Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3158

Effiziente vorhersage-basierte Verteilung von Kontext-Informationen in mobilen Systemen

Tobias Fink

Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	Dipl.-Inform. Stefan Föll
begonnen am:	1. März 2011
beendet am:	25. September 2011
CR-Klassifikation:	H.4.3, H.3.4

Zusammenfassung

Die weite Verbreitung von sozialen Netzwerken und Smartphones in den letzten Jahren hat bisher zur Möglichkeit geführt von Smartphones mobil seinen Status upzudaten oder seinen Kontakten zu übermitteln, dass man sich an einem bestimmten Ort befindet.

In einem nächsten Schritt könnte das Smartphone bestimmte Informationen über den Benutzer über seine Sensoren automatisiert in bestimten Zeitintervallen ermitteln, die Sensordaten zu einem diskreten Kontext reduzieren, und diesen Kontext automatisch an die Smartphones der Kontakte in einem sozialen Netzwerk übermitteln.

Da die Datenübertragungen und Rechenoperationen auch bei den heutigen Smartphones noch einen großen Anteil der Akkuleistung kosten, wird in dieser Arbeit untersucht, wie sich durch eine Ressourcen sparende Vorhersage die nächsten Kontexte und das Zeitintervall deren Eintreffens bestimmen lässt. Durch eine synchronisierte Vorhersage beim Veröffentlicher und dem Empfänger der Informationen müssen nur im Falle von falschen Voraussagen Korrekturen vom Veröffentlicher zum Empfänger geschickt werden.

Ziel ist es, mit möglichst wenig Korrekturnachrichten eine hohe Quote an richtig vorhergesagten Kontexten zu erreichen. Zu diesem Zweck werden verschiedene Prädiktoren auf Markov- und Semi-Markov-Basis simuliert und auf ihre Leistungsfähigkeit geprüft.

Inhaltsverzeichnis

1 Einleitung **7**
- 1.1 Motivation . . . 7
- 1.2 Ziel der Diplomarbeit . . . 8
- 1.3 Übersicht . . . 8

2 Hintergrund **9**
- 2.1 Verwandte Arbeiten Updateprotokolle . . . 9
- 2.2 Verwandte Arbeiten Vorhersage . . . 10
- 2.3 Diskussion . . . 11

3 Energie-effiziente Updateprotokolle **12**
- 3.1 Anwendungsszenario . . . 12
- 3.2 Systemmodell . . . 12

4 Vorhersage-basierte Updates **15**
- 4.1 Vorhersagemodelle . . . 15
  - 4.1.1 Vorhersage auf Basis des einfachen Markov Modells . . . 16
    - Beschreibung des Markov Modells . . . 16
    - Markov Greedy Prädiktor . . . 17
    - Markov Erwartungswertprädiktor . . . 17
  - 4.1.2 Vorhersage auf Basis des Semi-Markov Modells . . . 19
    - Beschreibung des Semi-Markov Modells . . . 19
    - Semimarkov Greedy Prädiktor . . . 20
    - Semimarkov Plateau Prädiktor . . . 21
    - Semimarkov Erwartungswert Prädiktor . . . 22
- 4.2 Lernen und Updates von Markov Modellen . . . 23
  - 4.2.1 Lernen der Kontextübergänge . . . 23
  - 4.2.2 Lernmodelle . . . 23
  - 4.2.3 Updates . . . 25
    - Kontext Updates . . . 25
    - Modell Updates . . . 26
- 4.3 Metrik zur Bestimmung der Abweichung . . . 28

5 Evaluation **29**
5.1 Evaluationsframework 29
5.2 Ergebnisse auf Basis von künstlichen Daten 31
5.2.1 Toleranz 31
5.2.2 Prädiktoren 33
Markov Greedy Prädiktor 33
Markov Erwartungswert Prädiktor 34
Semi-Markov Greedy Prädiktor 37
Semi-Markov Plateau 38
Semi-Markov Erwartungswert 41
5.2.3 Lernen 42
5.3 Ergebnisse auf Basis von Real-Welt Daten 44
5.3.1 CenseMe 44
5.3.2 Wang 47

6 Zusammenfassung und Ausblick **50**

Literaturverzeichnis **51**

Abbildungsverzeichnis

3.1 Zusammenspiel von Publisher und Client beim Kontext Update 14

4.1 Beispiel: Plateau 21
4.2 Überblick über mögliche Lernverhalten 24
4.3 Überblick: Beziehung von Server, Publisher, Client 25

5.1 Evaluationsframework 30
5.2 Versuch 1 - Markov Greedy und ohne Vorhersage, 2 Zustände, Eigentransitionswahrscheinlichkeit 0,7 32
5.3 Versuch 2 - Markov Greedy und ohne Vorhersage, 2 Zustände, Eigentransitionswahrscheinlichkeit 0,3 34
5.4 Ergebnisse Versuch 3 - Links: Modell 1 - Rechts: Modell 2 36
5.5 Ergebnisse Versuch 4 - Links: Modell 2 - Rechts: Modell 3 38
5.6 Ergebnisse Versuch 5 Prediction Accuracy und Absolute Message Overhead . 39
5.7 Ergebnisse Versuch 6 Prediction Accuracy bei Plateau-Breiten von 1 (links) und 4(rechts) 41
5.8 Ergebnisse Versuch 7 Markov Erwartungswert und Semi-Markov Erwartungswert 43
5.9 Ergebnisse Versuch 8 Lernen 44
5.10 CenseMe - Wahrscheinlichkeiten der Übergänge pro Zeitschritt 45
5.11 CenseMe - Prediction Accuracy für statisches Lernen (links) und dynamisches Lernen(rechts) 46
5.12 Wang - Wahrscheinlichkeiten der Übergänge pro Zeitschritt 48
5.13 Wang - Prediction Accuracy für statisches Lernen (links) und dynamisches Lernen(rechts) 49

Tabellenverzeichnis

4.1 Markov Modell in Matrix-Darstellung 16
4.2 Semi-Markov Modell in Matrixdarstellung 19
4.3 Ausschnitt aus einem Beispiel-Semi-Markov Modell 20
4.4 Ausschnitt aus einem Beispiel-Semi-Markov Modell 20
4.5 Gelerntes Semi-Markov Modell 23

5.1 Markov Modell mit 2 Zuständen und einer Eigentransitionswahrscheinlichkeit von 0,7 31
5.2 Daten zum Versuch 1 32
5.3 Daten zum Versuch 2 33
5.4 Semi-Markov Modelle zur Evaluation des Markov Erwartungswert Prädiktors 35
5.5 Daten zum Versuch 3 35
5.6 Semi-Markov Modell 3 zur Evaluation des Semi-Markov Greedy Prädiktors . 37
5.7 Daten zum Versuch 4 37
5.8 Daten zum Versuch 5 39
5.9 Daten zum Versuch 6 40
5.10 Semi-Markov Modell 4 zur Evaluation des Semi-Markov Erwartungswert Prädiktors 42
5.11 Daten zum Versuch 7 42
5.12 Daten zum Versuch 8 43
5.13 Daten zum Versuch 9 46
5.14 Daten zum Versuch 10 48

1 Einleitung

1.1 Motivation

In letzter Zeit werden mobile Geräte immer leistungsfähiger und sind immer weiter verbreitet. Mit diesen Smartphones, PDAs und Tablets lassen sich verschiedene Informationen über den Benutzer feststellen: Bewegt er sich? Wo ist er gerade? Sitzt er oder steht er? Die gleichzeitige Verbreitung von sozialen Netzwerken wie Facebook, Google+ oder StudiVZ legt den Schritt nahe, dass ein Mensch genau diese Informationen, auch mit seinen Freunden teilen können soll.

Dieses jedoch kostet Ressourcen, die auf mobilen Geräten immer noch beschränkt sind. Die Rechenleistung der mobilen Geräte entwickelt sich zwar rasant, verbraucht unterwegs jedoch spürbar viel Energie aus dem Akku, wenn sie abgerufen wird. Der Arbeitsspeicher der mobilen Geräte nimmt zwar ebenfalls zu, ist jedoch beispielsweise aktuell auf einem Android-Smartphone auf circa 20 MB pro Applikation beschränkt. Den größten Anteil an den Energiekosten unterwegs aber hat bei den meisten Anwendungen das Übertragen der Daten.

Es ist also sinnvoll, die unterwegs zu übertragende Datenmenge durch verschiedene Techniken zu reduzieren. Eine solche Technik basiert darauf, dass sich die Informationen über den Benutzer vorhersagen lassen. Er könnte 7 Stunden im Bett verbringen, danach 20 Minuten im Bad, dann 30 Minuten in der S-Bahn, und dann 8 Stunden im Büro. Statt jede Änderungen jedes mal neu zu übertragen, ließe sich eine Vorhersage einführen, durch die die Kontakte im sozialen Netzwerk über den Standort informiert sind. Daten müssten dann nur noch bei Abweichungen dieses Ablaufes zur Korrektur einer falschen Vorhersage gesendet werden.

Diese Arbeit beschäftigt sich mit verschiedenen Vorhersagemodellen und deren Leistungsfähigkeit.

Die Probleme für die Privatsphäre, die entstehen, wenn sensible persönliche Daten wie Standort und Bewegung weitergegeben werden, werden in dieser Arbeit nicht behandelt.

1.2 Ziel der Diplomarbeit

Das Ziel dieser Diplomarbeit ist es, einen Weg zu finden, die übertragenen Daten in einem mobilen sozialen Netzwerk zu reduzieren. Dieses soziale Netzwerk würde auf dem Smartphone des Benutzers laufen, und automatisch in bestimmten Abständen verschiedene Informationen über den Benutzer an seine Kontakte übermitteln. Diese Informationen beinhalten diskrete, abgeleitete Daten wie "Benutzer sitzt zu Hause" und werden mit dem Begriff *Kontext* zusammengefasst.

Um Datenübertragungen einzusparen, werden Prädiktoren eingeführt, die den nächsten Kontext und den Zeitpunkt des Wechsels zum neuen Kontext vorhersagen. Wenn die Vorhersage zutrifft, lässt sich die Datenübertragung eines Kontextes sparen.

1.3 Übersicht

Zunächst soll im Kapitel 2 der Begriff des Kontexts eingeführt werden und ein Überblick über die verwandten Arbeiten zu den Themen Updateprotokolle und Vorhersage gegeben werden. Anschließend werden die Unterschiede zu dieser Arbeit aufgezeigt.

Danach wird im Kapitel 3 noch einmal das Anwendungsszenario umrissen um dann einen Überblick über das Systemmodell zu geben. Dort werden die vorkommenden Akteure und die wichtigen Begriffe eingeführt.

Auf die wichtigen Grundlagen wird in Kapitel 4 eingegangen. Dort werden die benutzten Vorhersagemodelle und Prädiktoren vorgestellt, die es ermöglichen sollen den Kontext des Benutzers ohne Datenübertragung vorherzusagen. Dann wird auf die zu übertragenden Daten eingegangen und ein Tradeoff zwischen der Genauigkeit der Vorhersage und übertragener Datenmenge vorgestellt.

Die Evaluation der Prädiktoren mit verschiedenen Datensätzen wird dann in Kapitel 5 präsentiert.

Am Schluss gibt es in Kapitel 6 noch eine zusammenfassung der Ergebnisse und einen Ausblick.

2 Hintergrund

Nach Dey [DAS99] bezeichnet der Kontext eines Objektes *jede Information, die benutzt werden kann um die Situation des Objektes zu beschreiben. Dabei kann das Objekt eine Person, ein Ort, oder ein physikalisches oder virtuelles Objekt sein. Die Information kann physische Gesten, Beziehungen zwischen Leuten und Objekten in der Umgebung, Eigenschaften der physischen Umgebung wie räumliche Anordnung und die Temperatur, sowie Identität und Ort von Leuten und Objekten in der Umgebung bezeichnen.* Teile dieses Kontexts können von einem mobilen Gerät erfasst und an andere Geräte gesendet werden.

Im folgenden Kapitel 2.1 gibt es einen Überblick über verwandte Arbeiten, welche sich mit der Übertragung sich ändernder Kontexte befassen. In Kapitel 2.2 gibt es dann einen Abriss über verwandte Arbeiten zum Thema Kontextvorhersage. In Kapitel 2.3 werden schließlich die Unterschiede dieser Arbeit zu den anderen Arbeiten herausgestellt.

2.1 Verwandte Arbeiten Updateprotokolle

In dieser Arbeit wird nicht näher darauf eingegangen, wie genau der Kontext aus den Sensordaten eines Smartphones berechnet wird. Was das betrifft, möchte ich auf Mayrhofer [MRF04] verweisen. Er beschreibt dort eine Architektur, um aus Sensordaten semantische Informationen zu gewinnen. Dabei werden die Merkmale der Sensordaten analysiert, klassifiziert und daraus namentliche Kontexte vorhergesagt.

Um einen Kontext von einer Quelle zu einem Ziel zu übertragen, gibt es mehrere Möglichkeiten. Diese werden von Rothermel und Leonhardi [RL00] in Update Protokolle im Pull-Verfahren (querying) und Update Protokolle im Push-Verfahren (reporting) aufgeteilt. Bei Updates im Pull-Verfahren fordert die Quelle vom Ziel den neuen Kontext an. Bei Updates im Push-Verfahren wird der neue Kontext von der Quelle unaufgefordert zum Ziel geschickt. Updates im Pull-Verfahren lassen sich nach Rothermel und Leonhardi in drei Untergruppen einteilen: “simple”, wenn der Kontext, jedes Mal wenn er im Ziel benötigt wird, neu angefordert wird. “cached”, wenn der Kontext auf einem Server zwischengespeichert wird, und nur neu angefordert wird, wenn der zwischengespeicherte Wert möglicherweise nicht mehr genau genug ist. “periodic”, wenn der Server in einem regelmäßigen Intervall den aktuellen Wert von der Quelle lädt und zwischenspeichert. Updates im Push-Verfahren werden nach Rothermel und Leonhardi in vier Untergruppen aufgeteilt: “simple” - die

Quelle verschickt bei jeder Änderung den neuen Kontext. “Time-based”, wenn die Quelle in festen Zeitintervallen den aktuellen Kontext verschickt. “Distance-Based” - die Quelle verschickt ein Update nur, wenn der aktuelle Kontext mindestens um einen bestimmten Wert vom letzten übermittelten Kontext abweicht. “Dead-reckoning” beinhaltet einen Server, der den aktuellen Kontext des Publishers vorhersagt, und eine Quelle, die ein Update an den Server verschickt, wenn der aktuelle Kontext mindestens um einen bestimmten Wert von der Vorhersage abweicht. Nach Wolfson [WSCY99] lassen sich mit diesem Vorgehen bei kontinuierlichen Ortsdaten 85% der Update-Kosten sparen, wenn der Weg eines vorherzusagenden Objektes auf dem Server bekannt ist.

Musolesi und andere [MPF+10] diskutieren und erproben diverse Update-Strategien an Realwelt-Daten.

