dabei wurden in allen Messungen jeweils der Median und Mittelwert für alle drei vorgestellten Unterteilungen berechnet. Für die Augsburg Benchmarks wurde zusätzlich noch die statische Zuverlässigkeit untersucht. Dabei wurde zunächst das eigene Büro in die Menge der unzuverlässigen Kontexte C_{unc} aufgenommen, da die Aufenthaltsdauer im eigenen Büro stark variiert. Außerdem schwankt die Dauer der Abwesenheit einer Person sehr stark, so dass in einem weiteren Schritt auch die Abwesenheit in die Menge der unzuverlässigen Kontexte C_{unc} aufgenommen wurde.

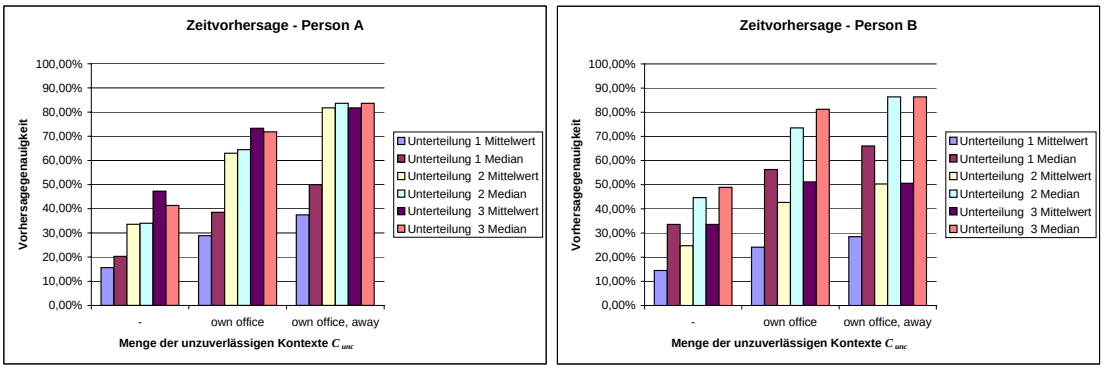


Abbildung 8.1: Zeitvorhersage ohne und mit statischer Zuverlässigkeit - Person A und B

Beim ersten Betreten eines Raumes bzw. Ortes sowie beim ersten Ausführen einer Aktivität ist keine Vorhersage der Verweildauer möglich, da die Berechnung des Mittelwertes bzw. des Medians mindestens einen Wert benötigt. Das heißt die Quantität der Vorhersagen ist wiederum unter 100%. Die kleinste Quantität wurde bei Person A mit statischer Zuverlässigkeit ermittelt, bei der das eigene Büro und die Abwesenheit als unzuverlässig eingestuft wurden. Die gemessene Quantität im beschriebenen Fall betrug 92,8%. Damit errechnet sich die Trefferwahrscheinlichkeit für die Vorhersage der Verweildauer wieder wie folgt:

$$t = \frac{c}{v_l}$$

wobei c die Anzahl der korrekten Vorhersagen und v_l die Anzahl der lieferbaren Vorhersagen ist. Die lieferbaren Vorhersagen sind die Vorhersagen, für die der Mittelwert bzw. der Median berechnet werden konnte.

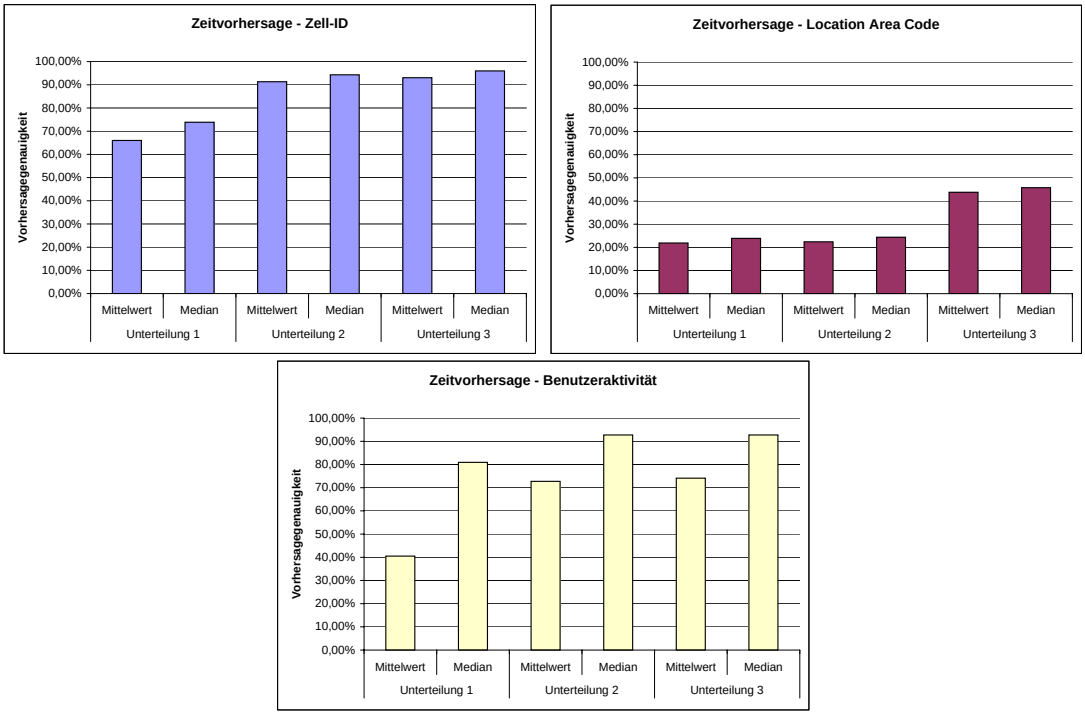


Abbildung 8.2: Zeitvorhersage - Nokia Context Daten

Zunächst soll der Unterschied zwischen Mittelwert und Median betrachtet werden. Ein klarer Vorteil des Median gegenüber dem Mittelwert ist bei Person B (Augsburg Benchmarks) und der Benutzeraktivität (Nokia Context Daten) zu erkennen. Bei diesen beiden Messreihen scheint es so genannte Ausreißer zu geben, die sehr stark von den Werten der Messreihe abweichen. Die Auswirkungen dieses Problems werden mit Hilfe des Median beseitigt. Bei allen anderen Messungen außer für Person A traten auch geringe Verbesserungen bei Verwendung des Medians gegenüber dem Mittelwert auf. Bei Person A wurden mit dem Mittelwert bei der Zeitunterteilung 3 mit $\mathcal{C}_{unc} = \emptyset$ sowie $\mathcal{C}_{unc} = \{\text{own office}\}$ bessere Werte als mit dem Median erzielt. Ein Grund hierfür liegt in der groben Zeitunterteilung im oberen Bereich, da bei Einordnung des Kontextes Abwesenheit als unzuverlässig dieses Phänomen nicht mehr auftritt.

Beim Vergleich der verschiedenen Zeitunterteilungen wurden wie erwartet bei den Augsburg Benchmarks mit der Unterteilung 3 die besten Resultate erzielt. Betrachtet man die Nokia Context Daten tritt dieser Vorteil nur bei dem Location Area Code auf, da hier die Aufenthaltsdauer sehr lang ist, d.h. bei einer groben Unterteilung im oberen Bereich werden natürlich auch die Treffer erhöht. Die Verweildauer für die Zell-ID sowie für die Benutzeraktivität bewegt sich dagegen im unteren Bereich, d.h. hier bringt die Grobunterteilung im oberen Bereich keine Verbesserung aber auch keine Verschlechterung. Bei Hinzunahme der 1-Minuten-

Stufe verhält es sich genau umgekehrt. Mit dem Location Area Code ist kein Unterschied festzustellen. Hingegen verschlechtern sich die mit der Zell-ID und der Benutzeraktivität erreichten Ergebnisse mit einer feineren Unterteilung im unteren Bereich. Eine Verschlechterung ist auch bei den Augsburg Benchmarks festzustellen, wenn die 1-Minuten-Stufe hinzugenommen wird.

Die Verwendung der statischen Zuverlässigkeit mit den Augsburg Benchmarks brachte wie erwartet eine Verbesserung. Im ersten Schritt wurde das eigene Büro des Mitarbeiters als unzuverlässig eingestuft, da die Verweildauer im eigenen Büro großen Schwankungen unterliegt. Im zweiten Schritt wurde noch die Abwesenheit als unzuverlässig eingestuft. Für beide Messungen wurden starke Verbesserungen der Trefferwahrscheinlichkeit registriert.

Ein Problem dieser einfachen Verfahren ist unter Umständen ein Festlernen, d.h. wenn über einen längeren Zeitraum der Benutzer immer ungefähr gleich lang in einem Kontext verbleibt, wird der Mittelwert sowie der Median gute Ergebnisse liefern. Ändert aber jetzt der Benutzer dieses Verhalten und verbleibt nun viel länger in diesem Kontext, wird der Mittelwert sowie der Median falsche Ergebnisse liefern. Dieses Problem könnte umgangen werden, indem für die Ermittlung des Mittelwertes bzw. des Median nur die letzten 200 Verweilzeiten betrachtet werden. Diese Einschränkung dient auch der Begrenzung der zusätzlichen Speicherkosten, die durch die parallele Zeitvorhersage entstehen.

8.4 Fazit Zeitvorhersage

Zusammenfassend kann geschlossen werden, dass eine Vorhersage der Aufenthaltsdauer im aktuellen Kontext mit dem Median sehr gut möglich ist, wenn diese relativ kurz sind. Dies verdeutlichen die Ergebnisse mit den Augsburg Benchmarks, wo bei der Einordnung des eigenen Büros und der Abwesenheit in die Menge der unzuverlässigen Kontexte eine hohe Treffergenauigkeit erreicht wurde, da dann nur noch Räume mit kurzer Aufenthaltsdauer betrachtet werden. Analog verhält es sich bei den Nokia Context Daten, für Kontexte mit kurzer Aufenthaltsdauer wie zum Beispiel der Zell-ID und der Benutzeraktivität ist eine Vorhersage mittels Median sehr gut möglich. Ist die Aufenthaltsdauer aber länger, ist eine gute Vorhersage nicht möglich, wie die Ergebnisse der Untersuchung des Location Area Code zeigen. Als beste Unterteilung der Zeit in verschiedene Stufen zeigt sich die Unterteilung 3, welche die feinen Abstufungen im unteren und oberen Bereich nicht hat.

9 Vergleich mit anderen Verfahren

In diesem Kapitel sollen die vorgestellten Zustands- sowie die Markov-Prädiktoren unter Verwendung der gezeigten Optimierungen wie Zuverlässigkeitsschätzung und Hybridprädiktoren mit anderen Vorhersageverfahren verglichen werden, die sich in vielen Bereichen der Informatik sowie auch zur Kontexterkennung und -vorhersage schon bewährt haben. Zum Vergleich werden dabei die dynamischen Bayesschen Netze [Pie04, PPB+05] und die Neuronalen Netze herangezogen werden. Als Vertreter der Neuronalen Netze werden das Multi-Layer-Perzeptron [VGPU04a, VGPU04b] sowie das Elman-Netz [Kuh05] betrachtet. Im Folgenden werden die Verfahren überblicksmäßig vorgestellt. Nähere Erklärungen sind den Literaturangaben zu entnehmen. Hidden-Markov-Modelle stellen ein weiteres Verfahren dar, welches für die Vorhersage des nächsten Kontextes eingesetzt werden kann. Die Hidden-Markov-Modelle stellen aber nur einen Spezialfall der dynamischen Bayesschen Netze dar [Mur02] und werden deshalb hier nicht näher betrachtet.

Bevor ein Vergleich der Verfahren bezüglich bestimmter Kriterien vorgenommen werden kann, werden die drei Verfahren Bayessche Netze, Multi-Layer-Perzeptron und Elman-Netz vorgestellt. Da diese drei Verfahren jeweils nur mit den Augsburg Benchmarks bewertet wurden, werden auch im Vergleich nur diese Benchmarks betrachtet. Verfahren wie Neuronale Netze werden vor ihrem eigentlichen Einsatz zunächst mit bestimmten Trainingsdaten angelernt. Aus diesem Grund wird in der Evaluierung wie folgt vorgegangen. Die Sommerdaten der Augsburg Benchmarks dienen als Trainingsdaten. Die Herbstdaten werden dann zur Messung der Vorhersagegenauigkeit sowie der Quantität verwendet. Für alle Verfahren werden dann die Ergebnisse mit Training mit den Ergebnissen ohne Training – also nur anhand der Herbstdaten gemessen – verglichen. Desweiteren sind die drei folgenden Verfahren für die Vorhersage des nächsten Aufenthaltsortes eines abwesenden Mitarbeiters modelliert. Dabei wird die Vorhersage am Türschild des Büros des Mitarbeiters angezeigt, so dass eine Vorhersage aus dem eigenen Büro nicht notwendig ist. Deshalb wird für alle Verfahren die statische Zuverlässigkeit betrachtet, wobei das eigene Büro in die Menge der unzuverlässigen Kontexte aufgenommen wird, d.h. eine Vorhersage, welcher Raum nach dem eigenen Büro betreten wird, wird nicht vorgenommen.

9.1 Bayessche Netze

Bayessche Netze eignen sich besonders gut dazu, Abhängigkeiten von Ereignissen direkt auszudrücken. Ein Bayessches Netz (BN) [Jen96] ist ein gerichteter azyklischer Graph. Die Knoten des Graphen repräsentieren die Variablen und die Kanten die Abhängigkeiten zwischen diesen Variablen. Jede Variable besitzt eine endliche Anzahl von sich gegenseitig ausschließenden Zuständen und die durch andere Variablen bedingten Wahrscheinlichkeiten für jeden dieser Zustände. Die Pfeilrichtung zwischen zwei Knoten $B \rightarrow A$ impliziert eine Abhängigkeit des Knotens A von seinem Vaterknoten B. Der Knoten A mit Vaterknoten $B_1, \ldots, B_n$ kennt somit die Wahrscheinlichkeiten für $P(A|B_1, \ldots, B_n)$ für alle Zustände von $A, B_1, \ldots, B_n$. Falls ein Knoten A keinen Vaterknoten besitzt, enthält dieser lediglich die unbedingten Wahrscheinlichkeiten $P(A)$ für alle Zustände von A. Falls zwischen zwei Knoten im Bayesschen Netz keine Kante existiert, sind diese beiden Ereignisse nicht direkt voneinander abhängig.