2.2 Verwandte Arbeiten Vorhersage

Der Hauptteil der ähnlichen Arbeiten, die zum Thema Kontextvorhersage existieren, befasst sich mit der Vorhersage von Ortsdaten. Diese wiederum dienen meist einem von zwei Anwendungszwecken: zur Vorhersage eines kontinuierlich wechselnden Standortes eines Objektes; oder zur Vorhersage der nächsten benutzten Funkzelle in einem drahtlosen Netzwerk. Im Letzteren Falle wird wie in dieser Arbeit mit einem diskreten Kontext gearbeitet, allerdings mit dem Unterschied, dass bei der Vorhersage der nächsten Funkzelle der Zeitpunkt des Wechsels keine Rolle spielt. Diesen Zeitpunkt vorher zu sagen ist eines der Ziele dieser Arbeit.

Wenn nun bereits eine Folge von historischen Kontexten existiert, lassen sich aus dieser Folge über verschiedene Verfahren die nächsten Kontexte und der Zeitschritt des Wechsels vorhersagen. Die Genauigkeit dieser Vorhersagen hängt dabei von verschiedenen Faktoren ab. Zu diesen Faktoren gehören die Häufigkeit des Kontext-Wechsels, wie stark sich Folgen darin wiederholen, und wie die Häufigkeit dieser Folgen sich über die Zeit ändert.

In [BEYY04] werden dazu verschiedene Informationsreihen mit verschiedenen Vorhersagemodellen getestet um die Leistungsfähigkeit der Vorhersagemodelle zu evaluieren. Als Vorhersagemodelle werden dort Modelle auf Basis von Markov-Ketten verschiedener Länge benutzt. Untersucht werden dort der verlustfreie Kompressionsalgorithmus von Lempel und Ziv (LZ78) [ZL78], Prediction By Partial Match (PPM) nach [CW84], Context Tree Weighting Method (CTM) nach [WS95], Probabilistic Suffix Trees (PST) nach [RST96] und eine verbesserte Version des Lempel-Ziv nach [BCW90]. Als Testreihen dienen verschiedene Texte, Midi-Dateien und Proteine. In den Texten und den Midi-Dateien ist CTM der beste Prädiktor. Bei den Proteinen ist es PPM, gefolgt von CTM.

Katsaros und Manolopoulos [KM09] untersuchen konkret die Vorhersage von Ortsdaten in einem drahtlosen Netzwerk mittels PPM, LZ78, PST und CTW. Sie kommen mit synthetischen

Tests zu dem Ergebnis, dass sich PPM bei stark begrenzten Hardwareressourcen am Besten eignet. Allerdings sollte die Ordnung der Markov-Ketten größer sein, als die Sequenzen gleicher vorherzusagender Zustände lang sind. Ist die Varianz der Sequenzlängen in einem Testfall hoch, eignen sich nach Katsaros und Manolopoulos PST besser.

In [SKJH03] wird anhand der Auswertung der Nutzung verschiedener Accesspoints des WLANs der Universität Dartmouth festgestellt, dass sich einstufige Markov-Ketten am besten zur Vorhersage des von einem Studenten als nächstes genutzten Accesspoints eignen.

Mayrhofer beschreibt in [May05] die verschiedenen Möglichkeiten, Probleme und Herausforderungen der proaktiven Verwendung historischer Kontexte. Diese lassen sich zur Adaption technischer Systeme an den Menschen, zur Unfallvermeidung, zur Warnung und als Planungshilfe verwenden. Dazu muss der Kontext während einer Lernphase über mehrere Perioden des zu vorhersagenden Zeitraumes gelernt werden. für das Lernen gibt es zwei Arten: Eine Art ist überwachtes Lernen, bei denen der erste Teil einer Kontexthistorie gelernt wird, und der zweite Teil vorhergesagt werden muss. Die zweite Art ist unüberwachtes lernen. Hierbei wird die gesamte Kontexthistorie gelernt und die Zielwerte sind unbekannt. Als wichtige Herausforderungen werden unter anderem die Verbesserung der Vorhersagegenauigkeit, das zurecht kommen mit begrenzten Ressourcen und das Teilen des Kontextes zwischen verschiedenen Geräten genannt.

2.3 Diskussion

Der Hauptpunkt, der diese Arbeit von den meisten unterscheidet, ist die Verbreitung des Kontextes über ein mobiles soziales Netzwerk. Abgesehen von der Notwendigkeit die verfügbare Energie auf dem mobilen Gerät effizient zu nutzen, wird in dieser Arbeit die Zeitkomponente der Kontextübergänge stark betont. In einem sozialen Netzwerk kommt es auf den Zeitpunkt der Kontextänderung genauso an wie auf den neuen Kontext. Hierzu wird ein Semi-Markov Modell eingeführt, um das Zeitverhalten der Kontextübergänge besser zu beschreiben. Dabei wird von Erkennung des Kontextes durch ein Smartphone in einem regelmäßigen Intevall ausgegangen. Um das Zeitverhalten der Kontextübergänge vorherzusagen werden neue Prädiktoren auf Basis von Markov- und Semi-Markov Modellen vorgeschlagen und evaluiert. Um falsche Vorhersagen zu korrigieren wird das Konzept des Dead-reckoning aus [RL00] modifiziert.

Nicht eingegangen wird in dieser Arbeit auf Unterbrechungen des Datenflusses im mobilen Netz. Die korrekte Erkennung des Kontexts wird ebenfalls vorausgesetzt.

3 Energie-effiziente Updateprotokolle

3.1 Anwendungsszenario

Weltweit vernetzen sich immer mehr Personen in sozialen Netzwerken wie Facebook, Google+ etc. um miteinander Informationen auszutauschen. Diese Informationen bestehen bisher zumeist aus selbst eingegebenen Textnachrichten.

Mit den heutigen Smartphones, die von der Rechenleistung immer näher an Notebooks heranwachsen, lassen sich automatisiert Informationen, die auch für Freunde in sozialen Netzwerken interessant sein können, erfassen. Diese Informationen wie der momentanen Standort, die Bewegungsart, und andere lassen sich dank Smartphones, die immer online sind, direkt an die Freunde verbreiten.

Auf dieser technischen Grundlage sind bereits eine Vielzahl von Diensten wie 4Square und Plazes im Internet entstanden. Mit diesen Diensten kann man seinen eigenen Standort mit Freunden teilen. Auch in den verbreiteten sozialen Netzwerken wie Twitter, Facebook und Google+ lässt sich inzwischen der eigene Standort an Freunde weitergeben.

Dadurch wächst die Anzahl der Personen, die dieses Feature benutzen, im Moment sehr stark. Auch müssen die geteilten Informationen nicht nur auf den Ort beschränkt sein. Somit wächst auch die Belastung des Mobilfunknetzes durch eben diese Updates potentiell stark. Weiterhin verkürzen die Updates über das Mobilfunknetz die Akkulaufzeit des Smartphones. Durch eine Reduzierung oder Optimierung dieser Updates besteht also ein wachsendes Einsparpotenzial, das es zu nutzen gilt.

3.2 Systemmodell

Das Modell, in dem das soziale Netzwerk und die Verbreitung der Kontextinformationen simuliert wird, setzt sich wie folgt zusammen:

In einem verteilten Netzwerk gibt es mehrere Teilnehmer, die in der Form eines sozialen Netzwerkes miteinander befreundet sind. Teilnehmer können den Kontext anderer Teilnehmer abonnieren. Die Beziehungen der Teilnehmer entsprechen einem gerichteten Graphen. Ein Teilnehmer, der seinen Kontext anderen Teilnehmern zur Verfügung stellt, wird als *Publisher* bezeichnet. Ein Teilnehmer, der den Kontext eines anderen Teilnehmers abonniert

hat, wird in dieser Funktion als *Client* bezeichnet. Damit nicht jeder Publisher mit jedem Client kommunizieren muss, wird ein *Server* zwischengeschaltet. Der Publisher sendet alle Daten also erst zum Server, von wo aus die Daten an alle Clients weitergeleitet werden.

Der Kontext des Publishers wird vom mobilen Gerät in festen *Zeitschritten* ermittelt. Dazu lassen sich unter anderem Informationen aus Beschleunigungssensoren, Neigungssensoren, GPS, dem Mobilfunknetz oder dem eingebauten Mikrofon abgreifen. Aus diesen kontinuierlichen und abstrakten Informationen lassen sich dann diskrete und benannte Zustände wie "Benutzer sitzt zuhause am Schreibtisch" oder "Benutzer joggt im Park" extrahieren. Wie diese Extraktion funktionieren kann beschreibt Mayhofer in [MRF04]. Diese benannten Zustände stellen also den diskreten Kontext eines Publishers dar und werden im weiteren als *Kontext* bezeichnet.

Die Menge der möglichen benannten Kontexte C besteht also aus den Einzelkontexten $c_1, c_2, \ldots, c_{max}$, wobei ein Einzelkontext c_1 zum Beispiel für "Benutzer sitzt zuhause am Schreibtisch" stehen kann.

Die auf dem Smartphone entstandene Kontexthistorie kann für verschiedene Vorhersagemodelle zum Lernen verwendet werden. Das Lernen wird in Kapitel 4.2 genau beschrieben.

Eine *Vorhersage* V auf Basis eines Markovmodells M oder auf Basis eines Semi-Markov Modelles S soll in dieser Arbeit zu einem *aktuellen Zustand* c_t eines Publishers den *Folgezustand* und die Anzahl der Zeitschritte n bis zum *Eintrittszeitpunkt in den den Folgezustand* herausfinden. Dabei wird die Anzahl der Zeitschritte d berücksichtigt, die der Publisher schon im aktuellen Kontext verweilt.

Vorhersage $V(M, c_t, d) = (c_{t+n}, n)$

Eine Vorhersage mit $d > 0$ kommt nur am Anfang der Simulation und bei einem Kontext Update des Publishers an den Client vor.

Damit die Anzahl der gesendeten Nachrichten mit den Vorhersagen gesenkt werden kann, gibt es einen Vorhersagemechanismus, der auf Publisher und Client *synchron* läuft, d.h. nach einem Zeitschritt sind auf Publisher und Client die gleichen Informationen vorhanden. (siehe Abb. 3.1) Es wird also das Dead-reckoning-Verfahren angewendet. Dadurch kann der Publisher feststellen, wie sehr die Vorhersage auf dem Client vom tatsächlichen Kontext abweicht. Damit lässt sich eine Metrik definieren, die festlegt wann der Publisher den Client über den tatsächlichen Kontext informiert. Ziel ist es, die Informationen, die von Publisher zum Client geschickt werden zu reduzieren, da bei guter Vorhersage nicht bei jeder Änderung ein Update geschickt werden muss.

Die Updates werden unterwegs über das Mobilfunknetz gesendet und empfangen. Das hat zur Folge, dass Teilnehmer nicht zu jedem Zeitpunkt Updates verschicken oder empfangen können. Jeder Sendeversuch und jeder Empfang kostet Energie aus dem Akku des Smartphones. Zusätzlich variiert der Energieverbrauch noch je nach benutztem Netz und der Datenmenge. Was die Erreichbarkeit betrifft, gehe ich für die Untersuchungen davon aus,

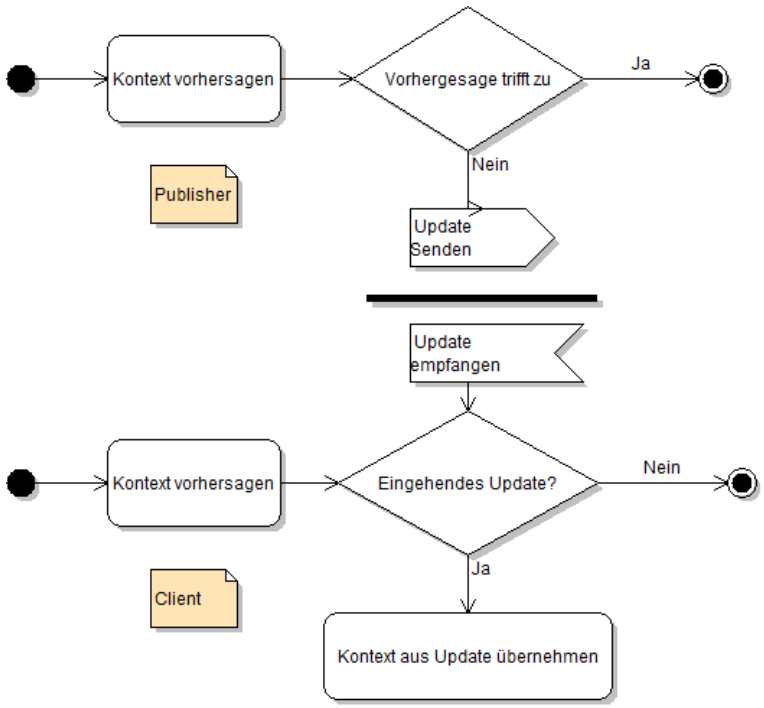

Abbildung 3.1: Zusammenspiel von Publisher und Client beim Kontext Update

dass ein Smartphone in der Stadt mindestens einmal in einem Zeitschritt erreichbar ist und Daten übertragen werden können.

Dazu muss die Länge der Zeitschritte entsprechend gewählt werden. Vorwiegend sollte die Länge der Zeitschritte von der Wechselhäufigkeit des Kontextes abhängen. In dieser Arbeit wird davon ausgegangen, dass die Zeitschritte so klein sind, dass sich der gemessene Kontext im Publisher maximal einmal pro Zeitschritt ändert.

4 Vorhersage-basierte Updates

Um die übertragenen Daten zwischen Publisher und Client zu reduzieren, läuft auf beiden Seiten synchron eine Vorhersage. Der Kontext muss dann nur gesendet werden, wenn die Vorhersage nicht eintrifft oder länger als toleriert einen falschen Wert annimmt..

Im folgenden Kapitel 4.1.1 wird das benutzte Markov-Modell und die darauf basierenden Prädiktoren beschrieben. Danach wird in Kapitel 4.1.2 ein um eine Zeitkomponente erweitertes Semi-Markov Modell beschrieben und darauf basierende Prädiktoren eingeführt. Kapitel 4.2 beschäftigt sich dann mit den Mechanismen zur Erstellung der Modelle durch Lernen und der Übertragung der für die Vorhersage nötigen Daten zwischen Publisher und Client. Schließlich wird in Kapitel 4.3 noch eine Metrik vorgestellt, die anhand der beim Client gewünschten Genauigkeit festlegt, wann ein Kontext übertragen werden muss.

4.1 Vorhersagemodelle

Wenn in einem Trivialbeispiel ohne Vorhersage ein Client den Kontext eines Publishers empfangen möchte, muss, um 100% Genauigkeit auf dem Client zu erhalten, bei jedem Kontextwechsel des Publishers eine Nachricht an den Client gesendet werden. Der Client wiederum geht in Abwesenheit eines Updates davon aus, dass sich der Kontext des Publishers nicht geändert hat.

Somit muss bei jeder Kontextänderung des Publishers ein Update versendet werden, außer, wenn die erlaubte Toleranz für inkorrekt vorhergesagte Zustände größer als 0 ist und der Publisher innerhalb der Toleranz in den letzten bekannten aktuellen Kontext zurückkehrt.

Die anderen Prädiktoren müssen sich also an diesem einfachen Fall ohne Vorhersage messen lassen.

$V_0(\emptyset, c_t, d) = (c_t, \infty)$

Wenn also der aktuelle Kontext bereits seit d Zeitschritten c_t ist, dann besagt die Vorhersage ohne Prädiktor, dass dieser Kontext für unendlich viele Zeitschritte der aktuelle Kontext bleibt.

In der Klassifikation von Rothermel und Leonardi [RL00] implementiert der Fall ohne Prädiktor also das “simple” Push-Verfahren. Wird ein Prädiktor eingesetzt, wird ein modifiziertes

"Dead-reckoning" Verfahren angewendet, bei dem die Vorhersage im Publisher und im Client synchron laufen. Dadurch kann der Publisher genau bei Abweichungen Kontext Updates versenden.

In dieser Arbeit werden als Vorhersage-Modelle einstufige Markov-Ketten und Semi-Markov-Ketten verwendet. Die Modelle auf Semi-Markov Basis sind gegenüber den normalen Markov-Modellen um einen Zeitaspekt angereichert.

Im Folgenden werden die beiden Vorhersage-Modelle und die zugehörigen Prädiktoren vorgestellt.

4.1.1 Vorhersage auf Basis des einfachen Markov Modells

Beschreibung des Markov Modells

Der Kontext wird in vielen der verwandten Arbeiten über so genannte Markov Modelle vorhergesagt. In einer n-stufigen Markovkette wird der *Folgekontext*, der nach dem aktuellen Kontext eintrifft, auf Basis der letzten n Kontexte vorhergesagt.

Einstufige Markovmodelle lassen sich als zweidimensionale Matrix darstellen:

	c_1	c_2
c_1	$P_M(c_1, c_1)$	$P_M(c_1, c_2)$
c_2	$P_M(c_2, c_1)$	$P_M(c_2, c_2)$

Tabelle 4.1: Markov Modell in Matrix-Darstellung

$P_M(c_1, c_2)$ bezeichnet die Wahrscheinlichkeit, dass der Publisher in einem Zeitschritt von Kontext c_1 nach Kontext c_2 übergeht. P_M steht für die Übergangswahrscheinlichkeit nach einem Markovmodell und hat als Parameter den Ausgangskontext zur Zeit t_n und den Folgekontext zur Zeit t_{n+1}. $P_M(c_1, c_2) = 0,5$ bedeutet in einem gelernten Markovmodell also, dass während der Lernphase, wenn der Publisher im Kontext c_1 war, der Publisher sich einen Zeitschritt später jedes zweite Mal im Kontext c_2 befand.

Markov Greedy Prädiktor

Der Markov Greedy Prädiktor basiert auf einem einstufigen Markovmodell, repräsentiert durch eine Matrix. Wenn man davon ausgeht, dass der Kontext sich in jedem Zeitschritt verändert, lässt sich aus der Wahrscheinlichkeit der Folgezustände der nächste Zustand vorhersagen. Nach dem Greedy-Prinzip wird dazu jeweils der Zustand mit der maximalen Wahrscheinlichkeit vorhergesagt.

Wenn also der aktuelle Kontext bereits seit d Zeitschritten c_t ist, dann besagt die Vorhersage nach einem Markov Modell , dass nach einem Schritt direkt in den Folgekontext mit der größten Wahrscheinlichkeit gewechselt wird:

$V_{GM}(M, c_t, d) = (c_{Folge}, 1)$, für $P_M(c_1, c_{Folge}) > P_M(c_1, c) \forall c \in C$

Dies funktioniert gut, wenn sich der Kontext auf dem Publisher jeden Zeitschritt ändert. Wechselt der Publisher allerdings von einem Kontext c_1 nur mit einer Wahrscheinlichkeit kleiner als $0,5$ in einem Zeitschritt in einen anderen Kontext, wird als Folgekontext für c_1 immer c_1 vorhergesagt. In diesem Fall verhält sich der Markov Greedy Prädiktor als gäbe es keinen Prädiktor. Für $c_t = c_{t+1}$ ergibt sich also abgeleitet: $V_{NC}(M, c_t, n) = (c_t, \infty)$

Für Semi-Markov Modelle ist die Vorhersage mit dem Markov Greedy Prädiktor nicht definiert.

Markov Erwartungswertprädiktor

Der Markov Erwartungswertprädiktor ist eine modifizierte Version des Markov Greedy Prädiktors. Die gedankliche Prämisse ist, dass auf dem Publisher nicht in jedem Zeitschritt ein Wechsel des Kontexts passiert. Der Markov Erwartungswertprädiktor berechnet die Verweildauer in einem Kontext anhand eines Markovmodells.

Da die Zeit-Informationen im Markov Modell fehlen, lässt sich der Zeitschritt des Wechsels nur aus den Transitionswahrscheinlichkeiten interpolieren. Konkret ergibt sich für die Wechselwahrscheinlichkeiten eine geometrische Verteilung, wie in [Kato6] beschrieben:

- Wahrscheinlichkeit, dass im ersten Zeitschritt gewechselt wird: $1 - P(c_t, c_t)$
- Wahrscheinlichkeit, dass im zweiten Zeitschritt gewechselt wird: $P(c_t, c_t) \times (1 - P(c_t, c_t))$
- Wahrscheinlichkeit, dass im dritten Zeitschritt gewechselt wird: $P(c_t, c_t)^2 \times (1 - P(c_t, c_t))$
- u.s.w

Der Erwartungswert für die Verweildauer im aktuellen Kontext c_t ist also $\frac{1}{1-P_M(c_t,c_t)}$

Ist die Wahrscheinlichkeit der Eigentransition zwischen $0,5$ und $1,0$, ist der Erwartungswert für die Verweildauer in diesem Zustand größer als 1. Erst nach Ablauf der Verweildauer wird in den zweitwahrscheinlichsten Zustand gewechselt.

$V_{EW}(M, c_t, d) = (c_{Folge}, \lceil \frac{1}{1-P_M(c_t,c_t)} \rceil)$ für $P_M(c_1, c_{Folge}) > P_M(c_1, c) \forall cinC/c_t$

Dies verhindert, dass sich der Markov Erwartungswertprädiktor in diesen Fällen wie der Markov Greedy Predictor verhält.

4.1.2 Vorhersage auf Basis des Semi-Markov Modells

Beschreibung des Semi-Markov Modells

In einem Semi-Markov Modell wird das zu Grunde liegende Markovmodell um weitere Attribute erweitert.

In dieser Arbeit wird das Markovmodell um die Anzahl der Zeitschritte vor dem Kontextwechsel erweitert. Das Ziel ist, dass die zusätzlichen Informationen gegenüber einem normalen Markovmodell auch die Vorhersage positiv beeinflussen. Die Vorhersage muss dazu die zusätzlichen Zeitinformationen nutzen.

Statt die Information $P_M(c_1, c_2)$ für alle Kontexte c zu speichern, wird also die Information $P_S(c_1, c_2, t)$ für alle c und t gespeichert. P_S steht für die Wahrscheinlichkeit in einem Semimarkov Modell, und hat drei Parameter: Den Ausgangskontext, den Folgekontext, und die Anzahl der Zeitschritte, die der Publisher im Ausgangskontext bleibt (t). $t = 0$ ist der "Eintrittszeitpunkt", an dem der Publisher in den Zustand c_1 wechselt.

$P_S(c_1, c_2, 4)$ bezeichnet also die Wahrscheinlichkeit, wenn ein Publisher frisch in den Kontext c_1 gewechselt ist, dass er dann nach vier Zeitschritten in den Kontext c_2 wechselt.

Tabellarisch lässt sich die 3-Dimensionale Matrix mit den Dimensionen c_1, c_2 und t für ein Modell mit den zwei Zuständen c_1 und c_2 wie folgt darstellen, dabei gibt es für jeden Ausgangskontext eine Tabelle mit den Wahrscheinlichkeiten der Folgekontexte für die jeweiligen Zeitschritte:

Ausgangskontext c_1:

c_1	$t = 1$	$t = 2$	...	$t = \infty$
c_2	$P_S(c_1, c_2, 1)$	$P_S(c_1, c_2, 2)$	...	$P_S(c_1, c_2, \infty)$

Ausgangskontext c_2:

c_2	$t = 1$	$t = 2$	...	$t = \infty$
c_1	$P_S(c_2, c_1, 1)$	$P_S(c_2, c_1, 2)$	...	$P_S(c_2, c_1, \infty)$

Tabelle 4.2: Semi-Markov Modell in Matrixdarstellung

Wird für einen Ausgangskontext die Spalte $t = n, n \in [1, \infty]$ nicht aufgeführt, sind in dieser Spalte alle Werte 0.

Die Summe aller Wahrscheinlichkeiten von einem Ausgangskontext c_2 zu einem anderen Kontext $c, c \in C/\{c_2\}$ zu wechseln $\sum_{i=1}^{\infty}(\sum_{c \in C/\{c_2\}} P_S(c_2, c, i))$ ist in einem gelernten Semi-Markov Modell entweder 1 oder 0. 1, wenn der Publisher während der Lernphase aus

diesem Kontext je wieder heraus gewechselt ist. o, wenn der Publisher nicht aus dem Kontext heraus gewechselt ist.

Semimarkov Greedy Prädiktor

Der Semimarkov Greedy Prädiktor ist ein einfacher Prädiktor auf Basis eines Semi-Markov Modells. Aus dem Semimarkov Modell sucht der Semimarkov Greedy Prädiktor für einen Kontext die wahrscheinlichste Kombination aus Folgekontext und Wechselzeitpunkt in Form des nächsten lokalen Wahrscheinlichkeits-Maximums heraus. Dazu wird das Semimarkov-modell des Ausgangskontextes einfach nach dem maximalen Wert durchsucht.

Sei der Ausgangskontext c_2 und der Abschnitt im Semimarkovmodell S für c_2 wie folgt, t sei 1:

c_2	$t = 1$	$t = 2$	$t = 5$	$t = 23$
c_1	$0,0$	$0,1$	$0,2$	$0,4$
c_3	$0,1$	$0,0$	$0,2$	$0,0$

Tabelle 4.3: Ausschnitt aus einem Beispiel-Semi-Markov Modell

Der maximale Wert für $t > 1$ ist $0,4$, also wird als Folgezustand c_1 und als Wechselzeitpunkt $t = 23$ vorhergesagt.

$V_{SG}(S, c_t, d) = (c_{Folge}, n - d)$ mit maximalem $P_S(c_t, c_{Folge}, n)$. $c_{Folge} \in C/\{c_t\}$, $n \in [d, \infty]$

Der Semimarkov Greedy Prädiktor hat allerdings Probleme, wenn die Wahrscheinlichkeiten für einen Wechsel über eine große Zeitspanne relativ gleichmäßig verteilt sind.

Sei eine Folge an Kontexten der Publishers in hintereinanderfolgenden Zeitschritten wie folgt: c_1, c_1, c_1, c_1, c_1, c_1, c_1, c_1, c_1, c_1, c_2, c_2, ...

Die richtige Vorhersage wäre also der Wechselzeitpunkt $t = 10$ und der Folgezustand c_2.

Sei das Semimarkovmodell für c_1 wie folgt:

c_1	$t = 1$	$t = 2$	$t = 5$	$t = 10$
c_2	$0,2$	$0,1$	$0,1$	$0,1$
c_3	$0,1$	$0,15$	$0,15$	$0,1$

Tabelle 4.4: Ausschnitt aus einem Beispiel-Semi-Markov Modell

Die Vorhersage des Semimarkov Greedy Prädiktors wäre der Wechselzeitpunkt $t = 1$ und der Folgezustand c_2. In jedem Zeitschritt bis $t = 10$ würde also als Zustand im nächsten

Zeitschritt fälschlicherweise c_2 oder c_3 vorhergesagt. Bei einer Toleranz von 0 müsste also bis zum tatsächlichen Wechsel zum Zeitpunkt $t = 10$ in jedem Zeitschritt eine Korrekturnachricht geschickt werden, damit der Client über den korrekten Zustand informiert ist.

Damit wäre der Semimarkov Greedy Prädiktor in diesem Fall um Faktor 10 schlechter als wenn überhaupt keine Vorhersage gemacht würde. Um diesem Malus abzuhelfen wurde der Semimarkov Plateau Prädiktor entwickelt.

Semimarkov Plateau Prädiktor

Der Semimarkov Plateau Prädiktor ist eine Erweiterung des Semimarkov Greedy Prädiktors. Ziel ist es die beim Semimarkov Greedy Prädiktor entstehenden Kosten für wiederholte falsche Voraussagen zu verkleinern.

Sei das Folgende der Wahrscheinlichkeitsverlauf von $P_S(c_2, c_1, t)$.

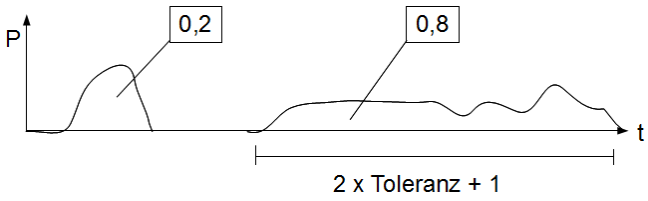

Abbildung 4.1: Beispiel: Plateau

Dann wäre ein lokales Maximum nach dem Semi-Markov Greedy Prädiktor am Anfang mit einem Peak mit einer Wahrscheinlichkeit von $0,2$. Viel wahrscheinlicher wäre allerdings ein Wechsel später. Wenn an dieser Stelle eine gewisse Ungenauigkeit beim Zeitpunkt der Vorhersage des Wechsels (Toleranz th, siehe 4.3) erlaubt ist, könnte man den Wechselzeitpunkt auf die Mitte des Plateaus mit Wahrscheinlichkeit $0,8$ vorhersagen.

Dazu wird im Semimarkov Modell eine Folge $P_S(c_2, c_1, n), P_S(c_2, c_1, n+1), \ldots P_S(c_2, c_1, n+b)$ gesucht mit $\sum_{i=n}^{n+b} P_S(c_2, c_1, i) > Sw$, wobei b die "Breite" an aufeinander folgenden Zeitschritten ist, deren Summe der Wahrscheinlichkeiten $P_S(c_2, c_1, i)$ einen Schwellenwert Sw erreichen oder übersteigen soll. Diese Folge wird *Plateau* genannt. Die Breite eines Plateaus sollte $2 \times th + 1$ nicht übersteigen: ein vorhergesagter Zeitpunkt und auf beiden Seiten die maximal erlaubte Abweichung th.

Vorhergesagt wird also der Folgekontext, der im Semimarkov Modell als nächstes ein solches Plateau aufweist. Als Zeitpunkt wird die Mitte des Plateaus $P_S(c_2, c_1, \lceil \frac{n+b}{2} \rceil)$ vorhergesagt. Wird eine solche Folge nicht gefunden, dann verhält sich der Semimarkov Plateau Prädiktor wie der No Change Prädiktor und sagt für die weiteren Zeitschritte immer den Ausgangszustand vorher.

$V_{PL}(S, c_t, n) = (c_{Folge}, d - n)$ wo c_{Folge} die Mitte eines Plateaus mit kleinst möglichem d ist. $c_{Folge} \in C/\{c_t\}$, $d \in [n, \infty]$

Semimarkov Erwartungswert Prädiktor

Der Semimarkov Erwartungswert-Prädiktor ist ebenfalls eine Erweiterung des Semimarkov Greedy Prädiktors. Er erzielt mit einer bereits im Ausgangszustand verbrachten Zeit $d = 0$ auf dem Semi-Markov Modell basierend die gleichen Ergebnisse wie der Markov Erwartungswert Prädiktor.