Wenn sich die Abhängigkeiten über die Zeit hinweg verändern oder sogar Variablen hinzugekommen oder verschwinden, ist das Modell der Bayesschen Netze nicht mehr ausreichend. Eine Erweiterung sind die dynamischen Bayesschen Netze (DBN) [Mur02, WM03]. Ein dynamisches Bayessches Netz besteht aus einer Menge von Zeitscheiben und spiegelt somit eine Zeitreihe wieder. Von Zeitscheibe zu Zeitscheibe können Änderungen bezüglich der Variablen und ihrer Abhängigkeiten modelliert werden. Außerdem können in dynamischen Bayesschen Netzen auch Abhängigkeiten zwischen Variablen verschiedener Zeitscheiben modelliert werden.

Ein Bayesschen Netz speichert für jede Variable X eine Tabelle mit den bedingten Wahrscheinlichkeiten $P(X|v(X))$, wobei $v(X)$ die Menge aller Vaterknoten von X ist. Die Wahrscheinlichkeitsverteilung über $X_1, ..., X_n$ wird dann mittels der Kettenregel definiert [DH93]:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|v(X_i))$$

Damit können die Wahrscheinlichkeiten in einem Bayesschen Netz zwar effizient gespeichert werden, aber sie sind nicht effizient berechenbar.

9.1.1 Modellierung

Für die Modellierung des Problems der Vorhersage des nächsten Aufenthaltsortes sowie des Zeitpunktes des Übergangs zum nächsten Aufenthaltsort, wurde ein dynamisches Bayessches Netz gewählt. Um ein Bayessches Netz zu modellieren,

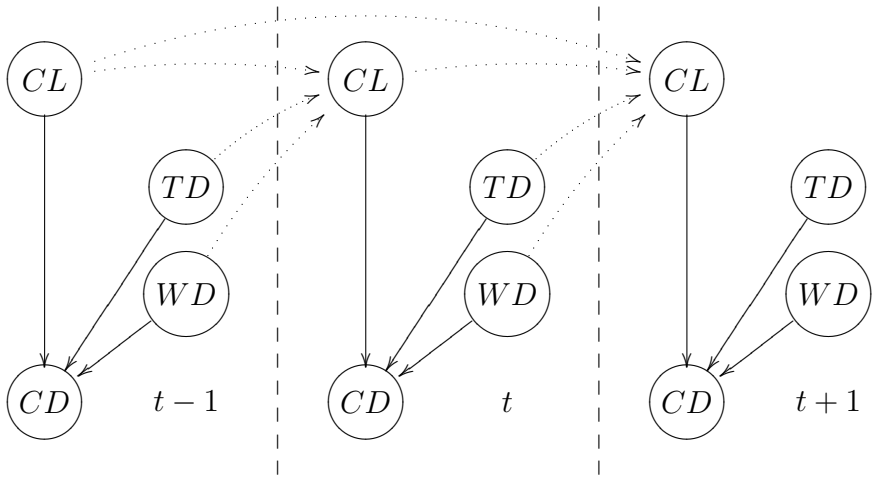


Abbildung 9.1: DBN für die Vorhersage des Ortes und der Dauer

muss zunächst analysiert werden, welche möglichen Variablen in dem vorgestellten Anwendungsszenario existieren. Die bekannten Daten zu jedem Zeitpunkt sind die Person, der Aufenthaltsort sowie der Zeitpunkt, zu dem der Ort betreten wurde. Da die Annahme getroffen wird, dass die Daten nicht zentral gesammelt werden, sondern jede Person ihre Daten aus Sicherheitsgründen selbst verwaltet, wird die Person nicht im Bayesschen Netz modelliert. Jede Person hat ihr eigenes Bayessches Netz und kann über einen Dienst die Vorhersagen anbieten, soweit sie dies wünscht. Die bekannten Daten einer Person sind die aufeinander folgenden Aufenthaltsorte sowie die Zeitpunkte, zu denen die Übergänge zwischen den Aufenthaltsorten stattfanden. Damit findet man als mögliche Variablen zunächst den aktuellen Aufenthaltsort (*current location CL*). Aus den zwei Zeitpunkten, wann ein Ort betreten wurde und wann der nächste Ort betreten wurde, kann die Verweildauer an diesem Ort berechnet werden. Die Verweildauer stellt somit eine weitere Variable (*current duration CD*) dar. Schließlich können aus dem Zeitpunkt eines Übergangs die Tageszeit (*time of day TD*) sowie der Wochentag (*weekday WD*) als Variablen ermittelt werden, mit denen eventuell vorhandene Zeitabhängigkeiten untersucht werden können. Diese Variablen bilden die Knoten eines Bayesschen Netzes, welches in jeder Zeitscheibe vorhanden ist.

Im nächste Schritt sollen nun die Abhängigkeiten zwischen den gefunden Variablen erarbeitet werden. Innerhalb jeder Zeitscheibe ist eine Abhängigkeit der aktuellen Verweildauer CD vom aktuellen Aufenthaltsort CL vorhanden. Weiterhin kann die Verweildauer CD von der Tageszeit TD und dem Wochentag WD des Übergangs in den aktuellen Aufenthaltsort abhängen. Abhängigkeiten über mehrere Zeitscheiben sind zunächst zwischen den aktuellen Aufenthaltsorten zu finden. Der aktuelle Aufenthaltsort CL_t zum Zeitpunkt t hängt von den aktuellen Aufenthaltsorten CL_{t-1} bis CL_{t-n} ab, wobei n der Ordnung im Falle der

Zustands- und Markov-Prädiktoren entspricht. Eine weitere Abhängigkeit kann zwischen dem aktuellen Aufenthaltsort CL_t und der Tageszeit TD_{t-1} sowie dem Wochentag WD_{t-1} des Betretens des letzten Aufenthaltsorts bestehen. Einen Überblick der eventuellen Einflüsse der Variablen aufeinander und die daraus gefundenen Abhängigkeiten gibt die Arbeit von Pietzowski [Pie04]. Abbildung 9.1 zeigt das beschriebene dynamische Bayessche Netz für drei Zeitscheiben.

9.1.2 Evaluierung

Da das Bayessche Netz für die Vorhersage des nächsten Aufenthaltsortes sowie für die Vorhersage der Verweildauer modelliert ist, werden im ersten Teil der Evaluierung die Vorhersagegenauigkeit und die Quantität der Vorhersagen des nächsten Aufenthaltsortes betrachtet. Im zweiten Teil wird dann die Vorhersagegenauigkeit und die Quantität der Vorhersage der Verweildauer untersucht.

Vorhersage des Aufenthaltsortes

Für die Vorhersage des nächsten Aufenthaltsortes wird die Quantität wie in den vorherigen Kapiteln verwendet. Wenn eine Folge von Orten das erste Mal abgelaufen wird, kann keine Vorhersage bezüglich des nächsten Ortes gemacht werden, da keine Wahrscheinlichkeiten für die möglichen nächsten Orte vorliegen. Damit entspricht auch die hier verwendete Quantität der Definition aus Abschnitt 5.5.1. Zur Vorhersage wird der Ort gewählt, für den das Bayessche Netz die höchste Wahrscheinlichkeit angibt. Dieser Ort wird mit dem tatsächlich betretenen Ort verglichen und die Vorhersage wird als korrekt oder falsch markiert. Die Vorhersagegenauigkeit wird dann wie in Abschnitt 5.5.1 berechnet. Für die Evaluierung ohne Training werden für den Vergleich gegenüber der Evaluierung mit Training die Vorhersagegenauigkeiten nur mit den Herbstdaten gemessen.

In der Evaluierung des Bayesschen Netzes wurden zum Einem die Abhängigkeit von der Historie, d.h. der Länge der Folge der zuletzt betretenen Orte gemessen. Zum Anderen wurde die Abhängigkeit von den Zeitparametern Tageszeit und Wochentag untersucht. Die Verwendung der Zeitparameter brachte aber keine Verbesserung gegenüber der Vorhersage ohne Zeitparameter. Im Gegensatz dazu hat die Historie einen Einfluss auf die Vorhersagegenauigkeit, wie Tabelle 9.1 zeigt.

Es ist keine optimale Länge der Folge der zuletzt betretenen Orte auszumachen. Für Person A wird das beste Resultat mit einer Historie von 3 erreicht. Für die anderen Personen liefert die Historie von 4 die höchste Genauigkeit. Die Quantität sinkt bei allen Testpersonen mit einer größeren Historie, da mit längeren Folgen von Orten auch mehr Muster existieren, die ein erstes Mal durchlaufen werden müssen.

Tabelle 9.1: Genauigkeit der Vorhersage des Ortes mittels DBN ohne Training

Historie		1	2	3	4	5
Person A	Genauigkeit	90,19%	89,47%	90,47%	87,27%	88,24%
	Quantität	91,15%	83,19%	59,29%	48,67%	19,47%
Person B	Genauigkeit	77,65%	80,17%	78,18%	81,25%	78,57%
	Quantität	94,83%	85,98%	64,94%	50,18%	24,35%
Person C	Genauigkeit	66,67%	67,45%	61,19%	72,83%	70,97%
	Quantität	95,88%	82,02%	59,18%	42,32%	18,73%
Person D	Genauigkeit	75,00%	75,84%	76,07%	76,82%	76,74%
	Quantität	95,43%	80,91%	61,83%	44,81%	27,80%

Um dem Problem der nicht möglichen Vorhersagen aufgrund von leeren Wahrscheinlichkeitstabellen entgegenzutreten, kann das Bayessche Netz vor der eigentlichen Nutzung trainiert werden. Für das Training werden die Sommerdaten verwendet. Die Vorhersagegenauigkeit wird dann mit den Herbstdaten der Augsburg Benchmarks gemessen. Der Einfluss eines Trainings auf die Genauigkeit der Vorhersage des nächsten Ortes wird in [Pie04, PPB+05] mit einer Historie von 2 untersucht. Tabelle 9.2 stellt die Ergebnisse dar, welche zeigen, dass mit einem Training eine Verbesserung der Vorhersagegenauigkeit sowie der Quantität erzielt werden kann.

Tabelle 9.2: Genauigkeit der Vorhersage des Ortes mittels DBN mit Training

	ohne Training		mit Training	
	Genauigkeit	Quantität	Genauigkeit	Quantität
Person A	89,47%	83,19%	89,65%	90,27%
Person B	80,17%	85,98%	83,72%	92,99%
Person C	67,45%	82,02%	72,14%	88,76%
Person D	75,84%	80,91%	76,49%	87,55%

Vorhersage der Verweildauer

Das zweite Ziel, welches mit der Modellierung des Bayesschen Netzes verfolgt wurde, ist die Vorhersage der Verweildauer am aktuellen Ort. Die Messwerte sind wieder die Vorhersagegenauigkeit und die Quantität. Auch hier wird auf Grund des Szenarios die statische Zuverlässigkeit eingesetzt, d.h. es wird keine Vorhersage der Verweildauer im eigenen Büro gemacht. Schließlich wird unter-

sucht, ob ein Training auch auf die Vorhersage der Verweildauer einen positiven Effekt hat.

Um die Vorhersagegenauigkeit zu verbessern, wurde der Einfluss der Zeitparameter Tageszeit und Wochentag getestet. Für die Wochentage werden die diskreten Werte *Montag*, *Dienstag*, *Mittwoch*, *Donnerstag*, *Freitag*, *Samstag* und *Sonntag* verwendet. Für eine Unterteilung des Tages müssen diskrete Werte gefunden werden. Wird der Tag in zu viele Abschnitte unterteilt, sind für einige dieser Intervalle zu wenige Daten vorhanden, um eine gute Vorhersage machen zu können. Deshalb wurde der Tag in folgende Abschnitte unterteilt: *morgens* (7 bis 11 Uhr), *mittags* (11 bis 14 Uhr), *nachmittags* (14 bis 18 Uhr) und *nachts* (18 bis 7 Uhr).

Eine Vorhersage der genauen Verweildauer ist, wie in Kapitel 8 erwähnt, nicht notwendig. Deshalb wird hier die folgende Unterteilung der Zeit verwendet (siehe Kapitel 8):

$$5, 10, 15, 20, 30, 45, 60, 120, > 120$$

Tabelle 9.3 zeigt inwieweit die Zeitparameter Tageszeit und Wochentag einen Einfluss auf die Vorhersagegenauigkeit haben.

Tabelle 9.3: Genauigkeit der Vorhersage der Dauer mittels DBN ohne Training

Parameter		keiner	Tageszeit	Wochentag	beide
Person A	Genauigkeit	77,67%	84,62%	71,25%	80,00%
	Quantität	91,15%	80,53%	70,80%	48,67%
Person B	Genauigkeit	86,38%	87,87%	83,48%	88,20%
	Quantität	94,83%	88,19%	82,66%	65,68%
Person C	Genauigkeit	83,59%	83,97%	83,56%	83,62%
	Quantität	95,88%	88,76%	84,27%	66,29%
Person D	Genauigkeit	68,70%	68,63%	61,42%	63,64%
	Quantität	95,44%	84,65%	81,74%	59,34%

Die Vorhersagegenauigkeit verbessert sich bei Verwendung des Parameters Tageszeit gegenüber keinem Parameter für die Personen A, B und C. Die Berücksichtigung des Wochentages bringt keine Verbesserungen für alle Personen. Die Kombination beider Parameter liefert aber wieder für die Personen A, B und C eine bessere Genauigkeit gegenüber keinem Zeitparameter. Der Grund für die Verschlechterung bei Verwendung des Wochentages kann die zu geringe Datenbasis sein, welche keine wöchentliche Struktur wiederspiegelt. Mit Hinzunahme von mehr Zeitparametern sinkt wie erwartet auch die Quantität.