Als Erwartungswert für die Verweildauer wird hier eine nach Wahrscheinlichkeit gewichtete Summe über die Zeitschritte des wahrscheinlichsten Folgezustands gewählt:

Der vorhergesagte Folgekontext für einen Kontext c_1 ist derjenige der möglichen Folgekontexte C_f aus dem Markovmodell mit der höchsten Summe der Wahrscheinlichkeit über alle Zeitschritte. Es wird also der Folgekontext c_f aus C_f gesucht, für den $\sum_{i=1}^{\infty} P_S(c_1, c_f, i)$ maximal ist. Der vorhergesagte Zeitpunkt ist der mit P_S gewichtete Durchschnitt der Zeitschritte n: $\sum_{n=0}^{\infty} P_S(c_1, c_2, n) \times n$

$V_{DU}(S, c_t, d) = (c_f, \sum_{n=d}^{\infty} P_S(c_t, c_f, n) \times n)$ mit maximalem $\sum_{i=1}^{\infty} P_S(c_t, c_f, i)$.

Für $d > 0$ fließen also die Wahrscheinlichkeiten $P_S(c_t, c_f, n)$ mit $n < d$ nicht mehr in die Summe ein und es ergibt sich ein Unterschied zum Markov Erwartungswert Prädiktor.

4.2 Lernen und Updates von Markov Modellen

4.2.1 Lernen der Kontextübergänge

Das Lernen funktioniert grundlegend für Markov- und Semimarkov Modelle jeweils so, dass die absolute Zahl der Kontextübergänge als Summe abgelegt wird. Daraus werden dann die einzelnen Wahrscheinlichkeiten berechnet.

Es gebe die 2 Kontexte a und b und die nacheinander in jeweils einem Zeitschritt zu lernenden Kontexte aaababaabbaababa. Dann wäre das gelernte Markov-Modell eine 2×2 Matrix, bei der an jeder Stelle die Anzahl der entsprechenden Übergänge geteilt durch die Gesamtanzahl der Übergänge steht. 4 mal a $\rightarrow$ a; 5 mal a $\rightarrow$ b; 5 mal b $\rightarrow$ a;1 mal b $\rightarrow$ b:

$$\begin{pmatrix} \frac{4}{15} & \frac{5}{15} \\ \frac{5}{15} & \frac{1}{15} \end{pmatrix}$$

Das gelernte Semi Markov Modell sähe dann so aus, dass 2 mal nach einem a in b gewechselt wurde, 2 mal nach 2 a und 3 mal nach 3 a, das ganze auf 1 normiert:

Ausgangskontext a:

	$t=1$	$t=2$	$t=3$
b	$\frac{2}{5}$	$\frac{2}{5}$	$\frac{1}{5}$

Ausgangskontext b analog:

	$t=1$	$t=2$	$t=3$
a	$\frac{3}{4}$	$\frac{1}{4}$	0

Tabelle 4.5: Gelerntes Semi-Markov Modell

4.2.2 Lernmodelle

Abbildung 4.2 gibt einen Überblick, wie die gelernten Vorhersage-Modelle verwendet werden können.

Im entwickelten Framework lassen sich verschiedene Lernverhalten simulieren. Dazu gibt es einerseits die Möglichkeit, mit einem statischen Vorhersage-Modell zu arbeiten. Dieses wird im Publisher und im Client am Anfang einmal initialisiert und ändert sich danach nicht mehr. Das Vorhersage-Modell kann nun verschiedene Ursprünge haben: Ein synthetisches Modell ist ein Modell, dass nicht durch einen Lernvorgang entstanden ist. Es wird entweder von Hand oder algorithmisch generiert. Ein synthetisches Modell ist zum Beispiel nützlich um gezielt einzelne Aspekte der Vorhersage zu evaluieren. Dazu kann man sich aus dem Modell

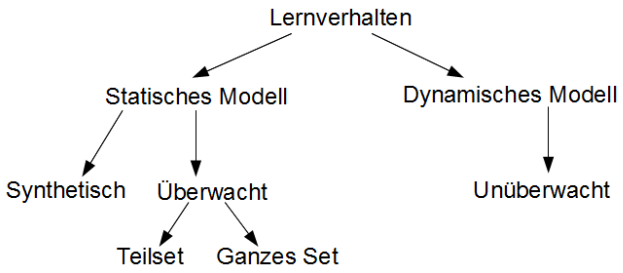

Abbildung 4.2: Überblick über mögliche Lernverhalten

auch ein Set an Kontext-Übergängen generieren lassen. Existiert bereits ein Set von Kontext-Übergängen, käme überwachtes oder unüberwachtes Lernen zum Einsatz. Von überwachten Lernen sprechen wir, wenn wir vor der eigentlichen Simulation aus dem gesamten Set oder einem Teil davon ein Vorhersage-Modell lernen. Dieses wird dann wiederum am gesamten Set, oder einem Ausschnitt davon, getestet.

Ein dynamisches Vorhersage-Modell wird direkt während der Simulation gelernt. Dadurch kann sich das Modell an wechselnde Benutzer-Gewohnheiten anpassen. Wie stark die neueren Kontextwechsel im Modell gewichtet werden lässt sich über einen *Lernfaktor* beeinflussen. Am Anfang der Simulation starten Publisher und Client mit einem leeren *gelernten Vorhersage-Modell* als *aktivem Vorhersage-Modell* M_a oder S_a. Auf Basis dieses aktiven Vorhersage-Modelles wird jeweils die Vorhersage für den nächsten Kontext errechnet. Über eine festgelegte Anzahl von Zeitschritten lernt der Publisher nebenher ein Teilset. Das daraus gewonnene Vorhersage-Modell M_l oder S_l wird nun mit einem Faktor, dem Lernfaktor $l : l \in [0;1]$ in das Vorhersage-Modell eingerechnet:

Im Markov-Modell wird damit jeder Eintrag $P_{M_a}(c_t, c_f)$ aus dem aktiven Vorhersage-Modell mit dem entsprechenden Eintrag $P_{M_l}(c_t, c_f)$ aus dem gelernten Modell des Publishers mit dem Lernfaktor zu einem neuen gelernten Modell M_{neu} zusammengerechnet:

$$P_{M_{neu}}(c_t, c_f) = P_{M_a}(c_t, c_f) \times (1 - l) + P_{M_l}(c_t, c_f) \times l$$

Analog dazu bei Semi-Markov Modellen für jeden Eintrag $P_M(c_t, c_f, d)$ aus dem aktiven und dem gelernten Vorhersage-Modell:

$$P_{S_{neu}}(c_t, c_f, d) = P_{S_a}(c_t, c_f, d) \times (1 - l) + P_{S_l}(c_t, c_f, d) \times l$$

Dieses neue gelernte Modell lässt sich dann direkt als *aktives Vorhersage-Modell* übernehmen, indem es vom Publisher an den Client geschickt wird und von beiden Seiten für die Vorhersage benutzt wird. Alternativ lässt sich anhand des Prädiktors überprüfen, ob sich durch Benutzung des neuen Vorhersage-Modelles etwas an der Vorhersage ändern würde. Ein Update an den Client muss dann nur erfolgen, wenn das der Fall ist.

Diese Prozedur des Lernens des aktuellen Teilsets und der darauffolgenden Prozedur aus dem Update des gelernten Modells und einer eventuellen Übernahme als aktives Vorhersage-Modell ist ein *Lernschritt*.

4.2.3 Updates

Mit *Updates* sind jegliche Arten von Informationsfluss vom Publisher zu einem Client gemeint. Ein Update läuft dabei immer vom Publisher über einen Server, der es dann an die Clients weiterverteilt, die am Kontext des Publishers interessiert sind. Ein Client kann also theoretisch den Kontext von beliebig vielen Publishern empfangen. Ein Publisher kann theoretisch beliebig viele Clients haben, die seinen Kontext empfangen.

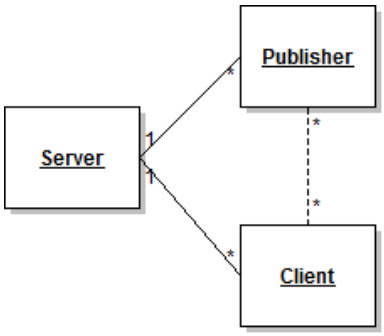

Abbildung 4.3: Überblick: Beziehung von Server, Publisher, Client

Da der Server nur eine passive Vermittlungsstation darstellt, wird der Server bei den Updates nicht jedes mal erwähnt und Beispielhaft die Übertragung von einem Publisher zu einem Client dargestellt.

Es gibt in der Simulation zwei Arten von Updates: *Kontext Updates* setzen den Client über den aktuellen Kontext in Kenntnis, z.B. wenn dieser von der Vorhersage abweicht. *Modell Updates* beinhalten ein neues Vorhersage-Modell, das ab dem Update von Client und Publisher angewendet wird.

Kontext Updates

Damit die Vorhersage funktioniert, müssen die Vorhersage-Modelle auf dem Publisher und auf dem Client synchron sein. Zusätzlich müssen Publisher und Client einen gemeinsamen *synchronisierten Kontext* c_{a_s} haben. Es gibt also durch ein Update einen aktuellen synchronisierten Kontext c_{a_s}, der auf dem Publisher und dem Client gleich ist. Im nächsten Zeitschritt wird dann auf beiden Seiten anhand des synchronen aktiven Vorhersage-Modelles VM die Anzahl der Zeitschritte n in c_{a_s} und der Folgekontext c_{f_s} berechnet:

$V(VM, c_{a_s}, d) = (c_{f_s}, n)$

Die Vorhersage besagt also, dass der Publisher noch für n Zeitschritte im Kontext c_a bleibt, und danach in Kontext c_{f_s} übergeht.

Damit nach einem Kontext Update auf Basis der Semi-Markov Prädiktoren eine korrekte Vorhersage gemacht werden kann, muss das Kontext Update außer dem aktuellen synchronisierten Kontext c_{a_s} auch die real vom Publisher schon in c_{a_s} verbrachten Zeitschritte enthalten.

Ein Kontext Update ist also ein Tupel (c_{a_s}, d).

Der Publisher prüft nun in den nachfolgenden Zeitschritten, ob die Vorhersage tatsächlich mit dem aktuellen Kontext auf dem Publisher übereinstimmt. Falls nein, entscheidet eine Metrik, ob die Abweichung noch im "erlaubten" Bereich ist. Auf diese Metriken wird im nächsten Kapitel 4.3 eingegangen. Ist die Abweichung innerhalb des Erlaubten, läuft die synchrone Vorhersage auch mit einem falschen synchronisierten Kontext synchron weiter. Ist die Abweichung außerhalb des Erlaubten, wird ein Kontext Update mit dem tatsächlichen aktuellen Kontext auf dem Publisher gesendet und anhand dessen auf Publisher und Client eine neue synchrone Vorhersage erstellt.

Liegt die Vorhersage nach n Zeitschritten noch im erlaubten Bereich, wird in Publisher und Client c_{f_s} als synchroner Kontext übernommen und eine neue Vorhersage gestartet.

Beim Kontext Update wird ein einzelner Kontext versendet. Wenn die Zahl der Kontexte 256 oder weniger ist, muss also,vom Overhead durch die Netzwerkprotokolle abgesehen, nur ein einzelnes Byte übertragen werden.

Modell Updates

Modell Updates dienen dazu, die aktiven Vorhersage-Modelle auf Publisher und Client synchron zu halten. Sie treten nur bei Simulationen mit dynamischem Modell auf.

In jedem Lernschritt wird das vorangegangene Teilset in das gelernte Modell des Publishers eingebaut. Wenn nun das neue gelernte Modell das Vorhersage-Modell so verändert, dass für manche Eingaben andere Vorhersagen getroffen werden, wird das neue Modell als Modell Update vom Publisher an den Client geschickt und fortan von beiden Seiten als aktives Vorhersage-Modell genutzt.

Ob sich durch ein modifiziertes Vorhersage-Modell andere Vorhersagen ergeben, hängt vor allem vom Prädiktor ab:

- Beim Markov Greedy Prädiktor ändert sich die Vorhersage, wenn sich für einen Kontext der wahrscheinlichste Folgekontext ändert.

- Beim Markov Erwartungswert Prädiktor ändert sich die Vorhersage, wenn sich entweder die Wahrscheinlichkeit der Selbsttransition P(c,c) für einen beliebigen Kontext c ändert, oder wenn sich für einen Kontext der wahrscheinlichste Folgekontext ändert.
- Beim Semimarkov Greedy Prädiktor ändert sich die Vorhersage $V_{SG}(S, c, d)$, wenn sich für ein beliebiges c und d die Position des nächsten lokalen Maximums ändert.
- Beim Semimarkov Erwartungswert Prädiktor ändert sich die Vorhersage, wenn sich der gewichtete Durchschnitt der Zeitschritte ändert. (siehe 4.1.2)
- Beim Semimarkov Plateau Prädiktor ändert sich die Vorhersage, wenn sich die Position eines Plateaus ändert.

Um beim Update die Größe der übertragenen Daten möglichst gering zu halten, bietet es sich an, die übertragenen Modelle zu modifizieren, dass sie zwar für den Prädiktor semantisch gleich bleiben, aber in möglichst wenig Bytes übertragbar sind.

Wie das funktionieren kann, lässt sich daraus ableiten, wann Änderungen Auswirkungen auf die Vorhersage haben:

- Markov Greedy Prädiktor: Die Markov-Matrix ist semantisch gleich, wenn sie jeweils an den Stellen des wahrscheinlichsten Folgezustandes den Wert 1 und sonst an allen Stellen den Wert 0 hat.
- Markov Erwartungswert Prädiktor: Die Markov-Matrix ist semantisch gleich, wenn die Selbsttransitionswahrscheinlichkeiten und die Position des wahrscheinlichsten Folgezustandes erhalten bleiben.
- Semimarkov Greedy Prädiktor: Für alle $P(c_1, c_2, d)$ muss jeweils das nächste lokale Maximum m mit $t > d$ erhalten bleiben. Die Werte zwischen d und m können auf 0 gesetzt werden.
- Semimarkov Erwartungswert Prädiktor: Jeweils die Stelle der nächsten Vorhersage bleibt erhalten, die anderen Stellen können auf 0 gesetzt werden.
- Semimarkov Plateau Prädiktor: Der letzte Eintrag des Plateaus wird auf die Gesammtsumme des Plateaus gesetzt, die restlichen Einträge werden 0.

Durch die Kompression lassen sich einige zu übertragende Informationen sparen, was den Energieerbrauch für die Übertragung besonders bei großen Modellen deutlich senken dürfte.

4.3 Metrik zur Bestimmung der Abweichung

Da die regelmäßigen Kontext-Updates immer noch den Akku eines Smartphones deutlich belasten können, soll an dieser Stelle ein Trade-Off zwischen der Genauigkeit der Kontexte auf dem Client und häufigen Übertragungen des Kontextes eingeführt werden.

Dazu wird zwischen Publisher und Client eine *Toleranz th* festgelegt, eine Anzahl von Zeitschritten, für die auf dem Client ein falscher Kontext vorhergesagt werden darf, ohne dass ein Update gesendet wird. Das ermöglicht es, dass Client und Publisher ohne Update wieder im richtigen Zustand sind, falls die Vorhersage zwar den richtigen Folgekontext vorhersagt, aber um bis zu *th* Zeitschritte beim Wechselzeitpunkt abweicht.

Der Publisher kann also den Clients den Kontext mit festgelegten Toleranzen zum abonnieren anbieten. Für diese Toleranzen muss er dann lokal eine Vorhersage betreiben, um zu wissen, wann er Kontext Updates verschicken muss. Dazu wird in jedem Zeitschritt bei Abweichung des aktuellen synchronen Kontextes c_{a_s} vom tatsächlichen aktuellen Kontext c_a ein Zähler hochgezählt. Überschreitet dieser Zähler den Wert th, wird ein Kontext Update verschickt. Sobald ein Kontext Update verschickt wird oder c_{a_s} und c_a ohne Update wieder übereinstimmen, wird der Zähler zurückgesetzt.

Es wäre nun möglich, die Toleranz abhängig vom momentanen Kontext zu setzen. Dadurch könnte der Fall realisiert werden, dass ein Client an den Zustand eines Publishers mit einer sehr niedrigen Genauigkeit interessiert ist - es sei denn er befindet sich an einem bestimmten Alarm-Kontext, der mit einem niedrigeren Toleranz-Wert verknüpft ist (z.B. “sitzt am Treffpunkt und wartet”). In unserem System müsste der Publisher dazu für jeden Client, der in einem anderen Alarm Kontext interessiert ist, eine eigene lokale Vorhersage betreiben. Dies wäre schon bei wenigen Clients ein zu hoher Rechenaufwand und wird daher in dieser Arbeit nicht untersucht.

5 Evaluation

Um die verschiedenen Prädiktoren zu testen, wurde im Rahmen dieser Arbeit ein Framework zur Simulation geschrieben. Dieses soll in Kapitel 5.1 beschrieben werden. Danach folgen in Kapitel 5.2 die Ergebnisse auf Basis von künstlichen Daten. In Kapitel 5.3 folgen dann die Ergebnisse auf Basis von Daten aus der realen Welt.

5.1 Evaluationsframework

In diesem Kapitel wird das entwickelte Framework zur Evaluation der Prädiktoren vorgestellt. Ziel war es, ein möglichst flexibles Frameworkzu entwickeln, mit dem sich viele verschiedene Evaluationen einfach und flexibel durchführen lassen. Dazu wurde auf ein Baukastenprinzip gesetzt, bei dem sich alle wichtigen Teile einfach austauschen lassen. Abbildung 5.1 gibt einen Überblick über die wichtigen Komponenten.

Diese Komponenten werden im Folgenden vorgestellt. um einen Simulationsdurchlauf durchzuführen, werden zuerst zu einer neuen Instanz der *Simulation* entsprechend Publisher mit ihren *Drivern* hinzugefügt. Ein Driver ist die Komponente, die in jedem Zeitschritt den aktuellen Kontext des Publishers festgelegt.

In dieser Arbeit benutzte Driver:

Markov-Driver - wechselt den aktuellen Kontext zufällig anhand der Wahrscheinlichkeitsverteilung eines Markov-Modells

Semi-Markov-Driver - wechselt den aktuellen Kontext anhang der in einem Semi-Markov Modell angegebenen Wahrscheinlichkeitsverteilung.

CenseMe-Driver - bezieht die Kontexte aus einer mit einem Smartphone aufgezeichneten Kontext-Historie des CenseMe Projektes.

Das Gegenstück zum Driver ist der *Prädiktor*, der auf dem Publisher und dem Client synchron läuft und vorhersagen soll, in welchem Kontext sich der Publisher befindet. Die benutzten Prädiktoren wurden bereits in Kapitel 4.1.1 und 4.1.2 vorgestellt.

Wann der Publisher nun ein Kontext Update verschickt, legt eine *Metrik* fest. Die in dieser Arbeit benutzte Metrik wurde bereits in Kapitel 4.3 beschrieben. Es sind in Framework aber

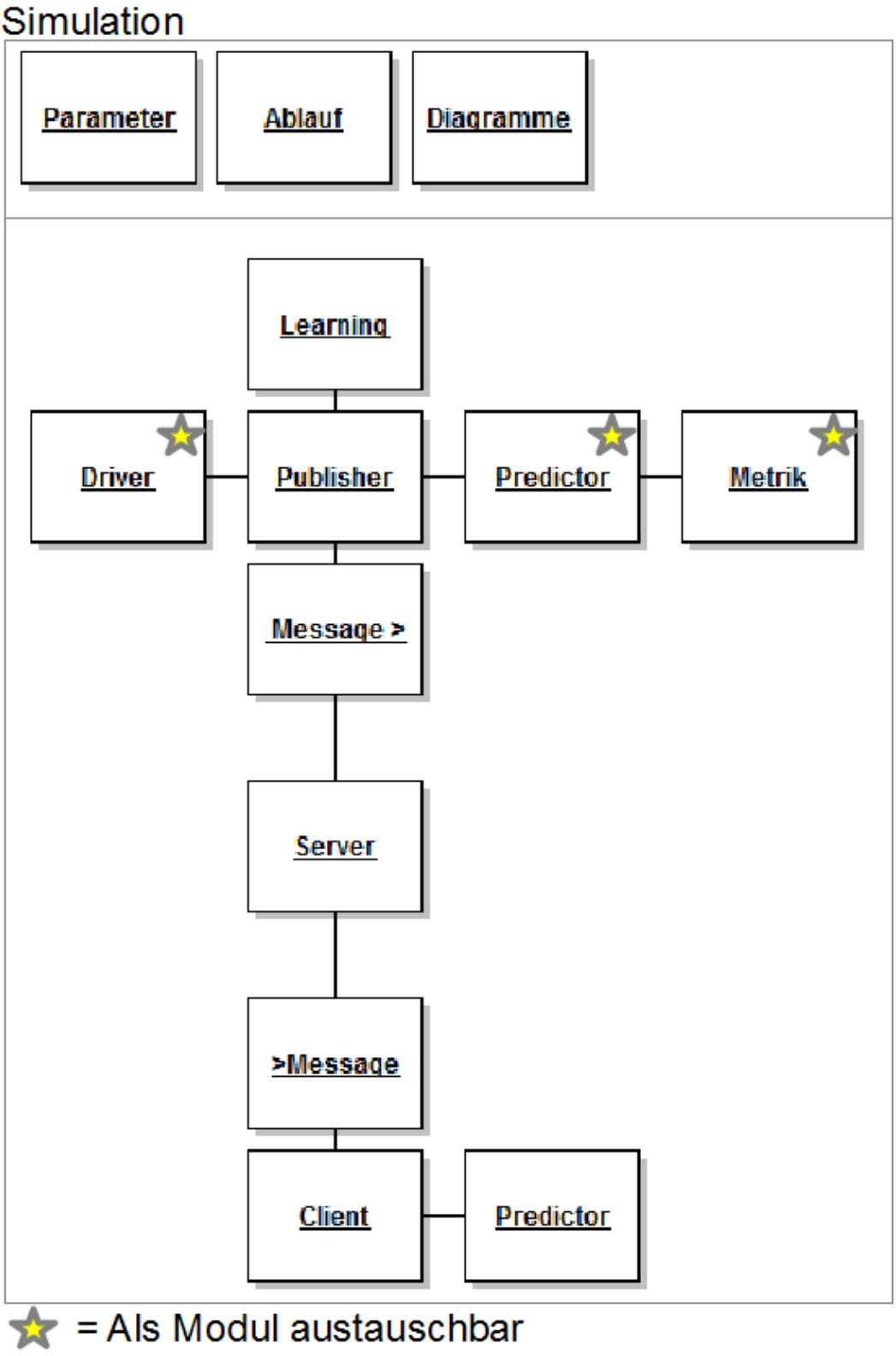

Abbildung 5.1: Evaluationsframework

auch andere Metriken vorhanden, z.B. “Persistent Change”, das nur ein Update verschickt, wenn der Publisher über eine bestimmte Zeitdauer in einem Kontext bleibt.

Die Kontext Updates und Modell Updates sind als eigene *Message*-Klassen gekapselt, die die Größe der gesendeten Nachricht angeben können. Die Nachrichten werden dann in jedem Zeitschritt vom Publisher aus verschickt und vom *Server* an die Clients zugestellt. Diese können dann evtl. den dem Benutzer angezeigten Kontext korregieren.

Damit der Publisher sein Vorhersage-Modell beim dynamischen Lernen im laufenden Betrieb weiterentwickeln kann, gibt es eine flexible Learning-Komponente, die aus eingegebenen Kontext-Übergängen sowohl ein Markov- als auch ein Semi-Markov Modell erstellt. Die Vorgehensweise dazu wird in Kapitel 4.2 beschrieben. Die Learning-Komponente lässt sich auch in einem Driver benutzen, um ein Modell ohne eine echte Simulation z.B. aus einer aufgezeichneten Kontext-Historie zu lernen.

Alle diese Komonenten, werden von der *Simulation* gekapselt, in der die einzelnen Zeitschritte simuliert werden. Hier können vor und während des Simulationslaufs Publisher und Clients hinzugefügt und verknüpft werden, und Prädiktoren und Driver ausgetauscht werden. Außerdem fließen in der Simulation die erhobenen Daten über Updates und Genauigkeit zusammen, welche von dort aus in ein Diagramm exportiert werden können.

5.2 Ergebnisse auf Basis von künstlichen Daten

Als Einstieg in die Evaluierung der Prädiktoren eignen sich am besten künstliche Daten, um einzelne Aspekte der Vorhersage zu untersuchen.

5.2.1 Toleranz

Die Toleranz th soll durch das Erlauben von $th \in [1, \infty]$ inkorrekt vorhergesagten Kontexten in Folge auf dem Client die Zahl der Kontextupdates deutlich senken.

Wir betrachten die Auswirkung der Toleranz mit dem Markov Greedy Prädiktor in einem Modell mit 2 Zuständen c_1 und c_2. Die Eigentransitionswahrscheinlichkeit beträgt $0,7$.

	c_1	c_2
c_1	$0,7$	$0,3$
c_2	$0,3$	$0,7$

Tabelle 5.1: Markov Modell mit 2 Zuständen und einer Eigentransitionswahrscheinlichkeit von 0,7

Dieses Modell wird nun sowohl zur Bestimmung des nächsten Kontextes, als auch als Vorhersage-Modell benutzt.

Versuchsaufbau 1

Prädiktoren:	ohne, Markov Greedy
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Markovmodell mit Eigentransitionswahrscheinlichkeit 0,7
Lernen:	ohne, statisches, synthetisches Modell
Zeitschritte:	10000/ Toleranz

Tabelle 5.2: Daten zum Versuch 1

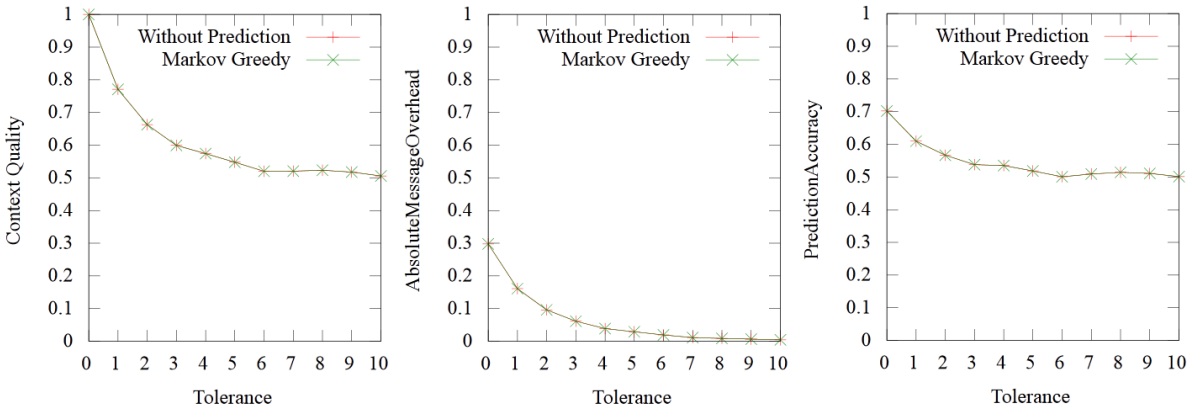

Abbildung 5.2: Versuch 1 - Markov Greedy und ohne Vorhersage, 2 Zustände, Eigentransitionswahrscheinlichkeit 0,7

Auf der y-Achse ist jeweils der Anteil zwischen 0 und 1 angegeben, auf der x-Achse die Toleranz.

Da der Markov Greedy Prädiktor bei Eigentransitionswahrscheinlichkeiten über 0,5 als Kontext für den nächsten Zeitschritt immer den aktuellen Kontext vorhersagt, überlagern sich die Kurven des Markov Greedy Prädiktors und des Vergleichswertes ohne Vorhersage wie erwartet.

Die *Context Quality*, der Anteil an Zeitschritten, in denen der Client korrekt über den aktuellen Kontext auf dem Publisher informiert war, beträgt bei einer Toleranz von 0 falschen Kontexten ohne Update 1. In den Fällen, in denen die Vorhersage nicht zutrifft, werden also Updates geschickt.

Wie viele Kontext Updates geschickt wurden, zeigt das Diagramm *Absolute Message Overhead*. Es stellt den Anteil der Zeitschritte dar, in denen eine Nachricht gesendet wurde. Bei einer Toleranz von 0 wurde in 30% der Zeitschritte ein Update versendet, also genau in den Fällen, in denen keine Selbsttransition eingetreten ist.

Im Diagramm *Prediction Accuracy* wird die Context Quality vom Absolute Message Overhead abgezogen, um die Anzahl der korrekten Vorhersagen zu erhalten. Es gehen dort also nur korrekte Vorhersagen ein, nicht wenn der Client durch ein Kontext Update über den korrekten Kontext auf dem Publisher informiert wurde.

Den Diagrammen in Abb. 5.2 lässt sich entnehmen, dass in diesem Beispiel die Qualität des Kontextes in einer ähnlichen Kurve wie die Anzahl der Updates abnimmt. Die Anzahl der Messages geht gegen 0, da es bei einer hohen Toleranz immer wahrscheinlicher wird, dass der Kontext des Publishers innerhalb der Toleranz in den Ausgangskontext wechselt. Die Qualität des Kontexts tendiert gegen $0,5$. Dies kommt daher, dass ohne ein Update immer weiter einer der beiden Kontexte vorhergesagt wird. Da beide Kontexte mit gleicher Häufigkeit auftreten, liegt die Vorhersage also in ungefähr der Hälfte der Fälle richtig. Die vereinzelten Messages bei langen Sequenzen ohne Kontextwechel beim Publisher bringen die Qualität des Kontextes dann knapp über $0,5$.

5.2.2 Prädiktoren

Markov Greedy Prädiktor

Im Versuch 1 mit hoher Selbsttransitionswahrscheinlichkeit liegt der Markov Greedy Prädiktor gleichauf mit dem Fall ohne Vorhersage. Mit einem Modell mit geringerer Selbsttransitionswahrscheinlichkeit kann der Markov Greedy Prädiktor nun eine Verbesserung gegenüber dem Fall ganz ohne Vorhersage erzielen.