Zur Untersuchung des Trainings des Bayesschen Netzes bei der Vorhersage der Verweildauer wurde das selbe Setup wie im Fall der Vorhersage des nächsten Aufenthaltsortes verwendet. Die Sommerdaten wurden als Trainingsdaten verwendet, mit den Herbstdaten wurden dann die Genauigkeit und die Quantität gemessen. Als Zeitparameter wurde die Tageszeit eingesetzt, da diese bei allen Personen die Vorhersagegenauigkeit verbesserte. Außer für Person B zeigt Tabelle 9.4 eine Verbesserung der Genauigkeit der Vorhersage der Verweildauer mit Training. Die Quantität ist für alle Personen höher, wenn das Bayessche Netz trainiert wurde.

Tabelle 9.4: Genauigkeit der Vorhersage der Dauer mittels DBN mit Training

	ohne Training		mit Training	
	Genauigkeit	Quantität	Genauigkeit	Quantität
Person A	84,62%	80,53%	85,58%	92,04%
Person B	87,87%	88,19%	86,54%	95,94%
Person C	83,97%	88,76%	86,77%	96,25%
Person D	68,63%	84,65%	69,78%	93,36%

Speicher- und Rechenaufwand

Das modellierte Bayessche Netz kann sich unter Umständen ähnlich der Markov-Prädiktoren auf ein bestimmtes Verhalten einlernen. Ändert der Benutzer sein Verhalten, benötigt das Netz unter Umständen sehr lange zum Umlernen. Dieses Phänomen kann mit einer Begrenzung der Anzahl der Datensätze gemildert werden, die für die Berechnung der Wahrscheinlichkeiten des Bayesschen Netzes verwendet werden. Die Anzahl der für die Berechnung der Wahrscheinlichkeiten genutzten Datensätze wird als interner Speicher bezeichnet. In [Pie04, PPB+05] wurde ein interner Speicher von 100 und 200 untersucht. Es wurde aber keine allgemein gültige Regel für die Bestimmung der Größe des internen Speichers gefunden, da dies von der Anwendung abhängt, in der das Bayessche Netz eingesetzt wird.

Für das Bayessche Netz jeder Person muss die Folge der letzten r Ortswechsel gespeichert werden, wobei r die Größe des internen Speichers ist. Für jeden Ortswechsel muss wiederum eine ID des Ortes, die Tageszeit, der Wochentag und die Verweildauer gespeichert werden. In den verwendeten Daten gab es 15 verschiedene Räume, welche mit 4 Bit unterschieden werden können. Für die Tageszeit werden 2 Bit und für den Wochentag 3 Bit benötigt. Für die Verweildauer wurden neun diskrete Werte festgelegt, welche mit 4 Bit zu speichern sind. Somit sind die Speicherkosten C_{bayes}, um die Daten für das Bayessche Netz einer Person

abzulegen, wie folgt zusammengesetzt:

$$\begin{aligned} C_{bayes} &= r \cdot (C_{ort} + C_{tageszeit} + C_{wochentag} + C_{dauer}) \\ &= r \cdot (4\,Bit + 2\,Bit + 3\,Bit + 4\,Bit) \\ &= r \cdot 13\,Bit \end{aligned}$$

Bei einem internen Speicher von 500 Datensätzen errechnen sich somit die folgenden Speicherkosten

$$C_{bayes} = 500 \cdot 13 = 6500\,Bit$$

Der Rechenaufwand des Bayesschen Netzes wird durch die Berechnung der bedingten Wahrscheinlichkeiten mittels der vorgestellten Kettenregel bestimmt, welche aber bei größeren Netzen ineffizient ist. Das hier vorgestellte dynamische Bayessche Netz wurde in Java realisiert und auf zwei verschiedenen Systemen getestet, zum Einen auf einem PC mit 2,4 GHz und einem Arbeitsspeicher von 1 GB, zum Anderen auf einem PDA mit 400 MHz Taktrate und einem Speicher von 64 MB. Auf dem PC wurde eine durchschnittliche Zeit von $4,01\,ms$ für eine Berechnung benötigt. Die Berechnungszeit auf dem PDA lag bei durchschnittlich $786,28\,ms$.

9.2 Multi-Layer-Perzeptron

Künstliche Neuronale Netze (KNN) oder kurz Neuronale Netze (NN) sind eine Möglichkeit, bestimmte Fähigkeit des menschlichen Gehirns auf Computersysteme zu übertragen. Sie bestehen aus einer großen Anzahl einfacher, parallel arbeitender Einheiten, den Neuronen [Gal93]. Diese sind nach dem Vorbild der Neuronen in einem Nervensystem miteinander über Kommunikationskanäle verbunden, um gemeinsam komplexe Aufgaben zu lösen. Künstliche Neuronale Netze zeichnen sich insbesondere durch ihre Lernfähigkeit, Fehlertoleranz, Generalisierungs- und Assoziationsfähigkeit und die Möglichkeit paralleler Verarbeitung aus [Zel94]. Neuronale Netz werden heute in vielen Bereichen eingesetzt – Mustererkennung, Klassifikation, Funktionsapproximation und Sprachanalyse sind nur einige wichtige Anwendungsfelder.

Aus der Vielfalt von Neuronalen Netzen wurde zunächst das Multi-Layer-Perzeptron (MLP), der wohl bekannteste Vertreter, zur Lösung der Vorhersage des nächsten Aufenthaltsortes gewählt.

9.2.1 Modellierung

Das Multi-Layer-Perzeptron besteht aus einer Eingabeschicht, mehreren internen Schichten und einer Ausgabeschicht. Für die Problemstellung wurde das einfachste Multi-Layer-Perzeptron mit einer internen Schicht ausgewählt (siehe Abbildung 9.2). Als Lernverfahren wird ein modifizierter *Back-Propagation-Algorithmus* verwendet. Die Aufenthaltsorte werden binär kodiert, um Rechenaufwand zu sparen, was von großem Interesse für mobile Geräte ist, die bestimmten Energie- und Echtzeitanforderungen unterliegen. Mit einer Bit-Kodierung wird eine Komplexität von $\log_2 n$ (Neuronen der Eingabeschicht) erreicht, wobei n die Anzahl der vorhandenen Orte ist. Eine 1:1-Kodierung (ein Neuron pro Ort) erreicht nur eine Komplexität von n. Diese Wahl der Kodierung bringt Vorteile vor allem bei einer großen Anzahl von Aufenthaltsorten. Mehr Details bezüglich der Kodierung sind in [VGPU04b] zu finden.

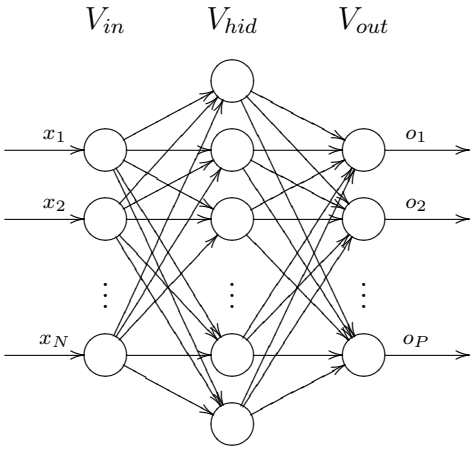


Abbildung 9.2: Struktur Multi-Layer-Perzeptron

Auch im Fall der Neuronalen Netze wird das modellierte Multi-Layer-Perzeptron auf dem persönlichen Gerät jedes Nutzers ausgeführt, so dass der Nutzer selbst nicht modelliert werden muss. Die Eingabedaten bestehen somit nur aus den zu letzt besuchten Räumen. Die Anzahl der zu letzt besuchten Räume (Historie) ist ein Parameter des Netzes und entspricht der Ordnung der Zustands- und Markov-Prädiktoren. Für die Augsburg Benchmarks mit 15 Räumen genügen 4 Bits für die Kodierung. Werden jetzt die vier zuletzt betretenen Räume betrachtet, ist

$$V_{in} = 0101\ 0010\ 0001\ 0011$$

ein möglicher Eingabevektor. Die Anzahl der Neuronen in der internen Schicht können von der Anzahl der Eingabeneuronen abhängen. Diese Abhängigkeit wurde schon in anderen Arbeiten festgestellt [ESQ+03]. Das Multi-Layer-Perzeptron

gibt über seine Ausgabezellen den vorhergesagten Raum binär kodiert zurück. Im konkreten Fall mit 15 Räumen besteht die Ausgabeschicht aus 4 Neuronen ($P = 4$),

$$V_{out} = 0010$$

ist dann ein möglicher Ausgabevektor.

Für den Lernprozess wurde ein modifizierter Back-Propagation-Algorithmus [VGPU04a, VGPU04b] verwendet. Für bessere Ergebnisse werden im Lernverfahren die Binärwerte -1 und 1 statt 0 und 1 verwendet. Ein Neuronales Netz muss vor seinem Einsatz mit schon vorhandenen Bewegungsmustern trainiert werden. Dafür dienen wie vorab schon erwähnt die Sommerdaten der Augsburg Benchmarks. Ein wichtiger Parameter für das Training ist der Schwellenwert t. Ist der ermittelte Fehler kleiner als der Schwellenwert, werden die Ausgabewerte akzeptiert. Andernfalls wird eine Rückwärts-Propagierung durchgeführt bis die Bedingung erfüllt ist. Auch die Lernrate α ist ein wichtiger Parameter für den Trainingsprozess, welcher für die Anpassung der Gewichte benötigt wird.

Während des Einsatzes des Netzes soll dieses mit den tatsächlich eingetretenen Ereignissen weiter trainiert werden. Die Vorhersage wird basierend auf einem Vorwärtsschritt vorgenommen. Das bedeutet, wenn der Ausgabewert eines Neurons der Ausgabeschicht im Intervall $[-1; 0)$ liegt, wird er als -1 betrachtet, wenn er im Intervall $[0; 1]$ liegt, wird er als 1 betrachtet. Falls die Vorhersage korrekt war, wird eine Rückwärts-Propagierung durchgeführt. Falls die Vorhersage falsch ist, werden normalerweise so viele Rückwärts-Propagierungen durchgeführt bis der gewünschte Wert ermittelt wird. Dies stellt aber Probleme mit Echtzeitanforderungen dar. Eine realistischere Lösung, welche sich auch gut für mobile Geräte wie PDAs eignet, wird in [VGPU04a, VGPU04b] vorgeschlagen. Falls eine Vorhersage falsch war, wird nur eine Rückwärts-Propagierung durchgeführt. Somit wird das Netz schneller angepasst und ist für eine folgende Anfrage schneller wieder bereit.

9.2.2 Optimale Parameter

In einer Simulation mit den Augsburg Benchmarks wurden für die folgenden Parameter des Neuronalen Netzes optimale Werte bestimmt: die Anzahl der Eingabeneuronen N, die von der Historie H abhängt, d.h. der Länge der verwendeten Räume, die Anzahl der internen Neuronen M, der Schwellenwert t für das Training vor dem Einsatz und die Lernrate α. Weiterhin sollte die beste Anzahl an Rückwärtsschritten B für den Lernprozess im eigentlichen Einsatz gefunden werden. Alle diese Parameter wurden variiert, um die optimale Konfiguration des Neuronalen Netzes zu erhalten [VGPU04a]. Tabelle 9.5 zeigt die untersuchten und

Tabelle 9.5: Untersuchte und optimale Werte der Parameter des MLP

Parameter	untersuchte Werte	optimaler Wert
Netzstruktur		
Historie	[1;6]	2
Anzahl interne Neuronen	{5;7;...;15}	9
Lernverfahren		
Schwellenwert	{0,1;0,3;...;0,9}	0,1
Lernrate	[0,05;0,30]	0,10
Anzahl Rückwärtsschritte	[1;5] und unbegrenzt	1

optimalen Werte der Parameter, die in der Simulation gefunden wurden. Als beste Anzahl von Rückwärtsschritten wurde $B = 1$ gefunden. Dieses Ergebnis bestärkt den Vorschlag für den Lernprozess im Einsatz nur eine Rückwärts-Propagierung durchzuführen.

9.2.3 Evaluierung

Mit den ermittelten optimalen Parametern wurden die in Tabelle 9.6 gezeigten Vorhersagegenauigkeiten erreicht. Das Training vor dem eigentlichen Einsatz des Netzes erzielte bei Person B und C einen positiven Effekt. Bei Person A bleibt die Genauigkeit mit Training unverändert gegenüber dem Einsatz ohne Training. Bei Person D verschlechtert sich aber die Vorhersagegenauigkeit mit Training. Die Quantität ist in allen Fällen annähernd 100%, da das Multi-Layer-Perzeptron immer einen Ausgabevektor produziert, der bis auf einen Code einen Raum darstellt.