Versuchsaufbau 2

Prädiktoren:	ohne, Markov Greedy
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Markovmodell mit Eigentransitionswahrscheinlichkeit $0,3$
Lernen:	ohne, statisches, synthetisches Modell
Zeitschritte:	10000/ Toleranz

Tabelle 5.3: Daten zum Versuch 2

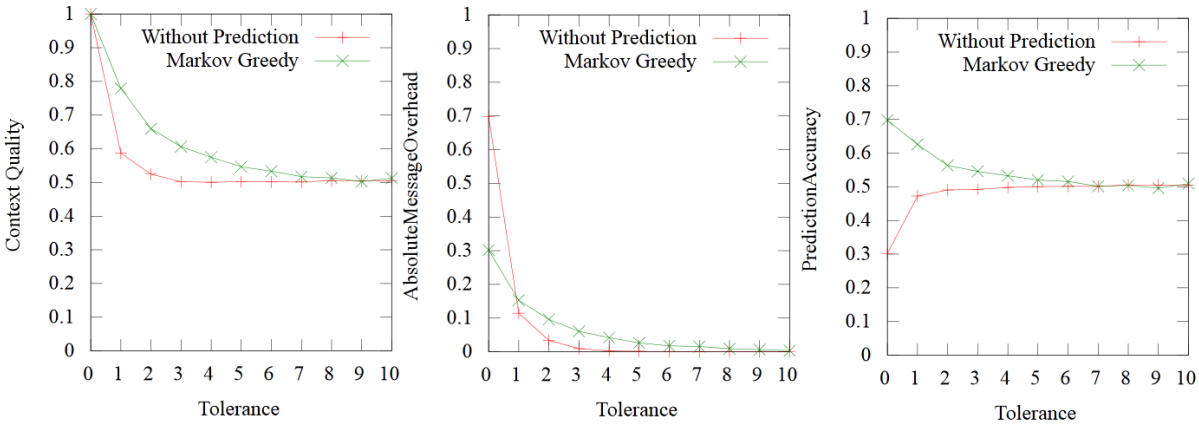

Abbildung 5.3: Versuch 2 - Markov Greedy und ohne Vorhersage, 2 Zustände, Eigentransitionswahrscheinlichkeit 0,3

Die Verbesserung, die eine einfache Vorhersage mit dem Markov Greedy Prädiktor bringt, wird in Abb. 5.3 deutlich. Während der Fall ohne Vorhersage hier eine deutlich schlechtere Kontext Qualität als in Versuch 1 ergibt, bleiben die Kurven für den Markov Greedy Prädiktor auf den Werten von Versuch 1. Das beruht darauf, dass weiterhin der wahrscheinlichste Zustand vorhergesagt wird und die Wahrscheinlichkeiten vom Wert her gleich geblieben sind und nur die Position geändert haben.

Der Markov Greedy Prädiktor bringt also eine Verbesserung, wenn die Selbsttransitionswahrscheinlichkeit unter 0,5 liegt, und ein Kontext damit meistens direkt wieder verlassen wird. Mit einer Toleranz von 1 lässt sich noch in diesem Modell mit Selbsttransitionswahrscheinlichkeit 0,3 eine Kontext Qualität von ca. 80% erreichen, bei einer geringen Erhöhung der Update-Anzahl.

Sieht man sich die Prediction Accuracy an, wird deutlich, dass der Greedy Markov Prädiktor nur in den 30% der Fälle eine falsche Voraussage macht, in denen die Selbsttransition eintritt. Tatsächlich liegt die Prediction Accuracy des Greedy Markov Prädiktors in Versuchen mit 2 Kontexten über genügend Zeitschritte immer bei der Wahrscheinlichkeit des wahrscheinlicheren Kontextes. Damit ist die Prediction Accuracy des Markov Greedy Prädiktors immer mindestens so gut, wie die ohne Vorhersage.

Markov Erwartungswert Prädiktor

Der Markov Erwartungswert Prädiktor soll nun ein komplexeres Zeitverhalten vorhersagen können. Um dieses zu modellieren wird in Versuch 3 ein Semi-Markov Modell zur Generierung des Kontext-Sets benutzt. Da somit kein Markov Modell für die Prädiktoren zur Verfügung steht, wird zunächst das Markov Modell über 10000 Zeitschritte gelernt. Dann

werden die Ergebnisse vom Markov Erwartungswert Prädiktor, Markov Greedy Prädiktor und ohne Vorhersage verglichen.

Um die Vorteile und Nachteile des Markov Erwartungswert Prädiktors zu veranschaulichen, vergleichen wie die Ergebnisse von zwei Semi-Markov Modellen:

Modell 1: Ausgangskontext c_1:

c_1	$t = 3$
c_2	1

Modell 1: Ausgangskontext c_2:

c_2	$t = 3$
c_1	1

Modell 2: Ausgangskontext c_1:

c_1	$t = 2$	$t = 5$
c_2	0,65	0,35

Modell 2: Ausgangskontext c_2:

c_2	$t = 2$	$t = 5$
c_1	0,65	0,35

Tabelle 5.4: Semi-Markov Modelle zur Evaluation des Markov Erwartungswert Prädiktors

In der Simulation wird also mit Modell 1 genau alle 3 Zeitschritte der Kontext gewechselt. Mit Modell 2 wird in beiden Ausgangs-Kontexten mit einer Wahrscheinlickeit von 0,65 im zweiten Zeitschritt nach dem Wechsel der Kontext gewechselt, mit einer Wahrscheinlichkeit von 0,35 erst im fünften Zeitschritt.

Versuchsaufbau 3

Prädiktoren:	ohne, Markov Greedy, Markov Erwartungswert
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Semi-Markovmodell aus Tabelle 5.4
Lernen:	überwacht, 10000 Zeitschritte
Zeitschritte:	10000/ Toleranz

Tabelle 5.5: Daten zum Versuch 3

Das gelernte Markov-Modell aus dem Semi-Markov Modell 1 sieht dadurch, dass genau nach jedem dritten Zustand der Kontext gewechselt wird, auf 2 Stellen gerundet wie folgt aus:

$$\begin{pmatrix} 0{,}67 & 0{,}33 \\ 0{,}33 & 0{,}67 \end{pmatrix}$$

Das gelernte Markov-Modell 2 sieht nun, ebenfalls auf 2 Stellen gerundet, genau gleich aus, resultiert aber aus einem ganz anderen Zeitverhalten. Dies wird bei der Auswertung der Prediction Accuracy sehr deutlich:

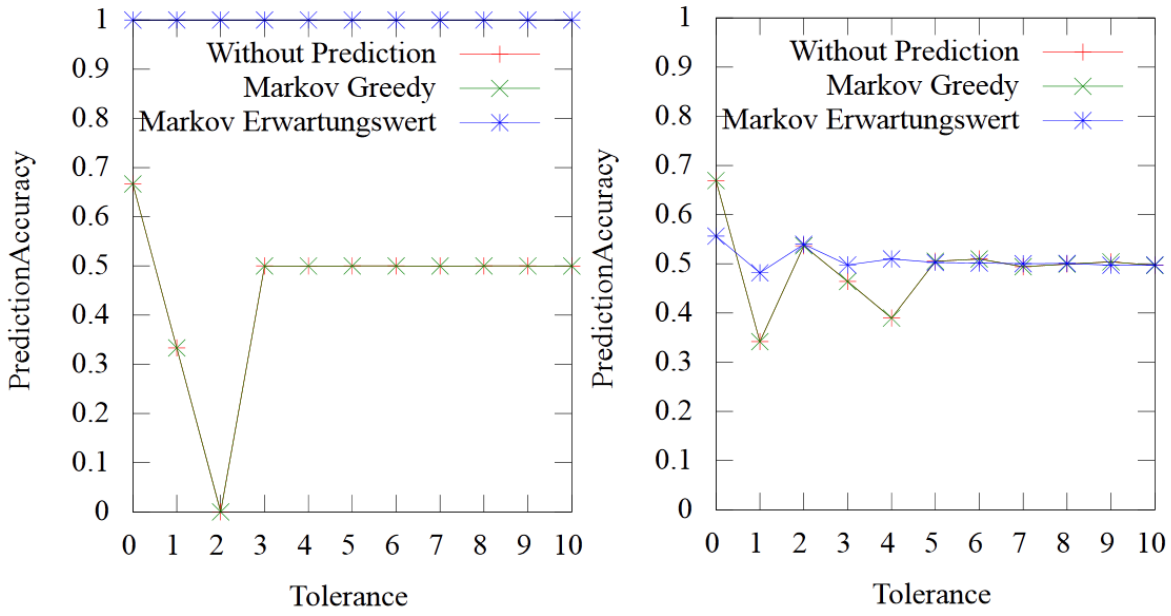

Abbildung 5.4: Ergebnisse Versuch 3 - Links: Modell 1 - Rechts: Modell 2

Die Simulation mit Semi-Markov Modell 1, das nur einen einzigen Peak besitzt, zeigt der Markov Erwartungswert Prädiktor eine Prediction Accuracy von 100%. Es wurde nicht ein einziges Kontext-Update benötigt, da das Zeitverhalten perfekt wiedergegeben wird.

In der Simulation mit Semi-Markov Modell 2 (S_2) sieht es hingegen ganz anders aus. Die beste Prediction Accuracy zeigen mit $0,67$ der Markov-Greedy Prädiktor und der Fall ohne Vorhersage bei einer Toleranz von 0. Der Markov Erwartungswert Prädiktor kommt selbst mit einer Toleranz von 0 nur auf eine Prediction Accuracy von $0,556$, was fast einem zufälligen Raten entspricht.

Der Markov Erwartungswert Prädiktor ist also nur im Falle eines einzelnen Peaks mit $P_S(c_1, c_2, d) \approx 1$ erfolgreich.

Eine kurze Erklärung bedarf noch in Modell 1 der Wert des Markov Greedy Prädiktors und des Wertes ohne Vorhersage: Mit der Selbsttransitionswahrscheinlichkeit von $0,67$ wird

ohne Kontext Update immer der aktuelle Kontext als Folgekontext vorhergesagt. Da 2 inkorrekte Vorhersagen in Folge toleriert werden, und nach jedem dritten Zeitschritt der Kontext gewechselt wird, bleibt der vorhergesagte Kontext 2 Schritte lang inkorrekt. Im dritten Schritt wird ein Kontext Update gesendet, was zwar den Kontext auf dem Client berichtigt, aber nicht als korrekte Vorhersage gewertet wird. Im vierten Schritt hat dann der Kontext auf dem Publisher wieder gewechselt und die Vorhersage ist abermals falsch.

Semi-Markov Greedy Prädiktor

Der Semi-Markov Greedy Prädiktor arbeitet auf Semi-Markov Modellen und kann daher mit komplexem Zeitverhalten wie dem, aus dem Semi-Markov Modell 2 (S_2) von oben, bessere Ergebnisse erzielen. Im Folgenden werden also die Ergebnisse des Semi-Markov Greedy Prädiktors mit Modell 2 mit denen auf einem neuen Modell 3 gegenübergestellt.

Modell 3: Ausgangskontext c_1:

c_1	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
c_2	0,15	0,15	0,15	0,15	0,4

Modell 3: Ausgangskontext c_2:

c_2	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$
c_1	0,15	0,15	0,15	0,15	0,4

Tabelle 5.6: Semi-Markov Modell 3 zur Evaluation des Semi-Markov Greedy Prädiktors

Das Semi-Markov Modell 3 zeichnet sich durch eine gleichmäßige Verteilung der Wahrscheinlichkeiten zwischen $t = 2$ und $t = 5$ aus.

Versuchsaufbau 4

Prädiktoren:	Markov Greedy, Markov Erwartungswert, Semi-Markov Greedy
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Modell 2 aus Tabelle 5.4 und Modell 3 aus Tabelle 5.6
Lernen:	überwacht, 10000 Zeitschritte
Zeitschritte:	10000/ Toleranz

Tabelle 5.7: Daten zum Versuch 4

Der Vergleichsfall ohne Prädiktor überlagert sich wieder mit dem Markov Greedy-Ergebnis und wird daher nicht dargestellt.

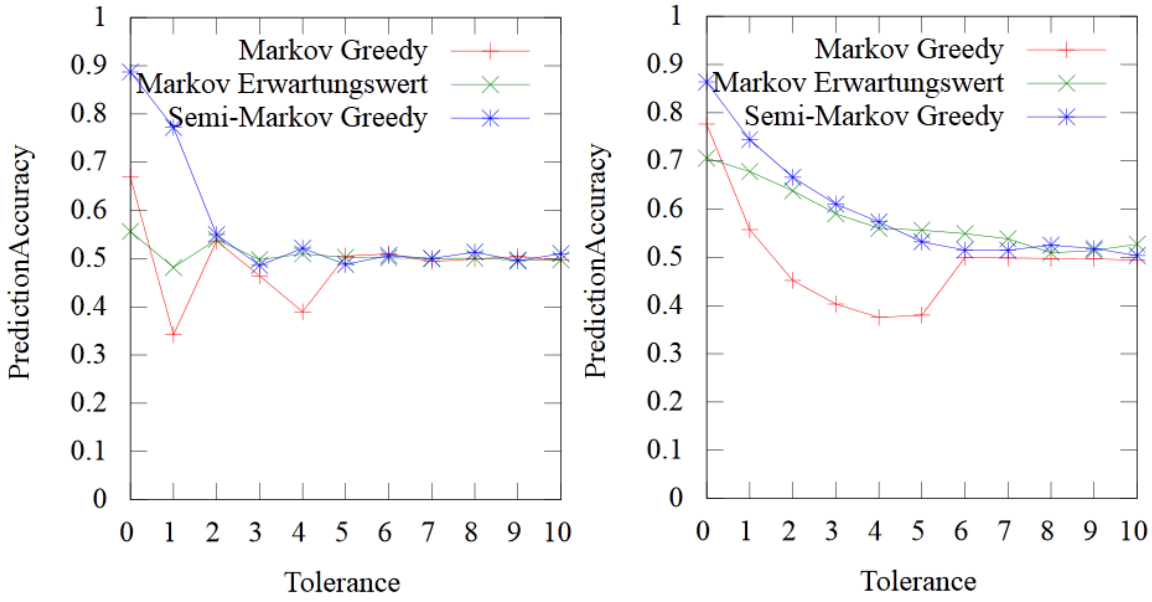

Abbildung 5.5: Ergebnisse Versuch 4 - Links: Modell 2 - Rechts: Modell 3

Die Ergebnisse in Tabelle 5.5 zeigen einen klaren Vorteil für den Semi-Markov Greedy Prädiktor. Die Vorhersage läuft folgendermaßen ab: Für beide Kontexte ist die Vorhersage analog. Beispielhaft für den Ausgangszustand c_1 ist sie $V_{SG}(S_2, c_1, 0) = (c_2, 2)$, d.h. im zweiten Zeitschritt wird in den anderen Kontext gewechselt. Diese Vorhersage tritt mit $P_S(c_1, c_2, 2) = 0,65$ ein. Im anderen Fall kommt nach Ablauf der Toleranz ein Kontext Update (c_1, d), welches $V_{SG}(S_2, c_1, d) = (c_2, 5)$, $d > 2$ liefert, da $P_S(c_1, c_2, 5) = 0,35$. Bei einer Toleranz über 2 kommt der Vorteil der zweiten, richtigen Vorhersage allerdings nicht zum tragen, da der Zeitpunkt mit $t = 5$ schon innerhalb der Toleranz erreicht ist.