Tabelle 9.6: Genauigkeit der Vorhersage des Ortes mittels MLP

	ohne Training	mit Training
A	87.39%	87.39%
B	75.28%	75.66%
C	63.40%	68.68%
D	77.82%	74.06%

Speicher- und Rechenaufwand

Die Speicherkosten für das Multi-Layer-Perzeptron können wie folgt berechnet werden. Mit einer verwendeten Raumfolge von 2 ($H = 2$) und 15 Räumen, welche mit 4 Bit kodiert werden können, ergibt sich für die Anzahl der Eingabeneuronen

$$N = 4 \cdot H = 8$$

für die Anzahl der Neuronen der internen Schicht

$$M = N + 1 = 4 \cdot H + 1 = 9$$

sowie für die Anzahl der Ausgabeneuronen

$$P = 4$$

Die Eingabeneuronen können als Binärwerte gespeichert werden, wogegen die Neuronen der internen Schicht sowie die Ausgabeneuronen als Fließkommazahlen gespeichert werden müssen. Weiterhin müssen die Gewichte auf den Kanten zwischen den Neuronen der verschiedenen Schichten gespeichert werden. Die Anzahl der Gewichte ist

$$W = N \cdot M + M \cdot P = M \cdot (N + P) = 108$$

Diese müssen ebenfalls als Fließkommazahlen gespeichert werden. Betrachtet man einen 32-Bit-Rechner ergibt sich für die Speicherkosten des Multi-Layer-Perzeptron

$$C_{multi} = 32 \cdot (M \cdot (N + P) + P + M) + N = 32 \cdot 121 + 8 = 3880\,Bit$$

Für das Training des Multi-Layer-Perzeptron wurden zwischen 10 und 45 Sekunden auf einem Rechner mit 650 MHz und 128 MB Hauptspeicher benötigt. Der Rechenaufwand in der dynamischen Phase, d.h. während des tatsächlichen Einsatzes, ist vernachlässigbar. Dies liegt auch daran, dass die neuen Gewichte im modifizierten Back-Propagation-Algorithmus in nur einem Rückwärtsschritt berechnet werden. Das Netz kann somit auch in mobilen Geräten mit begrenzter Rechenleistung eingesetzt werden.

9.3 Elman-Netz

Das Elman-Netz enthält gegenüber dem Multi-Layer-Perzeptron weitere spezielle interne Zellen, die Kontextzellen. Durch diese wird ein Speichermechanismus im Netz realisiert, da diese die Aktivierungszustände der internen Zellen sichern. So entsteht eine Abhängigkeit zwischen zwei Propagierungen innerhalb des jeweiligen Netzes, da den internen Zellen nicht nur die Eingabeinformation, sondern auch die Informationen der Kontextzellen über gewichtete Verbindungen zur weiteren Berechnung übergeben werden [Kuh05]. Diese Netze sind insbesondere für die Erkennung und Klassifizierung von Mustern geeignet, die sich im zeitlichen Verlauf ändern. Außerdem sind sie zur Prognose von Zeitreihen verwendbar [Moz93].

9.3.1 Modellierung

Die Struktur eines Elman-Netzes ist in Abbildung 9.3 dargestellt. Das Elman-Netz erweitert das Multi-Layer-Perzeptron um eine weitere interne Schicht mit den so genannten Kontextzellen. Die Anzahl der Kontextzellen ist gleich der Anzahl der Neuronen der internen Schicht. Mit Hilfe der Kontextzellen werden die Zustände der internen Schicht gespeichert. Deshalb sind die internen Zellen eins zu eins über Rückkopplungen mit den Kontextzellen verbunden, so dass ihre Zustände als Eingabe dienen. Die Gewichte dieser Rückkopplungsverbindungen sind alle gleich 1 und bleiben unverändert. Sie werden somit nicht trainiert. Die Kontextzellen dienen dann den internen Zellen neben den Eingabezellen als weitere Eingabe in der nächsten Propagierung.

Die Person wird wiederum nicht im Elman-Netz modelliert, da das Netz auf dem eigenen Gerät jeder Person laufen soll. Beim Elman-Netz wurde im Gegensatz zum Multi-Layer-Perzeptron eine 1:1-Kodierung der Räume gewählt, d.h. es werden genau so viele Eingabeneuronen wie Räume multipliziert mit der Länge der Raumfolge benötigt. Die Größe des Netzes verhält sich in diesem Fall linear zur Anzahl der Räume. Die gleiche Kodierung wurde auch für die Ausgabeneuronen gewählt, d.h. die Anzahl der Ausgabeneuronen entspricht der Anzahl der Räume.

Als Lernverfahren wird auch hier der *Back-Propagation-Algorithmus* [RHW86] eingesetzt. Für detaillierte Information bezüglich des Algorithmus sei auf [Kuh05] verwiesen. Da der Ausgabevektor kontinuierliche Werte liefert und die kodierten Räume aber nur mit den diskreten Werten 0 und 1 vergleichbar sind, müssen die ermittelten kontinuierlichen in diskrete Werte umgewandelt werden. Für die 1:1-Kodierung wird dabei der größte Wert aller Ausgabeneuronen gewählt und als 1 interpretiert, alle anderen Werte sind als 0 zu interpretieren.

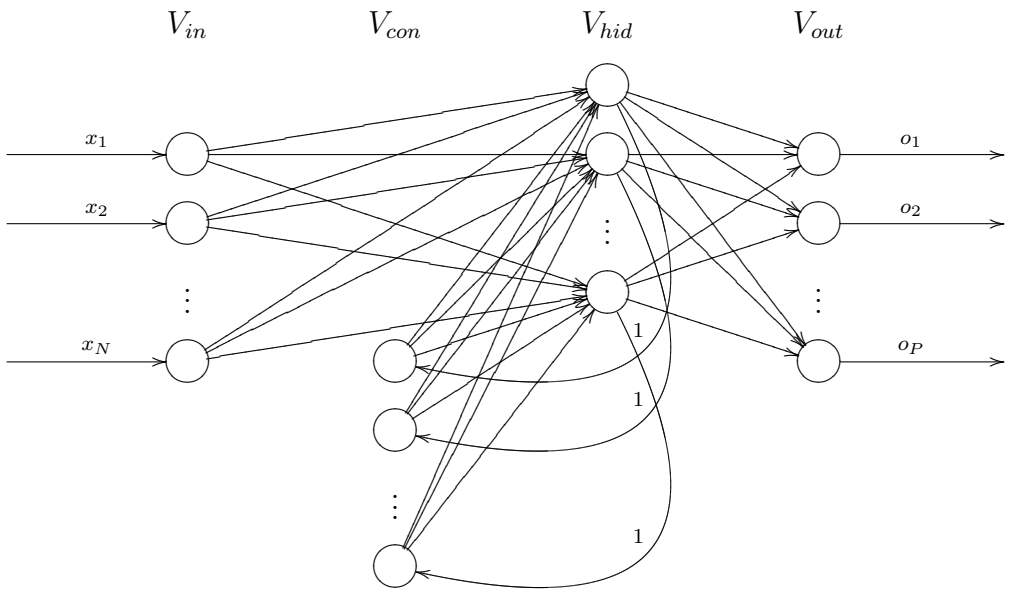


Abbildung 9.3: Struktur Elman-Netz

9.3.2 Optimale Parameter

In mehreren Simulationen mit den Augsburg Benchmarks wurde nach den optimalen Parametern des modellierten Elman-Netzes gesucht. Tabelle 9.7 zeigt alle untersuchten Parameter aufgeteilt nach Parametern für die Netzstruktur und Parametern für das Lernverfahren.

Tabelle 9.7: Untersuchte und optimale Werte der Parameter des Elman-Netzes

Parameter	untersuchte Werte	optimaler Wert
Netzstruktur		
Kodierung	binär, 1:1	1:1
Anzahl interne Zellen	[5;20]	5
Historie	[1;5]	1
Lernverfahren		
Initialisierung	zufällig, fest	fest
Aktivierungsfunktion	$\tanh(x)$, $\frac{1}{1+\exp(-x)}$	$\tanh(x)$
Lernzyklen	[5;150]	31
Lernrate η	[0,1;0,7]	0,1
Momentum α	[0,00;0,05;...;0,95]	0,00

Die Analyse der beiden Kodierungen binär und 1:1 zeigte, dass die 1:1-Kodierung die optimale Wahl ist. Mit der binären Kodierungen wurden extrem schlechte

Ergebnisse erreicht, was an der zusätzlichen Auflösung der Kodierung durch das Netz liegen kann. Für die Anzahl der internen Zellen wurde für das Elman-Netz keine Abhängigkeit von der Anzahl der Eingabezellen gefunden. Als bester Wert ist 5 geeignet. Mit einer höheren Anzahl wurden keine besseren Ergebnisse erzielt. Der Speicherbedarf wächst aber mit einer größeren Anzahl interner Zellen. Auch bei der Historie, welche die Länge der Sequenz der vergangenen Räume angibt, wurde mit größeren Werten keine Verbesserung erzielt. Somit wurde als Historie 1 gewählt, da längere Raumsequenzen außerdem die Anzahl der Eingabezellen erhöhen.

Für die Initialisierung der Gewichte werden im Normalfall zufällige Werte aus einem sehr kleinen Intervall gewählt. Mit den zufälligen Werten wurden aber über mehrere Testläufe stark schwankende Ergebnisse ermittelt. Aus diesem Grund wurde für jede Person eine feste Initialisierung gewählt, die sich in mehreren Testläufen als beste erwies. Als Aktivierungsfunktionen wurden die Funktionen Tangens hyperbolicus $\tanh(x)$ und die sigmoide Funktion $\frac{1}{1+\exp(-x)}$ untersucht. Unter beiden Funktionen stellte sich keine als beste heraus. Aufgrund der Erfahrungen, welche in [Zel94] erwähnt sind, wurde der Tangens hyperbolicus $\tanh(x)$ gewählt. Die Anzahl der Lernzyklen gibt an, wie oft die Sommerdaten der Augsburg Benchmarks trainiert wurden. Hierbei stellten sich 31 Lernzyklen als optimal heraus. Für Lernrate η wurde untersucht, wie sich eine feste Lernrate gegenüber einer Lernrate verhält, die im Verlaufe des Trainings dekrementiert wird. Dabei wurde die dekrementierende Lernrate 0,1 als optimale Lernrate gefunden. Das Momentum ist ein optionaler Parameter des Back-Propagation-Algorithmus, welches angibt inwieweit die zuletzt vorgenommene Adaption eines Gewichtes die aktuelle Anpassung beeinflusst. In den Simulationen wurde festgestellt, dass ohne Einfluss der letzten Adaption auf die aktuelle Anpassung bessere Werte zu erreichen sind.

9.3.3 Evaluierung

Tabelle 9.8 zeigt die Vorhersagegenauigkeit mit und ohne Training, welche mit den optimalen Parametern erreicht wurden. Als Trainingsdaten wurden die Sommerdaten verwendet. Die Messungen wurden dann in beiden Fällen mit den Herbstdaten durchgeführt. Die Quantität beträgt 100%, da das Elman-Netz aufgrund der 1:1-Kodierung der Ausgabeneuronen immer eine Vorhersage machen kann. Für alle Personen hat wie erwartet das Training zu einer Erhöhung der Vorhersagegenauigkeit geführt.

Tabelle 9.8: Genauigkeit der Vorhersage des Ortes mittels Elman-Netz

	ohne Training	mit Training
Person A	83,03%	91,07%
Person B	68,88%	78,88%
Person C	60,52%	69,92%
Person D	71,36%	78,83%

Rechen- und Speicheraufwand

Die Speicherkosten setzen sich aus den Kosten für die Zellzustände, den Kosten für die Schwellenwerte der Zellen und den Kosten für die Gewichte zusammen. Mit 15 betrachteten Räumen und den optimalen Parametern ergibt sich für die Anzahl der Eingabezellen

$$N = 15 \cdot H = 15 \cdot 1 = 15$$

für die Anzahl der internen Zellen und Kontextzellen

$$M = 5 \text{ und } K = M = 5$$

für die Anzahl der Ausgabezellen

$$P = 15$$

wobei die Zustände der Eingabezellen binär und die Zustände der Zellen der anderen Schichten als Fließkommazahlen gespeichert werden. Die Schwellenwerte müssen für die Zellen der internen, der Kontext- sowie der Ausgabeschicht als Fließkommazahlen gespeichert werden. Die Anzahl der zu speichernden Schwellenwerte ist somit

$$T = M + K + P = 5 + 5 + 15 = 25$$

Weiterhin müssen die Gewichte als Fließkommazahlen gespeichert werden. Die Anzahl der zu speichernden Gewichte beträgt

$$W = N \cdot M + K \cdot M + M \cdot P = M \cdot (N + K + P) = 5 \cdot (15 + 5 + 15) = 175$$

Somit ergibt sich für die Speicherkosten des Elman-Netzes C_{elman} auf einem 32-Bit-Rechner

$$\begin{aligned} C_{elman} &= 32 \cdot (M + K + P + T + W) + N \\ &= 32 \cdot (5 + 5 + 15 + 25 + 175) + 15 \\ &= 7215\, Bit \end{aligned}$$

Der Rechenaufwand für den Trainingsprozess ist mit unter $500\, ms$ pro Lernzyklus akzeptabel. Die Zeit für eine Vorhersage des nächsten Raumes beträgt auf einem Standard-PC (Taktrate 1,67 GHz, 512 MB Arbeitsspeicher) weniger als eine Millisekunde. Auf einem mobilen PDA (Taktrate 400 MHz, 60 MB Arbeitsspeicher) wurde eine durchschnittliche Berechnungszeit von $13\, ms$ ermittelt.

9.4 Vergleich

Den bisher in diesem Kapitel vorgestellten Verfahren sollen jetzt die Zustands- und Markov-Prädiktoren gegenüber gestellt werden. Dabei soll der Vergleich anhand der folgenden Kriterien stattfinden:

- Modellierungsaufwand
- Stabilität
- Vorhersagegenauigkeit
- Quantität
- Trainingsverhalten
- An- und Umlernverhalten
- Speicher- und Rechenaufwand
- Erweiterbarkeit
- Möglichkeit der Vorhersage der Zeit

Da die Zustands- und Markov-Prädiktoren bisher ohne Training betrachtet wurden, werden zunächst beide Prädiktoren wie zu Beginn des Kapitels beschrieben trainiert sowie die damit erzielten Werte den untrainierten Prädiktoren gegenübergestellt. Dabei werden der globale zweistufige Two-State-Prädiktor sowie der globale Markov-Prädiktor mit Ordnung 2 herangezogen, da im Fall des Multi-Layer-Perzeptron sowie des Bayesschen Netzes eine Historie von 2 gewählt wurde.