Im Versuch mit Semi-Markov Modell 3 übertrifft der Semi-Markov Greedy Prädiktor ebenfalls alle sonstigen Prädiktoren. Aufgrund der geringeren maximalen Wahrscheinlichkeit bei $P_S(c_1, c_2, 6) = 0,4$ erreicht der Semi-Markov Greedy Prädiktor hier nicht ganz den Wert wie bei S_2.

Der Semi-Markov Greedy Prädiktor liefert umso bessere Ergebnisse, desto klarer sich die Wahrscheinlichkeiten zum Kontextwechsel auf einen Zeitpunkt t fokussieren.

Semi-Markov Plateau

Die Grundidee des Semi-Markov Plateau Pädiktors ist es, dass eine hohe Wahrscheinlichkeit den Kontext zu wechseln über mehrere aufeinanderfolgende Zeitschritte t gestreut sein kann. Die Summe der Wahrscheinlichkeiten $P_S(c_1, c_2, t_{start})$ bis $P_S(c_1, c_2, t_{ende})$ muss über einem Schwellenwert Sw liegen, damit ein Plateau der Breite $t_{ende} - t_{start}$ erkannt wird. Die Breite b

der erkannten Plateaus wird in den Einstellungen festgelegt. Auf die Breite geht der nächste Versuch detaillierter ein. Dieser Versuch wird auf Basis des Semi-Markov Modells 3 S_3 aus Tabelle 5.6 durchgeführt.

Versuchsaufbau 5

Prädiktoren:	M Greedy, M Erwartungswert, SM Greedy, SM Plateau $(b = 3, Sw = 0,5)$
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Modell 3 aus Tabelle 5.6
Lernen:	überwacht, 10000 Zeitschritte
Zeitschritte:	10000/ Toleranz

Tabelle 5.8: Daten zum Versuch 5

Die Breite des zu findenden Plateaus wurde auf 3 festgelegt, der Schwellwert auf $0,5$ um sicher zu gehen, dass wir ein Plateau finden, in dem ein Wechsel wahrscheinlicher als außerhalb des Plateaus ist.

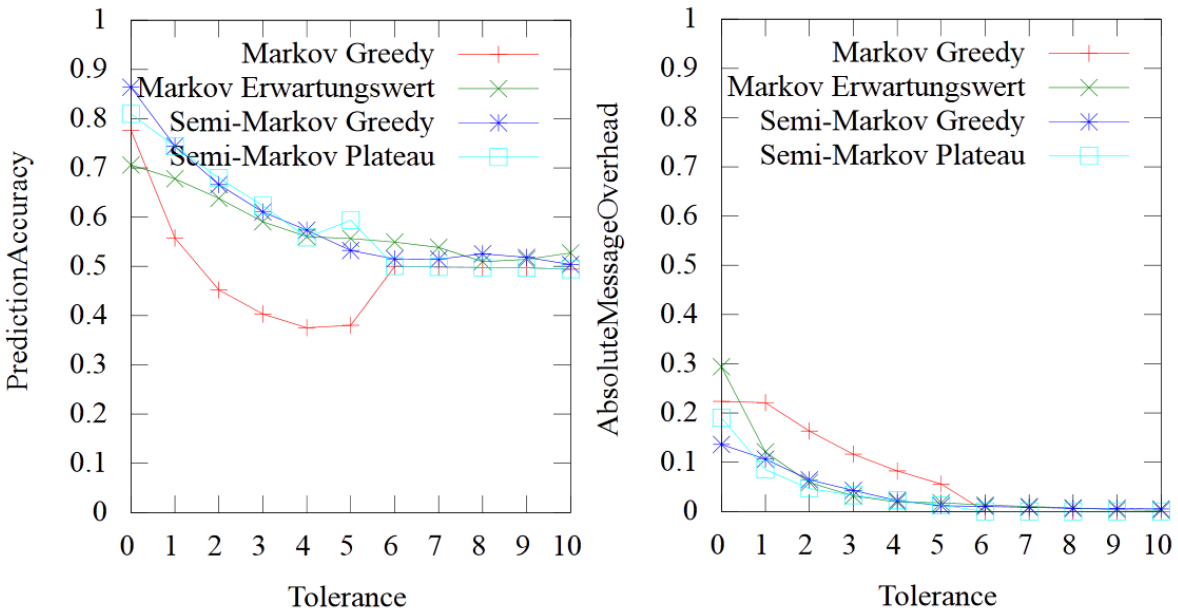

Abbildung 5.6: Ergebnisse Versuch 5 Prediction Accuracy und Absolute Message Overhead

Die Vorhersage des Semi-Markov Greedy Prädiktors ist, beispielhaft von Ausgangskontext c_1 ist $V_{SG}(S_3, c_1, 0) = (c_2, 6)$, die Vorhersage des Semi-Markov Plateau Prädiktors ist $V_{SG}(S_3, c_1, 0) = (c_2, 5)$. Ein Plateau der Breite 3 um den Wert $t = 5$ ergibt eine Gesamtwahrscheinlichkeit des Plateaus von $P(c_1, c_2, 4) + P(c_1, c_2, 5) + P(c_1, c_2, 6) = 0,15 + 0,15 + 0,4 =$

0,7 ergibt. Eine Breite von 3 wird bei einem Threshold von 1 optimal ausgenutzt. Bei einem Threshold von 2 läuft auch der Fall mit $t = 3$ ohne Kontext Update ab.

So sehen wir auch in den Ergebnissen in Abbildung 5.6, dass der Semi-Markov Plateau Prädiktor bei einer Toleranz von 0 schlechter abschneidet als der Semi-Markov Greedy Prädiktor. Das liegt daran, dass direkt der vorhergesagte Zeitschritt 5 für den Wechsel nur die Wahrscheinlichkeit 0,15 hat. Bei einer Toleranz von 1 liefert der Semi-Markov Plateau Prädiktor die gleiche Prediction Accuracy wie der Semi-Markov Greedy Prädiktor, bei weniger Kontext Updates. Bei einer Toleranz von 2 liefert er sogar bessere Ergebnisse mit weniger Kontext Updates.

Als nächstes wird die Auswirkung der Plateau-Breite untersucht. Modell ist wieder S_3 aus Tabelle 5.6. Nun wird die Breite des Plateaus nacheinander auf 1, 2, 3 und 4 gesetzt. In Abbildung 5.7 sind nur die Breiten 1 und 4 aufgeführt, da die Breiten 2 und 3 beide das in Abbildung 5.6 dargestellte Diagramm liefern.

Versuchsaufbau 6

Prädiktoren:	M Greedy, M Erwartungswert, SM Greedy,
	SM Plateau $(b = 1,4, Sw = 0,5)$
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Modell 3 aus Tabelle 5.6
Lernen:	überwacht, 10000 Zeitschritte
Zeitschritte:	10000/ Toleranz

Tabelle 5.9: Daten zum Versuch 6

Ein Plateau der Breite 1 wird also nicht gefunden. Dadurch verhält sich die Vorhersage als sei kein Prädiktor vorhanden. Dies wird an der Überlagerung mit der Linie des Markov Greedy Prädiktors deutlich. Die oben verwendete Plateau-Breite von 3 scheint optimal zu sein und ist zu einer Breite von 2 Equivalent, da die Mitte des Plateaus aufgerundet in beiden Fällen bei $t = 2$ liegt. Eine Plateau-Breite von 4 ist schlechter als eine Breite von 3, da dann die ersten Zeitschritte $t \in [2,5]$ mit der Summe 0,6 über den Schwellenwert von 0,5 kommen und als Wechselzeitschritt $t = 4$ vorhergesagt wird. Die dann nötige Toleranz resultiert dann in einer niedrigen Prediction Accuracy.

Der Semi-Markov Plateau Prädiktor ist also erfolgreich, wenn eine kleine Toleranz zugunsten einer Einsparung von Kontext Updates in Kauf genommen wird und im gelernten Semi-Markov Modell überhaupt ein Plateau vorhanden ist.

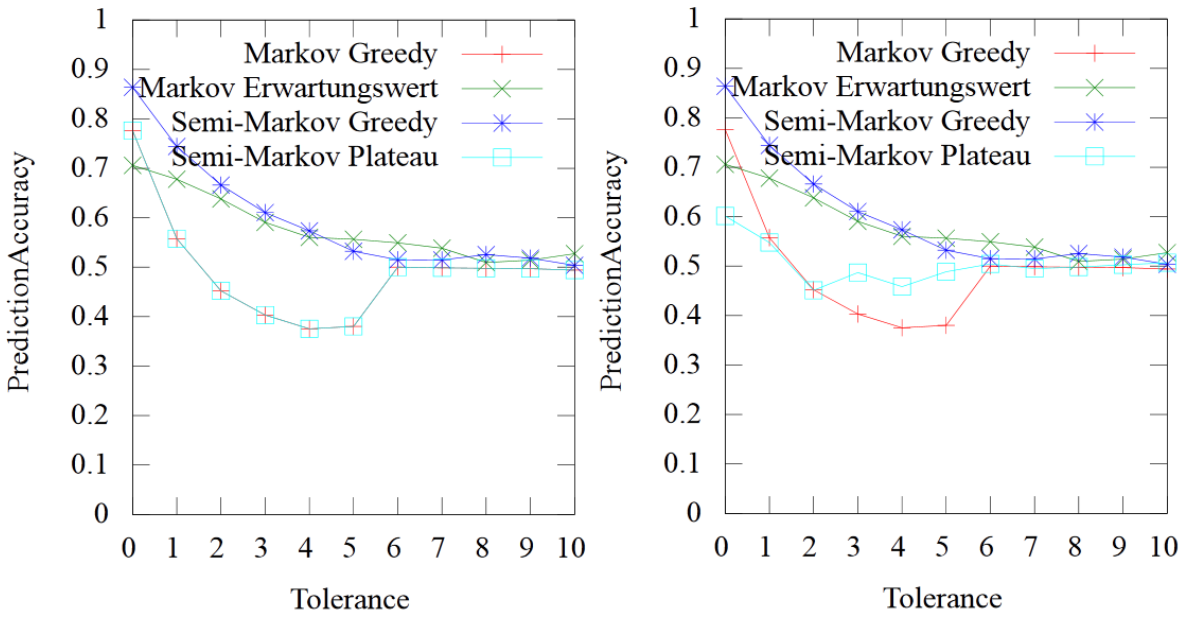

Abbildung 5.7: Ergebnisse Versuch 6 Prediction Accuracy bei Plateau-Breiten von 1 (links) und 4(rechts)

Semi-Markov Erwartungswert

Der Semi-Markov Erwartungswert Pädiktor ist für die Vorhersage zum Zeitpunkt $d = 0$ Equivalent zum Markov Erwartungswert Prädiktor. Darum soll in folgendem Versuch 7 ein Beispiel ausgewertet werden, in dem der Unterschied der beiden Erwartungwert Prädiktoren deutlich zum Tragen kommt.

Das Semi-Markov Modell für den Versuch hat für beide Ausgangskontexte als Wahrscheinlichkeitsverteilung für den Kontext-Wechsel 2 vereinzelte, auseinanderstehende Peaks bei 2 und 7:

Beide Erwartungswert-Prädiktoren werden nun direkt verglichen, als Referenz dient der Semi-Markov Greedy Prädiktor, der in diesem Modell 4 die Besten Ergebnisse liefert:

Versuchsaufbau 7

Für die Auswertung spielen sowohl die Kontext-Qualität als auch die Anzahl der Kontext Updates eine Rolle.

Wie man sehen kann, ist der Semi-Markov Erwartungswert Prädiktor nur in der Stelle mit Toleranz 1 besser als der normale Markov Erwartungswert Prädiktor, da er da von der Neuberechung ohne den ersten Peak profitieren kann.

Modell 4: Ausgangskontext c_1:

c_1	$t = 2$	$t = 7$
c_2	0,5	0,5

Modell 4: Ausgangskontext c_2:

c_2	$t = 2$	$t = 7$
c_1	0,5	0,5

Tabelle 5.10: Semi-Markov Modell 4 zur Evaluation des Semi-Markov Erwartungswert Prädiktors

Prädiktoren:	Markov Erwartungswert, Semi-Markov Erwartungswert, SM Greedy
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Modell 4 aus Tabelle 5.10
Lernen:	überwacht, 10000 Zeitschritte
Zeitschritte:	10000/ Toleranz

Tabelle 5.11: Daten zum Versuch 7

Jedoch werden beide Erwartungswert Prädiktoren vom Semi-Markov Greedy Prädiktor übertroffen, der in diesem Versuch auch unter den anderen Prädiktoren die Beste Prediction Accuracy liefert. Allerding liegt auch die Prediction Accuracy des Semi-Markov Plateau Prädiktors mit Modell 4 über den Ergebnissen der Erwartungswert Prädiktoren.

Der Semi-Markov Erwartungswert Prädiktor verwirft durch die Bildung des gewichteten Durchschnittes wichtige Zeitinformationen aus dem Semi-Markov Modell. Daher sind ihm die beiden anderen Semi-Markov Prädiktoren überlegen. Er findet daher keine produktive Anwendung.

5.2.3 Lernen

Als letzter wichtiger Punkt auf Basis von künstlichen Modellen wird nun das Lernverhalten bei dynamischem Lernen betrachtet. Dazu soll während einer Simulation ein bekanntes Modell gelernt werden und dabei der Zeitverlauf der Prediction Accuracy untersucht werden. Im weiteren werden die Kontexte auf dem Publisher auf ein anderes Modell umgestellt um das Adaptionsverhalten des Lernens zu überprüfen.

Das Ursprungsmodell ist ein Semi-Markov Modell, bei dem in jedem Zeitschritt der Zustand gewechselt wird. Das zweite Modell, das nach Lernschritt 4 als Driver benutzt wird, ist ebenfalls ein Semi-Markov Modell. Bei diesem wird allerdings immer nur in jedem vierten

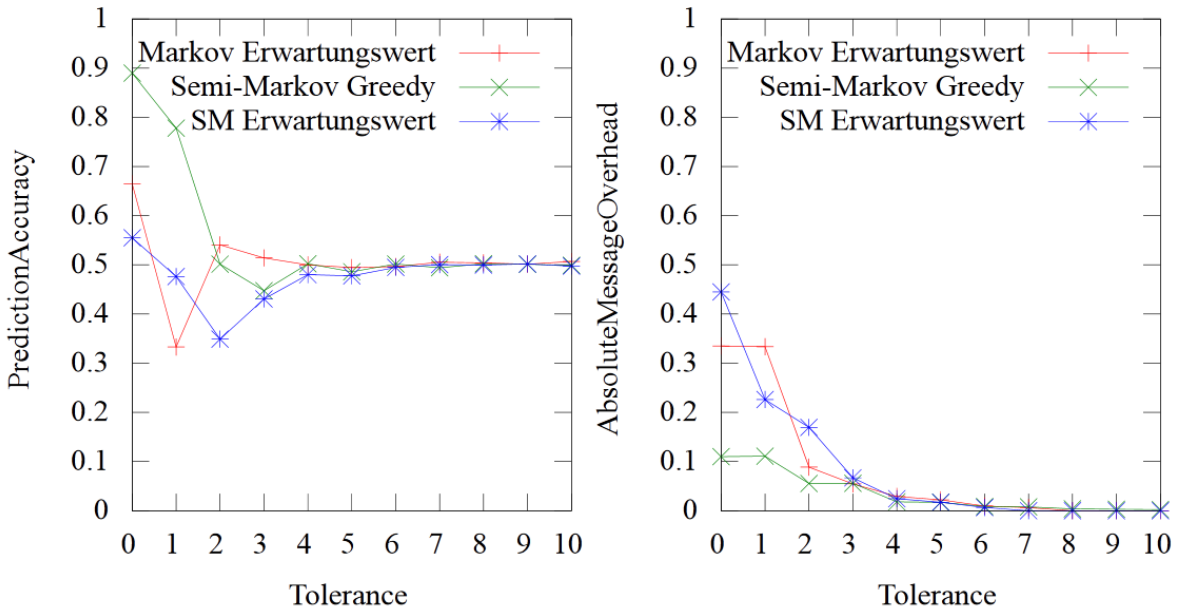

Abbildung 5.8: Ergebnisse Versuch 7 Markov Erwartungswert und Semi-Markov Erwartungswert

Zeitschritt der Zustand gewechselt. Diese Modelle sind für die Prädiktoren alle entweder mit einer Prediction Accuracy von entweder 0 oder 1 vorhersagbar. Im Diagramm 5.9 sind nach rechts die Lernschritte aufgetragen. Dort kann man gut erkennen, wann die aktuellen Modelle sich angepasst haben. Solange sich das gelernte Modell nicht semantisch vom aktuellen Modell unterscheidet, muss es auch im Lernschritt nicht als Modell Update übertragen werden.