Zusätzlich werden die beiden Prädiktoren unter Verwendung eines Zuverlässigkeitszählers evaluiert (siehe Kapitel 6), um eine erste Optimierungsmöglichkeit einfließen zu lassen. Der verwendete Zuverlässigkeitszähler hat vier Zustände ($\{00, 01, 10, 11\}$) und $k = 10$ dient als Barriere.

Tabelle 9.9: Genauigkeit der Vorhersage des Ortes mittels globaler Zustands- und Markov-Prädiktoren mit Training

			Markov-Prädiktor		Zustandsprädiktor	
			ohne Zuv.	mit Zuv.	ohne Zuv.	mit Zuv.
Person A	ohne Training	Genauigkeit	89,47%	89,25%	87,37%	88,89%
		Quantität	84,82%	83,04%	84,82%	80,36%
	mit Training	Genauigkeit	88,24%	88,30%	87,25%	88,17%
		Quantität	91,07%	83,93%	91,07%	83,04%
Person B	ohne Training	Genauigkeit	79,91%	82,35%	75,21%	82,01%
		Quantität	86,67%	75,56%	86,67%	70,00%
	mit Training	Genauigkeit	78,97%	83,80%	76,19%	82,30%
		Quantität	93,33%	80,00%	93,33%	77,41%
Person C	ohne Training	Genauigkeit	70,00%	70,37%	63,64%	71,18%
		Quantität	82,71%	71,05%	82,71%	63,91%
	mit Training	Genauigkeit	70,46%	70,73%	65,40%	71,81%
		Quantität	89,10%	77,07%	89,10%	70,68%
Person D	ohne Training	Genauigkeit	75,25%	81,01%	70,20%	81,51%
		Quantität	82,16%	65,56%	82,16%	60,58%
	mit Training	Genauigkeit	76,53%	81,14%	70,89%	81,88%
		Quantität	88,38%	72,61%	88,38%	66,39%

Tabelle 9.9 zeigt die Vorhersagegenauigkeiten sowie die Quantität im untrainierten und trainierten Fall, die mit diesen Prädiktoren erzielt wurden. Weiterhin wurde für alle Prädiktoren die statische Zuverlässigkeit betrachtet. Dafür wurde das eigene Büro in die Menge der unzuverlässigen Kontexte aufgenommen, d.h. aus dem eigenen Büro werden keine Vorhersagen gemacht. Ohne Training und ohne Zuverlässigkeitszähler liefert der Zustandsprädiktor mit Ordnung 2 signifikant schlechtere Ergebnisse als der Markov-Prädiktor mit Ordnung 2 bei erwartungsgemäß gleicher Quantität. Wird der Zuverlässigkeitszähler eingebunden, sind die erzielten Ergebnisse beider Prädiktoren annähernd gleich. Für Person C und D ist der Zustandsprädiktor sogar besser als der Markov-Prädiktor. Die Quantität der Zustandsprädiktoren liegt aber immer unter der der Markov-Prädiktoren. Das Training bewirkt in Bezug auf die Vorhersagegenauigkeit meist kleine Verbesse-

rungen, wogegen die Quantität in allen Fällen deutlich ansteigt. Bei Person A fällt auf, dass in allen Fällen die Vorhersagegenauigkeit mit Training leicht sinkt. Der Grund ist ein unterschiedliches Verhalten von Person A im Sommer und Herbst. Ohne das Training mit den Sommerdaten sind die Muster noch nicht angelernt und werden in die Berechnung der Genauigkeit nicht einbezogen. Mit den Sommerdaten sind diese Muster nun angelernt und werden als falsche Vorhersagen in die Berechnung einbezogen.

Neben diesen vier Prädiktoren sollen auch die Hybridprädiktoren mit Zustands- und Markov-Prädiktoren in der Prädiktormenge zum Vergleich herangezogen werden. Als Hybridverfahren wird der Mehrheitsprädiktor mit einfacher Mehrheit (siehe Kapitel 7) verwendet. Die Mehrheitsprädiktoren mit einfacher Mehrheit lieferten Vorhersagegenauigkeiten, die über dem Durchschnitt der Vorhersagegenauigkeiten der Prädiktoren der Prädiktormenge lagen. Weiterhin erreichten sie eine akzeptable Quantität.

Im Folgenden werden fünf verschiedene Prädiktormengen gewählt, die für einen besseren Vergleich die Speicherkosten nicht unnötig in die Höhe treiben sollen, d.h. als maximale Zahl für die Ordnung wurde 3 gewählt. Eine nicht zu große Ordnung verhindert auch eine zu geringe Quantität. Desweiteren kombinieren die Mengen lokale mit globalen Prädiktoren sowie Zustands- mit Markov-Prädiktoren. Tabelle 9.10 zeigt die untersuchten Mehrheitsprädiktoren mit der Prädiktormenge.

Tabelle 9.10: Verwendete Mehrheitsprädiktoren mit einfacher Mehrheit

Mehrheitsprädiktor	Prädiktormenge
Hybrid-Zustandsprädiktor (global) $H1$	globaler Two-State-Prädiktor (Ordnung 1, 2, 3)
Hybrid-Markov-Prädiktor (global) $H2$	globaler Markov-Prädiktoren (Ordnung 1, 2, 3)
Hybrid-Zustandsprädiktor $H3$	lokaler Two-State-Prädiktor (Ordnung 1, 2, 3) globaler Two-State-Prädiktor (Ordnung 1, 2, 3)
Hybrid-Markov-Prädiktor $H4$	lokaler Markov-Prädiktor (Ordnung 1, 2, 3) globaler Markov-Prädiktor (Ordnung 1, 2, 3)
Hybrid-Zustands-Markov-Prädiktor $H5$	lokaler Two-State-Prädiktor (Ordnung 1, 2, 3) globaler Two-State-Prädiktor (Ordnung 1, 2, 3) lokaler Markov-Prädiktor (Ordnung 1, 2, 3) globaler Markov-Prädiktor (Ordnung 1, 2, 3)

Um die gewählten Hybridprädiktoren mit den anderen Verfahren vergleichen zu können, müssen diese auch bezüglich ihres Verhalten im untrainierten und trainierten Fall betrachtet werden. Tabelle 9.11 zeigt die Vorhersagegenauigkeit der

Tabelle 9.11: Genauigkeit der Vorhersage des Ortes mittels Hybridprädiktoren

Hybridprädiktor			$H1$	$H2$	$H3$	$H4$	$H5$
Person A	ohne Training	Genauigkeit	87,37%	89,47%	92,41%	92,41%	92,50%
		Quantität	84,82%	84,82%	70,54%	70,54%	71,43%
	mit Training	Genauigkeit	88,12%	89,11%	91,01%	91,21%	91,30%
		Quantität	90,18%	90,18%	79,46%	81,25%	82,14%
Person B	ohne Training	Genauigkeit	78,41%	80,43%	80,77%	80,21%	79,58%
		Quantität	84,07%	85,19%	67,41%	69,26%	70,74%
	mit Training	Genauigkeit	78,84%	81,15%	80,66%	79,82%	79,11%
		Quantität	89,26%	90,37%	78,52%	84,44%	83,33%
Person C	ohne Training	Genauigkeit	67,49%	70,97%	68,15%	70,69%	71,35%
		Quantität	76,32%	81,58%	59,02%	65,41%	64,29%
	mit Training	Genauigkeit	69,41%	71,67%	69,40%	69,12%	70,05%
		Quantität	82,33%	87,59%	68,80%	76,69%	74,06%
Person D	ohne Training	Genauigkeit	73,68%	76,41%	82,91%	81,33%	82,72%
		Quantität	78,84%	80,91%	65,56%	68,88%	67,22%
	mit Training	Genauigkeit	74,15%	77,62%	81,03%	80,11%	81,22%
		Quantität	85,06%	87,14%	72,20%	77,18%	75,10%

fünf Hybridprädiktoren ohne und mit Training. Zum Vergleich wurde auch hier die statische Zuverlässigkeit mit dem eigenen Büro der jeweiligen Person in der Menge der unzuverlässigen Kontexte gewählt.

Die Ergebnisse zeigen zunächst, dass ein Training die Genauigkeit der Vorhersage nicht unbedingt verbessert. Nur in 8 von den aufgezeigten 20 Messreihen verbessert das Training die Vorhersagegenauigkeit. Dies kann zum Beispiel daran liegen, dass eine Gewohnheitsänderung zwischen Sommer- und Herbstdaten stattgefunden hat. Einen Vorteil bringt das Training für die Quantität, die in allen Fällen mit Training ansteigt. Bei einem Vergleich der Hybridprädiktoren untereinander, zeigen bei Person A und D die Hybridprädiktoren $H3$, $H4$ und $H5$ ein besseres Verhalten, was auf den Vorteil der Kombination von lokalen und globalen Prädiktoren zurückzuführen sein kann. Bei Person B und C kann nur für den untrainierten Fall festgestellt werden, dass einer dieser drei Hybridprädiktoren das beste Ergebnis liefert. Trotzdem sollen zum Vergleich mit den anderen Verfahren diese drei Hybridprädiktoren – der Hybrid-Zustandsprädiktor, der Hybrid-Markov-Prädiktor sowie der Hybrid-Zustands-Markov-Prädiktor – verwendet werden.

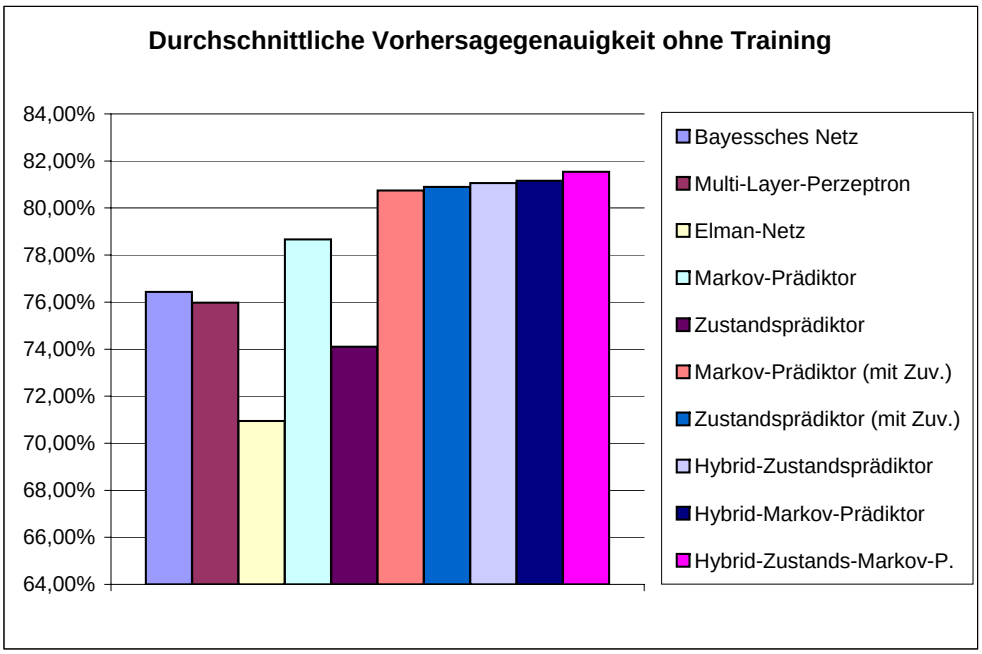


Abbildung 9.4: Vergleich der Vorhersagegenauigkeit ohne Training

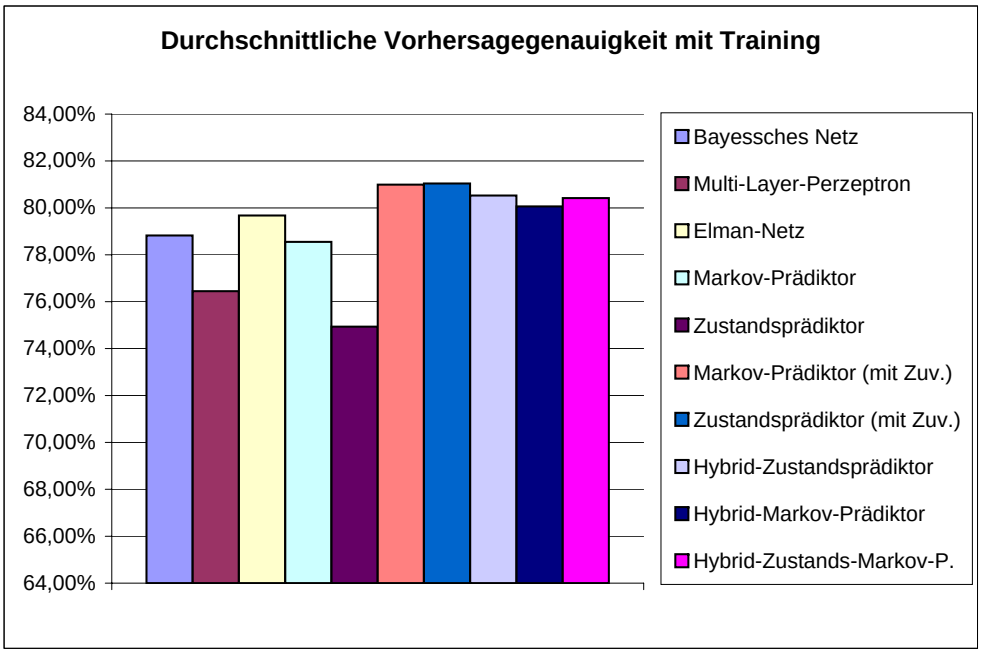


Abbildung 9.5: Vergleich der Vorhersagegenauigkeit mit Training

Die Zustands- und Markov-Prädiktoren ohne und mit Zuverlässigkeitszähler sowie die Hybridprädiktoren sollen nun mit den Bayesschen Netzen, dem Multi-Layer-Perzeptron und dem Elman-Netz bezüglich Modellierungsaufwand, Sta-

bilität, Vorhersagegenauigkeit, Quantität, Trainingsverhalten, An- und Umlernverhalten, Speicher- und Rechenaufwand, Erweiterbarkeit und Möglichkeit der Vorhersage der Zeit verglichen werden. Zunächst soll ein Vergleich bezüglich des dominierenden Kriteriums, der Vorhersagegenauigkeit, vorgenommen werden. Die Abbildungen 9.4 und 9.5 zeigen zunächst die durchschnittliche Genauigkeit der Vorhersage des nächsten Aufenthaltsortes ohne und mit Training aller Verfahren. Der Durchschnitt wurde hierbei über die vier Testpersonen gebildet.

Vergleicht man zunächst die Zustands- und Markov-Prädiktoren ohne Zuverlässigkeit und Hybridansatz mit den anderen Verfahren (die fünf linken Balken), liefert der Markov-Prädiktor ohne Training durchschnittlich das beste Ergebnis. Der Zustandsprädiktor ist nur besser als das schlechteste Verfahren, das Elman-Netz. Mit Training hat das Elman-Netz die größte Vorhersagegenauigkeit, wogegen der Zustandsprädiktor die kleinste Vorhersagegenauigkeit liefert. Bezieht man jetzt die Zuverlässigkeit sowie den Hybridansatz mit ein, ist festzustellen, dass das beste Resultat ohne und mit Training unter diesen fünf Verfahren zu finden ist. Die Vorhersagegenauigkeiten dieser fünf Verfahren liegen ohne und mit Training innerhalb nur eines Prozentpunktes. Ohne Training erreicht der Hybrid-Zustands-Markov-Prädiktor die höchste Vorhersagegenauigkeit, mit Training wird das beste Ergebnis durch den Zustandsprädiktor mit Zuverlässigkeitszähler erzielt.

Im Folgenden sollen nicht nur die Vorhersagegenauigkeiten sondern auch der Aufwand der Verfahren anhand der vorab schon erwähnten Kriterien bewertet werden.

Modellierungsaufwand. Bei der Modellierung des Bayesschen Netzes müssen mögliche Abhängigkeiten der verwendeten Variablen aus den vorliegenden Daten gefunden und modelliert werden. Beide Neuronalen Netze erfordern einen großen Aufwand für die Modellierung, welcher durch das Suchen nach den optimalen Parametern für die Netzstruktur und das Lernverfahren hervorgerufen wird. Der Modellierungsaufwand für die Zustands- und Markov-Prädiktoren ist dagegen sehr gering. Eine Entscheidung muss nur über die Größe der Ordnung getroffen werden. Bei Verwendung eines Zuverlässigkeitszählers muss die Entscheidung getroffen werden, wie viele Zustände der Zähler haben soll und welcher Zustand als Barriere dienen soll. Bei den Hybridprädiktoren muss nur vorab bestimmt werden, welche Prädiktoren in die Prädiktormenge aufgenommen werden.

Stabilität. Mit der Stabilität soll angegeben werden, wie stark der Einfluss einer Änderung der Parameter ist. Beim Bayesschen Netz geben die Historie sowie Zeitparameter eine Möglichkeit die Vorhersagegenauigkeit zu optimieren. Für das Multi-Layer-Perzeptron und das Elman-Netz gibt es eine ganze Reihe von Parametern, die die Vorhersagegenauigkeit beeinflussen.

Das Elman-Netz stellt sich deshalb auch als Verfahren mit der geringsten Stabilität heraus. Bei den Zustands- und Markov-Prädiktoren kann nur die Ordnung variiert werden. Die Vorhersagegenauigkeit der Hybridprädiktoren wird durch die Prädiktoren in der Prädiktormenge bestimmt. Da die Prädiktormenge für einem bestimmten Hybridprädiktor fest ist, kann diese nicht variiert werden, somit verhalten sich die Hybridprädiktoren am stabilsten. Abbildung 9.6 zeigt die Stabilität aller Verfahren ohne Training. Dabei ist für jedes Verfahren die durchschnittliche Vorhersagegenauigkeit, das Minimum und das Maximum der erreichten Vorhersagegenauigkeiten mit allen Parameterkombinationen und unter allen Testpersonen dargestellt. Tabelle 9.12 gibt für die Stabilität die Differenz zwischen Maximum und Minimum an.

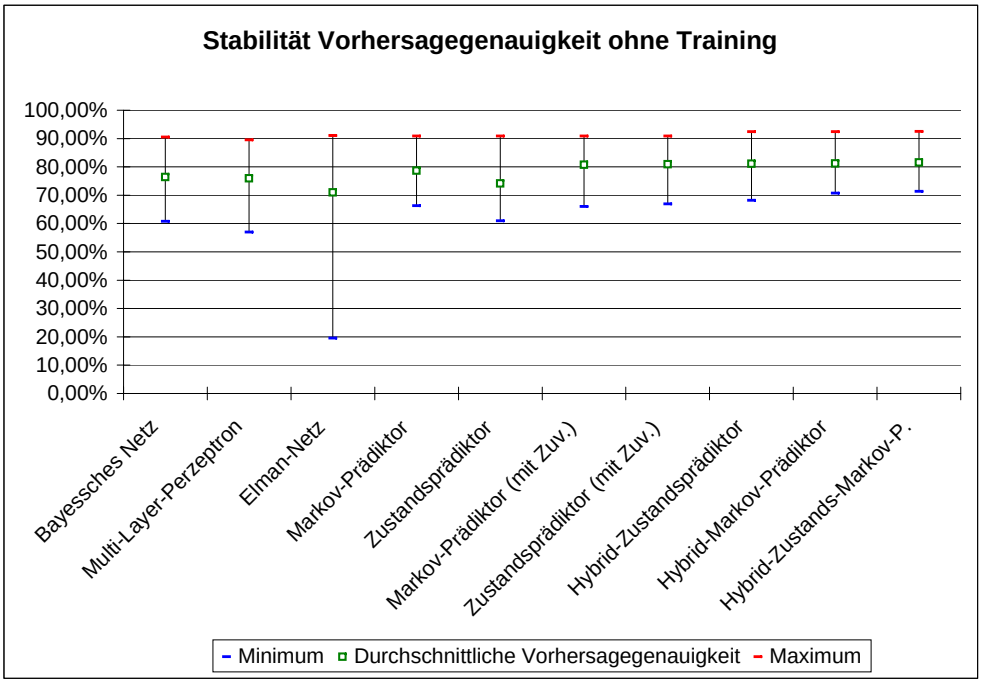


Abbildung 9.6: Stabilität der Vorhersagegenauigkeit ohne Training

Vorhersagegenauigkeit und Training. Betrachtet man die durchschnittliche Vorhersagegenauigkeit der vorgestellten Verfahren (siehe Tabelle 9.12), zeigen die Prädiktoren mit Zuverlässigkeitszähler und die drei Hybridprädiktoren ohne und mit Training das beste Verhalten. Die drei Hybridprädiktoren liefern mit Training ein schlechteres Ergebnis als ohne Training. Hingegen wird bei den anderen Verfahren mit Training eine höhere Genauigkeit erreicht. Beim Elman-Netz wird mit Training sogar einen Steigerung von neun Prozentpunkten verzeichnet.

Quantität. Das Elman-Netz erreicht eine Quantität von 100%, da immer ein

Ausgabevektor berechnet wird, der einen Kontext darstellt. Das Multi-Layer-Perzeptron hat fast eine Quantität von 100%, da nicht jeder Code für einen Kontext steht. Bayessche Netze, Zustands- und Markov-Prädiktor erreichen in etwa die gleiche Quantität. Sobald eine Optimierung wie Zuverlässigkeitszähler bzw. Hybridprädiktoren verwendet werden, sinkt die Quantität. Bei allen Verfahren steigt die Quantität erwartungsgemäß mit Training.

Anlernverhalten. Das Anlernverhalten der beiden Neuronalen Netze kann aufgrund des unausweichlichen Trainings vor dem eigentlichen Einsatz als sehr langsam angesehen werden. Das Bayessche Netz, die Zustands- und Markov-Prädiktoren ohne und mit Zuverlässigkeitszähler sowie die Hybridprädiktoren hingegen können beim zweiten Auftreten eines Muster eine Vorhersage erzeugen.

Umlernverhalten. Die beiden Neuronalen Netze, der Markov-Prädiktor ohne und mit Zuverlässigkeitszähler sowie der Hybrid-Markov-Prädiktor benötigen viel Zeit zum Umlernen einer Gewohnheit. Beim Bayesschen Netz ist das Umlernen auch langsam, wird aber durch die Verwendung des internen Speichers beschleunigt. Der Zustandsprädiktor ohne und mit Zuverlässigkeitszähler sowie der Hybrid-Zustandsprädiktor erlernen nach zwei Änderungen die neue Gewohnheit. Die Kombination aus Zustands- und Markov-Prädiktor liegt zwischen beiden Prädiktortypen.

Speicherkosten. Die Speicherkosten der beiden Neuronalen Netze sind sehr gering und unabhängig von der Anzahl der Ortswechsel einer Person. Das Bayessche Netz benötigt ebenfalls wenig Speicher, dieser ist aber abhängig von der Anzahl der Ortswechsel, deren zu speichernde Menge aber wiederum durch den internen Speicher begrenzt werden kann. Der Zustandsprädiktor hat die geringsten Speicherkosten, wogegen der Markov-Prädiktor durch die Speicherung aller Häufigkeiten die größten Speicherkosten unter den Verfahren ohne Optimierungen verursacht. Die Verwendung eines Zuverlässigkeitszähler erhöht die Speicherkosten unwesentlich. Die Hybridprädiktoren verbrauchen den meisten Speicher, da dieser durch die Verwendung mehrerer Ordnungen bzw. Verfahren die Summe der Speicherkosten aller verwendeten Prädiktoren ist. Die Kosten des Hybrid-Zustandsprädiktor sind dabei noch akzeptabel. Sobald die Markov-Prädiktoren verwendet werden, steigen die Kosten enorm. Tabelle 9.12 zeigt die Speicherkosten bei Verwendung einer oberen Grenze von 500 Ortswechsel. Damit sind die aufgelisteten Speicherkosten auch abhängig von der Anzahl der Ortswechsel.

Rechenaufwand. Der Rechenaufwand für das Training der Neuronalen Netze ist sehr hoch, da das Elman-Netz mittels mehrerer Lernzyklen und das Multi-Layer-Perzeptron, bis der Gesamtfehler kleiner als der Schwellenwert

ist, trainiert werden. Der Rechenaufwand der Neuronalen Netze im Einsatz ist dagegen gering, da bei beiden Netzen nur eine Rückwärts-Propagierung durchgeführt wird. Für das Bayessche Netz wird eine Berechnung der Wahrscheinlichkeiten mittels der Kettenregel durchgeführt, welche nicht effizient ist. Bei den Zustands- und Markov-Prädiktoren ohne und mit Zuverlässigkeitszähler besteht der Rechenaufwand aus der Suche des aktuellen Musters in einer Tabelle, bei den Hybridprädiktoren wird das Muster in mehreren Tabellen gesucht.

Erweiterbarkeit. Die Erweiterung um zusätzliche Kontexte, zum Beispiel bei Verwendung der Nokia Kontext Daten mit 125 Zell-IDs anstatt der Augsburg Benchmarks mit 15 Räumen, ist bei den Neuronalen Netzen nur mit einer erneuten Modellierung und einer damit verbundenen Suche nach den optimalen Netzparametern möglich. Das bedeutet die hier modellierten Netze sind nicht ohne Aufwand in anderen Anwendungen einsetzbar. Für das Bayessche Netz, die Zustands- und Markov-Prädiktoren sowie die Hybridprädiktoren ist solch eine Erweiterung um zusätzliche Kontexte, wie schon die Evaluierung mit den Nokia-Kontext-Daten gezeigt hat, ohne Aufwand möglich.

Vorhersage der Verweildauer. Die Vorhersage der Verweildauer ist direkt im Bayesschen Netz mit eventuellen Abhängigkeiten modelliert. Für die anderen Verfahren kann nur die parallele Modellierung verwendet werden.

Tabelle 9.12 zeigt eine Zusammenfassung der Vergleichskriterien aller untersuchten Verfahren. Beim Vergleich der Kriterien kann zunächst festgestellt werden, dass die Neuronalen Netze trotz ihrer hohen Quantität eher als ungeeignet erscheinen, da sie einen hohen Modellierungsaufwand haben, nicht ohne Aufwand um zusätzliche Kontexte erweiterbar sind und die niedrigste Stabilität aufweisen. Das Bayessche Netz sticht aufgrund der integrierten Modellierung der Vorhersage der Verweildauer hervor. Die Hybridprädiktoren haben einen großen Nachteil bei den Speicherkosten, welcher sich aber beim Hybrid-Zustandsprädiktor in Grenzen hält. Die Zustandsprädiktoren sind anhand der Kriterien die beste Wahl. Durch Verwendung des Zuverlässigkeitszählers wird auch eine sehr gute Vorhersagegenauigkeit erreicht. Gegenüber den Markov-Prädiktoren zeichnen sie sich durch schnelles Umlernen und den wesentlich geringeren Speicherkosten aus.

Tabelle 9.12: Vergleich aller Verfahren

	DBN	MLP	Elman-Netz	MP	ZP	MP mit Zuv.	ZP mit Zuv.	Hybrid-ZP	Hybrid-MP	Hybrid-ZP-MP
Modellierungsaufw.	mittel	hoch	hoch	gering	gering	gering	gering	gering	gering	gering
Stabilität (%)	29,67	32,59	71,57	24,67	29,97	24,95	23,99	24,25	21,72	21,15
$\bar{t}$ ($\bar{q}$) ohne Training (%)	76,43 (83,03)	75,97 (≈ 100)	70,95 (100)	78,66 (84,09)	74,11 (84,09)	80,75 (73,80)	80,90 (68,71)	81,06 (65,63)	81,16 (68,52)	81,54 (68,42)
$\bar{t}$ ($\bar{q}$) mit Training (%)	78,82 (89,89)	76,45 (≈ 100)	79,68 (100)	76,53 (90,47)	70,89 (90,47)	81,14 (78,40)	81,88 (74,38)	80,53 (74,74)	80,06 (79,89)	80,42 (78,66)
Anlernen	schnell	langsam	langsam	schnell	schnell	schnell	schnell	schnell	schnell	schnell
Umlernen	langsam	langsam	langsam	langsam	schnell	langsam	schnell	schnell	langsam	mittel
Speicher (Bit)	6.500	3.880	7.215	36.960	2.730	37.380	3.150	27.486	334.904	362.390
Rechenaufwand	Berechnung mittels Kettenregel	Training bis $E < t$, sonst eine Propagierung	Training über mehrere Lernzyklen, sonst eine Propagierung	Suche in einer Tabelle	Suche in einer Tabelle	Suche in einer Tabelle	Suche in einer Tabelle	Suche in sechs Tabellen	Suche in sechs Tabellen	Suche in zwölf Tabellen
Erweiterbarkeit	ja	nein	nein	ja	ja	ja	ja	ja	ja	ja
Vorhersage Dauer	integriert	parallel	parallel	parallel	parallel	parallel	parallel	parallel	parallel	parallel

10 Zusammenfassung

Die vorliegende Arbeit zeigt mit den Zustandsprädiktoren eine Möglichkeit zur Kontextvorhersage basierend auf der Kontexthistorie. Die Zustandsprädiktoren sind durch verschiedene Sprungvorhersagetechniken motiviert und mit den Markov-Prädiktoren verwandt. Um die Kontextvorhersage zu untersuchen, wurde zunächst der Begriff Kontext geeignet definiert. Anschließend wurde die Einordnung eines Kontextes in primäre und sekundäre Kategorien vorgestellt. Aus den vier primären Kategorien Ort, Identität, Zeit und Aktivität können alle weiteren Kontexte (sekundäre) hergeleitet werden. Im Blick auf die Kontextvorhersage wurde sich dann nur auf diese primären Kategorien beschränkt.

Um die entwickelten Verfahren zu bewerten, wurden die *Augsburg Indoor Location Tracking Benchmarks* und die *Nokia Context Daten* verwendet. Die Augsburg Benchmarks beinhalten Bewegungsmuster, welche im Rahmen der *Smart-Doorplates* entstanden. Diese Applikation diente gleichzeitig als erste Testumgebung für die Kontextvorhersage des nächsten Aufenthaltsortes. Die Nokia Context Daten enthielten neben zwei Ortsinformation auch die Aktivität eines Benutzers, die in der Evaluierungsphase vorhergesagt wurde. Somit konnten mit diesen Benchmarks drei der vier primären Kategorien abgedeckt werden, da auch die Zeit bei beiden Benchmarks enthalten ist. Eine Vorhersage der Identität erweist sich auch als nicht sinnvoll, da zudem vorgeschlagen wurde, die Speicherung der Kontexthistorie sowie die Vorhersage daraus für jede Person separat auf einem persönlichen Gerät durchzuführen, um so die Privatsphäre der Person zu schützen.

Mit der umgesetzten Grundidee der Zustandsprädiktoren konnten in der Evaluierung noch nicht die erhofften guten Resultate erzielt werden. Die Idee der lokalen Muster, welche sich aus den Nachfolgekontexten eines Kontextes zusammensetzen, brachte aber erste sehr gute Ergebnisse. Im Vergleich der Zustands- mit den Markov-Prädiktoren zeigte sich, dass das unter Umständen langsame Umlernen der Markov-Prädiktoren mit den Zustandsprädiktoren wie erwartet nicht mehr auftritt.

Die Kontextvorhersage soll zunächst einen Mehrwert in einem ubiquitären System bieten. Dabei wird es nicht zwingend erforderlich sein, immer eine Vorhersage anzubieten. Wenn ermittelt werden kann, dass eine Vorhersage nicht zuverlässig ist, kann diese unterbunden werden. Unter diesem Gesichtspunkt wurden verschiede-

ne Zuverlässigkeitsverfahren vorgestellt, mit denen eine Steigerung der Vorhersagegenauigkeit der Zustandsprädiktoren erreicht wurde.

Eine weitere Möglichkeit zur Verbesserung der Zustandsprädiktoren stellten die Hybridprädiktoren dar. Hier wurden mehrere Zustands- sowie Markov-Prädiktoren zu einem neuen Prädiktor zusammengefasst, der aus den Ergebnissen der Einzelprädiktoren das Gesamtergebnis mittels verschiedener Techniken wählt. Mit den Hybridprädiktoren wurden Vorhersagegenauigkeiten erreicht, die besser als die Ergebnisse der Einzelprädiktoren oder zumindest über dem Durchschnitt der Einzelprädiktoren lagen. Mit der Kombination der Zustands- und Markov-Prädiktoren sowie globalen und lokalen Mustern wurde zudem schon gezeigt, dass ein Zusammenspiel verschiedener Techniken die Vorteile der einzelnen Techniken hervortreten lassen kann.

Bei der Vorhersage des nächsten Kontextes ist eine Angabe zu welchem Zeitpunkt dieser Übergang stattfindet für den Benutzer unerlässlich. Deshalb wurden zwei Varianten der Vorhersage der Verweildauer im aktuellen Kontext aufgezeigt. Für eine Integration der Zeitvorhersage in die Zustandsprädiktoren wurde eine Idee vorgestellt, die sich aber schon aus theoretischer Sicht als nicht praktikabel erwies. Deshalb wurde eine parallele Zeitvorhersage mit Hilfe des Mittelwerts bzw. Medians der vorangegangenen Verweilzeiten im aktuellen Kontext vorgeschlagen. Diese Modellierung brachte Ergebnisse, die sich als akzeptabel herausstellten. Desweiteren kann die parallele Modellierung auch in anderen Verfahren eingesetzt werden, bei denen eine Integration der Zeit auch nur erschwert möglich ist, wie zum Beispiel dem Multi-Layer-Perzeptron und dem Elman-Netz.

Beim abschließenden Vergleich der Zustandsprädiktoren mit anderen Verfahren zur Vorhersage des nächsten Kontextes wurden ein Bayessches Netz, ein Multi-Layer-Perzeptron und ein Elman-Netz betrachtet. Auf Seiten der Zustands- und Markov-Prädiktoren wurden Hybridprädiktoren verwendet. Das Fazit dieses Vergleichs ist, dass es kein optimales Verfahren zur Kontextvorhersage gibt. Jedes Verfahren zeigt bei anderen Kriterien Nachteile bzw. Schwächen. Die beiden Neuronalen Netze benötigen einen enorm hohen Modellierungsaufwand und sind nicht erweiterbar um zusätzliche Kontexte. Das Bayesschen Netz sticht zwar hervor mit einer integrierten Zeitvorhersage, kann sich aber unter Umständen auf eine Gewohnheit einlernen, so dass ein Umlernen viel Zeit in Anspruch nimmt. Bis auf die erhöhten Speicherkosten, welche durch die Verwendung mehrerer Prädiktoren gleichzeitig zustande kommen, konnte der Hybridprädiktor mit lokalen und globalen Zustandsprädiktoren in der Prädiktormenge bezüglich der meisten Kriterien überzeugen.

Mit den Zustandsprädiktoren wurde ein Verfahren gezeigt, das sich sehr gut zur Kontextvorhersage basierend auf der Kontexthistorie eignet. Erweiterungen um Zuverlässigkeitsabschätzung steigern die Vorhersagegenauigkeit. Mit der Kombinationen mehrerer Variationen in Hybridprädiktoren bleibt die Suche nach der

passenden Ordnung erspart.

Für eine Weiterentwicklung der Zustandsprädiktoren gibt es noch verschiedene Ansatzpunkte. Zum Einen ist zu untersuchen, wie zusätzliche Informationen integriert werden können, die die vorherzusagenden Kontexte beeinflussen. Zum Beispiel können Abhängigkeiten von Tageszeiten, Wochentagen, Monaten usw., wie sie bei den Bayesschen Netzen schon modelliert wurden, eingebaut werden, um einzubeziehen, dass zu gewissen Zeitpunkten bestimmte Kontexte wahrscheinlicher eintreten können als andere. Das dafür benötigte Wissen kann aus den Benchmarks extrahiert werden. Es können weiterhin zusätzliche Informationen von außerhalb wie zum Beispiel die sicheren Daten aus einem elektronischen Terminkalender einfließen. Die zweistufigen Prädiktoren könnten dafür so abgeändert werden, dass in der Musterverlaufstabelle nicht nur die Kontexthistorie sondern ein Systemzustand gespeichert wird. Dieser Systemzustand kann dann zum Beispiel einen sicheren Eintrag des Terminkalenders, die Tageszeit sowie den aktuellen Aufenthaltsort enthalten. Zu jedem dieser Systemzustände wird dann ein Two-State-Prädiktor gespeichert, der den nächsten Systemzustand vorhersagt. Zum Anderen ist die Integration der Vorhersage der Verweildauer bzw. des Eintritts in den nächsten Kontext eine Herausforderung.

Nicht nur die Zustandsprädiktoren auch die vorgeschlagenen Verfahren zur Zuverlässigkeitsabschätzung sowie die Hybridprädiktoren sind neue Ideen, die anderen Verfahren eine Optimierungsmöglichkeit geben. Der Zuverlässigkeitszähler wurde zum Beispiel schon erfolgreich mit dem von Lucian Vintan entwickelten Multi-Layer-Perzeptron [VGPU04a, VGPU04b] eingesetzt. Es wurden dabei zwar zusätzliche Speicherkosten verursacht, aber die Vorhersagegenauigkeit konnte gesteigert werden. Weiterhin können in den Hybridprädiktoren, wie durch die Kombination der Zustands- und Markov-Prädiktoren gezeigt, auch andere Verfahren zur Vorhersage in den Prädiktormengen verwendet werden. Durch solch eine Kombination können die Vorteile verschiedener Verfahren genutzt werden. Eine zusätzliche Verwendung des Bayesschen Netzes mit den Zustands- und Markov-Prädiktoren könnte eine weitere Verbesserung bringen. Aber auch ein Hybridprädiktor, der nur die Bayesschen Netze mit unterschiedlicher Historie oder unterschiedlichen Zeitparametern verwendet, ist denkbar.

In der vorliegenden Arbeit wurde davon ausgegangen, dass die Eingabe der Prädiktoren, d.h. die Kontexthistorie, korrekt war. Sollen die Prädiktoren auf der Ebene von *high-level*-Kontexten arbeiten, ist diese Voraussetzung nicht mehr gegeben, da diese Kontexte meist selbst nur mit einer bestimmten Wahrscheinlichkeit aus Sensordaten gewonnen werden. Aus diesem Grund könnten Untersuchungen darüber durchgeführt werden, wie die Prädiktoren mit so genannten unscharfen bzw. verrauschten Eingabedaten bzw. Kontexten umgehen können. Ein weiterer Schritt in diese Richtung wäre dann die Kombination der Kontextvorhersage mit der Kontexterkennung. Bei dieser Kombination kann zum Beispiel

auch die Kontextvorhersage zum Prüfen der Korrektheit der Kontexterkennung verwendet werden.

Eine Betrachtung von Kosten, die bei einer falschen Vorhersage entstehen können, wurde bisher noch nicht vorgenommen. Kosten können dabei zum Beispiel für das Zurücksetzen unnötig ausgeführter Aktionen entstehen. In einer Weiterentwicklung sollten diese Kosten einbezogen werden.

Literaturverzeichnis

[AS03] Ashbrook, Daniel und Thad Starner: *Using GPS to learn significant locations and predict movement across multiple users.* Personal and Ubiquitous Computing, 7(5):275–286, Oktober 2003.

[BBC97] Brown, Peter J., John D. Bovey und Xian Chen: *Context-aware Applications: from the Laboratory to the Marketplace.* IEEE Personal Communications, 4(5):58–64, Oktober 1997.

[BD02] Bhattacharya, Amiya und Sajal K. Das: *LeZi-Update: An Information-Theoretic Framework for Personal Mobility Tracking in PCS Networks.* Wireless Networks, 8(2/3):121–135, März 2002.

[Beh99] Behrends, Ehrhard: *Introduction to Marcov Chains.* Vieweg, 1999.

[Bei02] Beigl, Michael: *Context Aware Computing.* In: *Tutorial at ARCS 2002*, Karlsruhe, Deutschland, April 2002.

[Bra97] Brants, Thorsten: *Implementierung statistischer Methoden in der Sprachverarbeitung.* Computerlinguistik, Universität des Saarlandes, Saarbrücken, 1997. Projektseminar.

[Bro96] Brown, Peter J.: *The Stick-e Document: a Framework for Creating Context-aware Applications.* In: *Proceedings of Electronic Publishing '96*, Seiten 259–272, Palo Alto, USA, Januar 1996.

[BU02] Brinkschulte, Uwe und Theo Ungerer: *Mikrocontroller und Mikroprozessoren.* Springer-Verlag, 2002.

[CCM96] Chen, I-Cheng K., John T. Coffey und Trevor N. Mudge: *Analysis of Branch Prediction via Data Compression.* In: *ASPLOS VII*, Seiten 128–137, Cambridge, USA, Oktober 1996.

[CMP00] Clarkson, Brian, Kenji Mase und Alex Pentland: *Recognizing user context via wearable sensors.* In: *Proceedings of the Fourth International Symposium on Wearable Computers (ISWC 2000)*, Seiten 69–76, Atlanta, USA, Oktober 2000.

[CP98] Clarkson, Brian und Alex Pentland: *Extracting context from environmental audio.* In: *Proceedings of the Second International Symposium on Wearable Computers*, Seiten 154–155, Montréal, Kanada, Mai 1998.

[DA00] Dey, Anind K. und Gregory D. Abowd: *Towards a Better Understanding of Context and Context-Awareness.* In: *Workshop on The What, Who, Where, When, and How of Context-Awareness, Conference on Human Factors in Computer Systems (CHI2000)*, Den Haag, Niederlande, April 2000.

[DAW98] Dey, Anind K., Gregory D. Abowd und Andy Wood: *CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services.* Knowledge-Based Systems, 11(1):3–13, September 1998.

[Dey98] Dey, Anind K.: *Context-Aware Computing: The CyberDesk Project.* In: *Proceedings of AAAI '98 Spring Symposium on Intelligent Environments (AAAI '98 IE)*, Seiten 51–54, Palo Alto, USA, März 1998.

[DH93] Dagum, Paul und Eric Horvitz: *A Bayesian Analysis of Simulation Algorithms for Inference in Belief Networks.* Technischer Bericht KSL-91-67, Stanford University, September 1993.

[DRD+00] Dix, Alan, Tom Rodden, Nigel Davies, Jonathan Trevor, Adrian Friday und Kevin Palfreyman: *Exploiting space and location as a design framework for interactive mobile systems.* ACM Transaction on Computer-Human Interaction (TOCHI), 7(3):285–321, September 2000.

[ESQ+03] Egan, Colin, Gordon Steven, Patrick Quick, Rubén Anguera und Lucian Vintan: *Two-Level Branch Prediction using Neural Networks.* Journal of Systems Architecture, 49(12-15):557–570, Dezember 2003.

[FF98] Franklin, David und Joshua Flachsbart: *All gadget and no representation makes Jack a dull environment.* In: *Proceedings of AAAI 1998 Spring Symposium on Intelligent Environments*, Seiten 155–160, Palo Alto, USA, März 1998.

[Fla04] Flanagan, Adrian: *Nokia Context Data. Context Database, Institute of Pervasive Computing, University of Linz, Austria.* http://www.soft.uni-linz.ac.at/Research/Context_Database/index.php, September 2004.

[Gal93] Gallant, Stephen I.: *Neural Networks and Expert Systems.* MIT Press, 1993.

[GKMP98] Grunwald, Dirk, Artur Klauser, Srilatha Manne und Andrew Pleszkun: *Confidence Estimation for Speculation Control.* In: *Proceedings of the 29th Annual International Symposium on Computer Architecture*, Seiten 122–131, Barcelona, Spanien, Juni/Juli 1998.

[Glo02] Globke, Wolfgang: *Markovketten und verdeckte Markovmodelle.* Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, 2002. Proseminar Bioinformatik.

[GSB02] Gellersen, Hans-W., Albrecht Schmidt und Michael Beigl: *Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts.* Mobile Networks and Applications, 7(5):341–351, Oktober 2002.

[HNBR97] Hull, Richard, Philip Neaves und James Bedford-Roberts: *Towards Situated Computing.* In: *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, Seite 146, Cambridge, USA, Oktober 1997.

[Jen96] Jensen, Finn V.: *An Introduction to Bayesian Networks.* UCL Press, 1996.

[JRS96] Jacobsen, Erik, Eric Rotenberg und James E. Smith: *Assigning Confidence to Conditional Branch Predictions.* In: *Proceedings of the 29th Annual International Symposium on Microarchitecture*, Seiten 142–152, Paris, Frankreich, Dezember 1996.

[KKP+03] Korpipää, Panu, Miika Koskinen, Johannes Peltola, Satu-Marja Mäkelä und Tapio Seppänen: *Bayesian approach to sensor-based context awareness.* Personal and Ubiquitous Computing, 7(2):113–124, Juli 2003.

[KLH02] Kaowthumrong, Khomkrit, John Lebsack und Richard Han: *Automated Selection of the Active Device in Interactive Multi-Device Smart Spaces.* In: *Workshop at UbiComp'02: Supporting Spontaneous Interaction in Ubiquitous Computing Settings*, Göteborg, Schweden, September/Oktober 2002.

[Kuh05] Kuhlmann, Mirjam: *Untersuchung von Neuronalen Netzen zur Kontextvorhersage in ubiquitären Systemen.* Diplomarbeit, Institut für Informatik, Universität Augsburg, Februar 2005.

[LAL01] Laerhoven, Kristof van, Kofi A. Aidoo und Steven Lowette: *Real-time Analysis of Data from Many Sensors with Neural Networks.* In: *Proceedings of the International Symposium on Wearable Computers (ISWC 2001)*, Zürich, Schweiz, Oktober 2001.

[LC00] Laerhoven, Kristof van und Ozan Cakmakci: *What Shall We Teach Our Pants?* In: *Proceedings of the 4th International Symposium on Wearable Computers (ISWC 2000)*, Seiten 77–83, Atlanta, USA, Oktober 2000.

[MHH03] Mäntyjärvi, Jani, Johan Himberg und Pertti Huuskonen: *Collaborative Context Recognition for Handheld Devices.* In: *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, Seiten 161–168, Dallas-Fort Worth, USA, März 2003.

[Moz93] Mozer, Michael C.: *Neural Net Architectures for Temporal Sequence Processing.* In: *Time Series Prediction: Forecasting the Future and Understanding the Past*, Seiten 243–264, Redwood City, USA, Dezember 1993.

[Moz98] Mozer, Michael C.: *The Neural Network House: An Environment that Adapts to its Inhabitants.* In: *AAAI Spring Symposium on Intelligent Environments*, Seiten 110–114, Menlo Park, USA, März 1998.

[MRF03] Mayrhofer, Rene, Harald Radi und Alois Ferscha: *Recognizing and Predicting Context by Learning from User Behavior.* In: *The International Conference on Advances in Mobile Multimedia*, Seiten 25–35, Jakarta, Indonesien, September 2003.

[Mur02] Murphy, Kevin Patrick: *Dynamic Bayesian Networks: Representation, Inference and Learning.* Doktorarbeit, University of California, Berkeley, 2002.

[Pas98] Pascoe, Jason: *Adding Generic Contextual Capabilities to Wearable Computers.* In: *The Second International Symposium on Wearable Computers*, Seiten 92–99, Pittsburgh, USA, Oktober 1998.

[PBT+04] Petzold, Jan, Faruk Bagci, Wolfgang Trumler, Theo Ungerer und Lucian Vintan: *Global State Context Prediction Techniques Applied to a Smart Office Building.* In: *The Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, USA, Januar 2004.

[PBTU03a] Petzold, Jan, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *Context Prediction Based on Branch Prediction Methods.*

Technischer Bericht 2003-14, Institut für Informatik, Universität Augsburg, Juli 2003.

[PBTU03b] Petzold, Jan, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *Global and Local Context Prediction.* In: *Artificial Intelligence in Mobile Systems 2003 (AIMS 2003)*, Seattle, USA, Oktober 2003.

[PBTU03c] Petzold, Jan, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *The State Predictor Method for Context Prediction.* In: *Adjunct Proceedings Fifth International Conference on Ubiquitous Computing*, Seattle, USA, Oktober 2003.

[PBTU04] Petzold, Jan, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *Confidence Estimation of the State Predictor Method.* In: *2nd European Symposium on Ambient Intelligence*, Seiten 375–386, Eindhoven, Niederlande, November 2004.

[Pet03] Petzold, Jan: *Einsatz von Sprungvorhersagetechniken zur Kontextvorhersage in ubiquitären Systemen.* Technischer Bericht 2003-4, Institut für Informatik, Universität Augsburg, März 2003.

[Pet04] Petzold, Jan: *Augsburg Indoor Location Tracking Benchmarks.* Technischer Bericht 2004-9, Institut für Informatik, Universität Augsburg, Februar 2004.

[Pie04] Pietzowski, Andreas: *Kontextvorhersage in ubiquitären Systemen mit Bayesschen Netzen.* Diplomarbeit, Institut für Informatik, Universität Augsburg, Oktober 2004.

[PLFK03] Patterson, Donald J., Lin Liao, Dieter Fox und Henry Kautz: *Inferring High-Level Behavior from Low-Level Sensors.* In: *5th International Conference on Ubiquitous Computing*, Seiten 73–89, Seattle, USA, Oktober 2003.

[PPB+05] Petzold, Jan, Andreas Pietzowski, Faruk Bagci, Wolfgang Trumler und Theo Ungerer: *Prediction of Indoor Movements Using Bayesian Networks.* In: *First International Workshop on Location- and Context-Awareness (LoCA 2005)*, Seiten 211–222, Oberpfaffenhofen, Deutschland, Mai 2005.

[PSR92] Pan, Shien-Tai, Kimming So und Joseph T. Rahmeh: *Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation.* In: *Proceedings of the fifth international conference on Architectural support for programming languages and operating systems (ASPLOS V)*, Seiten 76–84, Boston, USA, April 1992.

[RCDD98] Rodden, Tom, Keith Cheverst, Nigel Davies und Alan Dix: *Exploiting Context in HCI Design for Mobile Systems.* In: *Proceedings of the First Workshop on Human Computer Interaction for Mobile Devices*, Seiten 12–17, Glasgow, Großbritannien, Mai 1998.

[RHW86] Rumelhart, David E., Geoffrey E. Hinton und Ronald J. Williams: *Learning Internal Representations by Error Propagation.* In: *Parallel Distributed Processing: Explorations in the microstructure of cognition; Vol. 1: Foundations*, Seiten 318–362, Cambridge, USA, 1986. MIT Press.

[Ros85] Ross, Sheldon M.: *Introduction to Probability Models.* Academic Press, 1985.

[RPM98] Ryan, Nick, Jason Pascoe und David Morse: *Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant.* In: *Computer Applications in Archaeology 1997*, British Archaeological Reports, Oxford, Großbritannien, Oktober 1998.

[SAW94] Schilit, Bill, Norman Adams und Roy Want: *Context-Aware Computing Applications.* In: *IEEE Workshop on Mobile Computing Systems and Applications*, Seiten 85–90, Santa Cruz, USA, Dezember 1994.

[SBG99] Schmidt, Albrecht, Michael Beigl und Hans-W. Gellersen: *There is more to Context than Location.* Computer & Graphics Journal, 23(6):893–902, Dezember 1999.

[SDA99] Salber, Daniel, Anind K. Dey und Gregory D. Abowd: *The Context Toolkit: Aiding the Development of Context-Enabled Applications.* In: *Computer-Human Interaction (CHI'99)*, Seiten 434–441, Pittsburgh, USA, Mai 1999.

[Smi81] Smith, James E.: *A Study of Branch Prediction Strategies.* In: *Proceedings of the 8th Annual Symposium on Computer Architecture*, Seiten 135–147, Minneapolis, USA, Mai 1981.

[ŠRU99] Šilc, Jurij, Borut Robič und Theo Ungerer: *Processor Architecture - From Dataflow to Superscalar and Beyond.* Springer-Verlag, 1999.

[ST94] Schilit, Bill und Marvin Theimer: *Disseminating Active Map Information to Mobile Hosts.* IEEE Network, 8(5):22–32, September/Oktober 1994.

[TBPU03] Trumler, Wolfgang, Faruk Bagci, Jan Petzold und Theo Ungerer: *Smart Doorplate.* In: *First International Conference on Appliance Design (1AD)*, Bristol, Großbritannien, Mai 2003. Reprinted in Personal and Ubiquitous Computing 7(3-4): 221-226, Juli 2003.

[TLF97] Tyson, Gary, Kelsey Lick und Matthew Farrens: *Limited Dual Path Execution.* Technischer Bericht CSE-TR 346-97, University of Michigan, 1997.

[VGPU04a] Vintan, Lucian, Arpad Gellert, Jan Petzold und Theo Ungerer: *Person Movement Prediction Using Neural Networks.* In: *Modeling and Retrieval of Context*, Ulm, Deutschland, September 2004.

[VGPU04b] Vintan, Lucian, Arpad Gellert, Jan Petzold und Theo Ungerer: *Person Movement Prediction Using Neural Networks.* Technischer Bericht 2004-10, Institut für Informatik, Universität Augsburg, April 2004.

[WJH97] Ward, Andy, Alan Jones und Andy Hopper: *A New Location Technique for the Active Office.* IEEE Personal Communications, 4(5):42–47, Oktober 1997.

[WM03] Wittig, Frank und Christian Müller: *Implicit Feedback for User-Adaptive Systems by Analyzing the Users' Speech.* Technischer Bericht, Institut für Informatik, Universität des Saarlandes, Saarbrücken, 2003.

[YP92] Yeh, Tse-Yu und Yale N. Patt: *Alternative Implementation of Two-Level Adaptive Branch Prediction.* In: *Proceedings of the 19th Annual Symposium on Computer Architecture (ISCA-19)*, Seiten 124–134, Gold Coast, Australien, Mai 1992.

[YP93] Yeh, Tse-Yu und Yale N. Patt: *A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History.* In: *Proceedings of the 20th Annual International Symposium on Computer Architecture (ISCA-20)*, Seiten 257–266, San Diego, USA, Mai 1993.

[YS94] Young, Cliff und Michael D. Smith: *Improving the Accuracy of Static Branch Prediction Using Branch Correlation.* In: *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI)*, Seiten 232–241, San Jose, USA, Oktober 1994.

[Zel94] Zell, Andreas: *Simulation Neuronaler Netze*. Addison-Wesley, 1994.

[ZL77] Ziv, Jacob und Abraham Lempel: *A Universal Algorithm for Sequential Data Compression*. IEEE Transactions on Information Theory, 23(3):337–343, Mai 1977.

Abbildungsverzeichnis

Tabellenverzeichnis