Versuchsaufbau 8

Prädiktoren:	ohne, M Greedy, SM Greedy, SM Plateau
Threshold:	0
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Wechsel nach einem Schritt, nach 4 Schritten
Lernen:	dynamisch, alle 6 Zeitschritte mit Lernfaktor 0,1
Zeitschritte:	6 Zeitschritte pro Lernschritt

Tabelle 5.12: Daten zum Versuch 8

Da die für den Driver verwendeten Semi-Markov Modelle wenig komplex sind, passt sich das gelernte Modell für die Prädiktoren innerhalb weniger Lernschritte an. Eine genaue

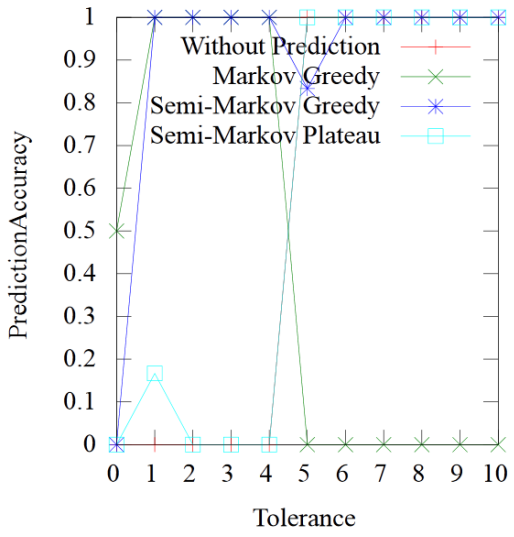

Abbildung 5.9: Ergebnisse Versuch 8 Lernen

Einstellung der Lernparameter hängt vom verwendeten Modell ab. Ein Lernschritt sollte mindestens einen kompletten Zyklus dauern. Hier wäre das ein Sprung von c_1 nach c_2 und zurück. Bei Anpassung an das menschliche Verhalten wäre ein Zyklus z.B. ein Tag oder eine Woche.

5.3 Ergebnisse auf Basis von Real-Welt Daten

Nachdem nun die grundsätzlichen Stärken und Schwächen der Prädiktoren untersucht wurden, werden diese im Folgenden mit echten Kontext-Historien untersucht. Dazu wurden 2 Datensätze verwendet. Einmal eine beispielhafte CenseMe-Historie auf den Daten von [MPF+10]. Diese wurden auf über 20 Smartphones gesammelt, welche Studenten und wissenschaftliche Mitarbeiter mit sich herum trugen.

Der zweite verwendete Datensatz “Wang” besteht aus mehreren Sets aus 2 diskreten Kontexten, die mit dem Beschleunigungssensor eines Smartphones aufgenommen wurden und unterscheiden ob der Träger in Bewegung ist oder nicht.

5.3.1 CenseMe

Die CenseMe Daten lassen sich auf `http:\www.crawdad.org` nach Anmeldung kostenfrei herunterladen. Sie enthalten unter anderem einen abgeleiteten diskreten Kontext. Zur Evaluation wird hier daraus der Datensatz 4 verwendet. Bei näherer Untersuchung stellt sich

heraus, dass nur genau 4 der diskreten Kontexte in der Kontext Historie erreicht werden: Die Kontexte mit den Nummern 0, 1, 2 und 5.

Da ein ausgeschriebenes Semi-Markov Modell der Kontextübergänge aus dieser Kontext-Historie den Rahmen sprengen würde, sind die Übergangswahrscheinlichkeiten für die verschiedenen Ausgangszustände in Abbildung 5.10 visualisiert.

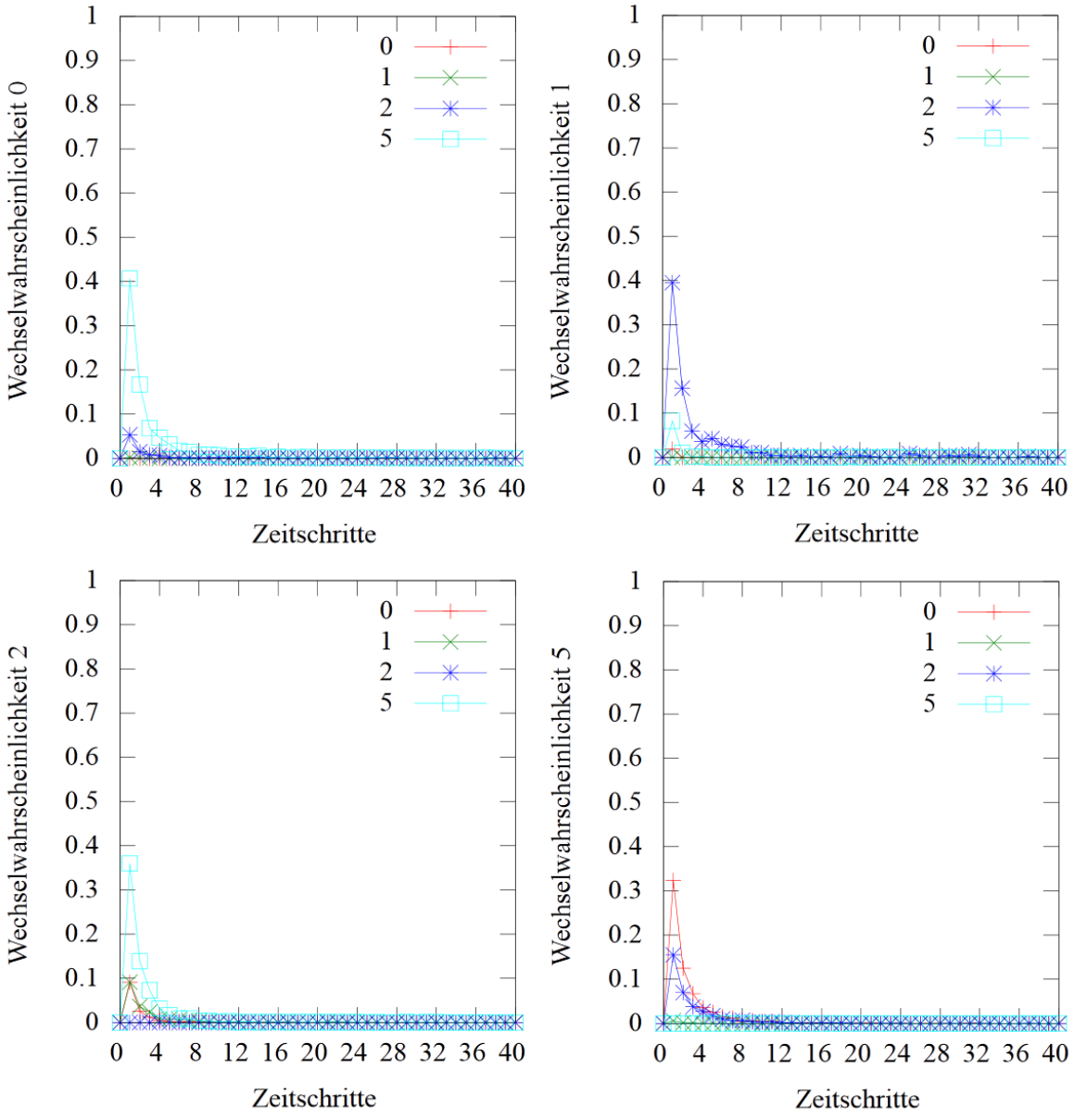

Abbildung 5.10: CenseMe - Wahrscheinlichkeiten der Übergänge pro Zeitschritt

Für alle Kontexte gibt es also einen mit Abstand wahrscheinlichsten Folgekontext, zu dem mit großer Wahrscheinlichkeit in den ersten 3 Zeitschritten gewechselt wird. Allerdings gibt es auch lange Stecken ganz ohne Kontextwechsel, die zwar seltener sind, aber dennoch einen spürbaren Anteil an allen Zeitschritten ausmachen.

Versuchsaufbau 9

Prädiktoren:	M Greedy, M Erwartungswert, SM Greedy, SM Plateau ($b = 1$,$Sw = 0,5$)
Threshold:	0
Anzahl der Kontexte in C:	4
Herkunft des Test-Sets:	CenceMe 4
Lernen:	links: statisch im Voraus über benutztes Teil-Set rechts: dynamisch, Intervall 5000, Faktor 0,5
Zeitschritte:	5000 × 10 Zeitschritte

Tabelle 5.13: Daten zum Versuch 9

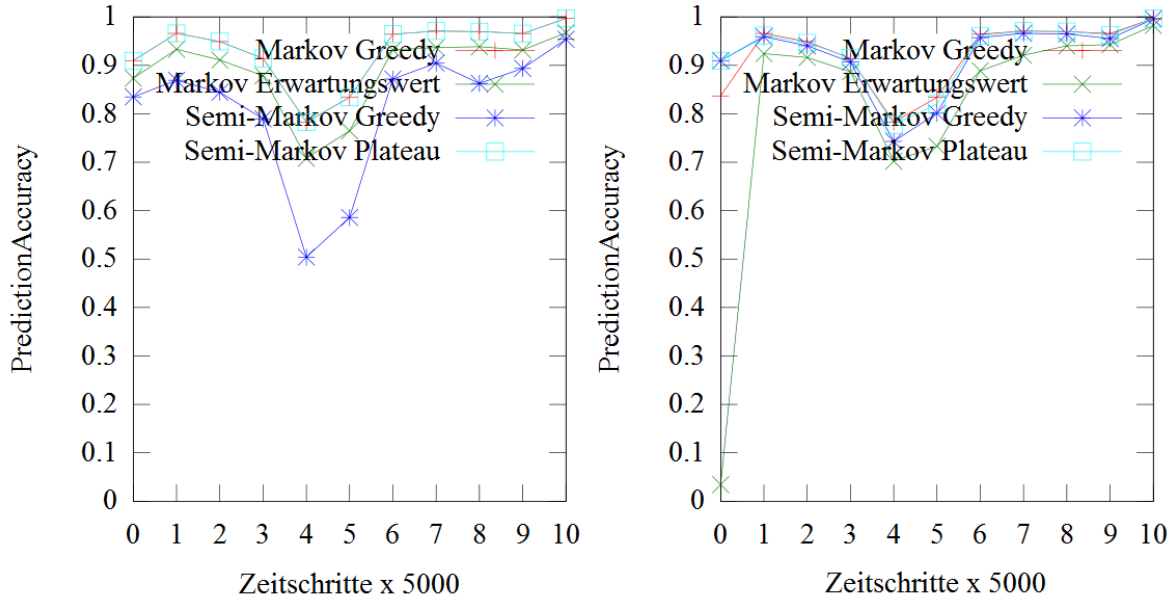

Abbildung 5.11: CenseMe - Prediction Accuracy für statisches Lernen (links) und dynamisches Lernen(rechts)

Durch die kurze Verweildauer pro Kontext erzielt hier der Greedy Markov Prädiktor sehr gute Ergebnisse. Der Unterschied zwischen statischem und dynamischem Lernen beeinflusst

vor allem den Semi-Markov Greedy Prädiktor. Die Kontext Historie lässt sich insgesamt sehr gut vorhersagen. Interessant ist der Zeitschritt an der Stelle 4×5000. Dort weichen die Kontext-Folgen vom Durchschnitt ab, was vor allem den auf den richtigen Peak angewiesenen Semi-Markov Greedy Prädiktor im statisch gelernten Fall stark beeinflusst. Beim dynmischen Lernen kann die Abweichung deutlich besser kompensiert werden. Der Markov Erwartungswert Prädiktor schneidet insgesamt am schlechtesten ab.

Im Verhältnis von Energieaufwand und Prediction Accuracy ist in diesem Fall der Ressourcen sparende Markov Greedy Prädiktor zu empfehlen.

Einen interessanten Einblick gibt das Lernen mit Lernschritten verschiedener Länge, die stark gewichtet werden. In einer Simulation analog zu Versuch 9 wurden die Lernschritte auf 50, 100 und 5000 gesetzt und ein Lernfaktor von 0,5 eingestellt. Es lässt sich feststellen, dass ein Lernschritt von 100 Zeitschritten von der Prediction Accuracy her wenig Unterschied zum einem Lernschritt von 5000 Zeitschritten macht. Bei einem kurzen Lerninterval von 50 Zeitschritten wird der sonst empfehlenswerte Markov Greedy Prädiktor zum schlechtesten Prädiktor. In der Praxis macht ein so kurzer Lernschritt jedoch keinen Sinn, da um davon zu profitieren, jedesmal ein teures Modell Update versendet werden müsste.

Das Aufkommen an Kontext Updates in Relation zur Anzahl der Kontextwechsel kann bei ungefähr gleich bleibender Prediction Accuracy mit einer höheren Toleranz von 3 je nach Prädiktor um 70-80% reduziert werden.

5.3.2 Wang

Die Sets wurden von Wang von der University of Southern California's Autonomous Networks Research Group, http://anrg.usc.edu für das Paper [WKZA10] aufgezeichnet. Sie lassen sich auf http://anrg.usc.edu/www/index.php/Downloads unter Punkt 14 herunterladen.

Es wird beispielhaft die Datei "acc_motion_record0925.txt" untersucht, welche eine Kontext Historie mit 2210 Einträgen umfasst.

Die Verteilung der Wahrscheinlichkeit der Kontextübergänge über die Zeitschritte ist in Abb 5.12 dargestellt.

Es ergibt sich ein ähnliches Bild wie bei den CenceMe Daten. Ein Wechsel in den Folgekontext im ersten Zeitschritt ist am wahrscheinlichsten, aber es treten auch viele lange Phasen ohne Kontextwechsel auf.

Das Setup für die Simulation ist ähnlich wie in Versuch 9.

Versuchsaufbau 10

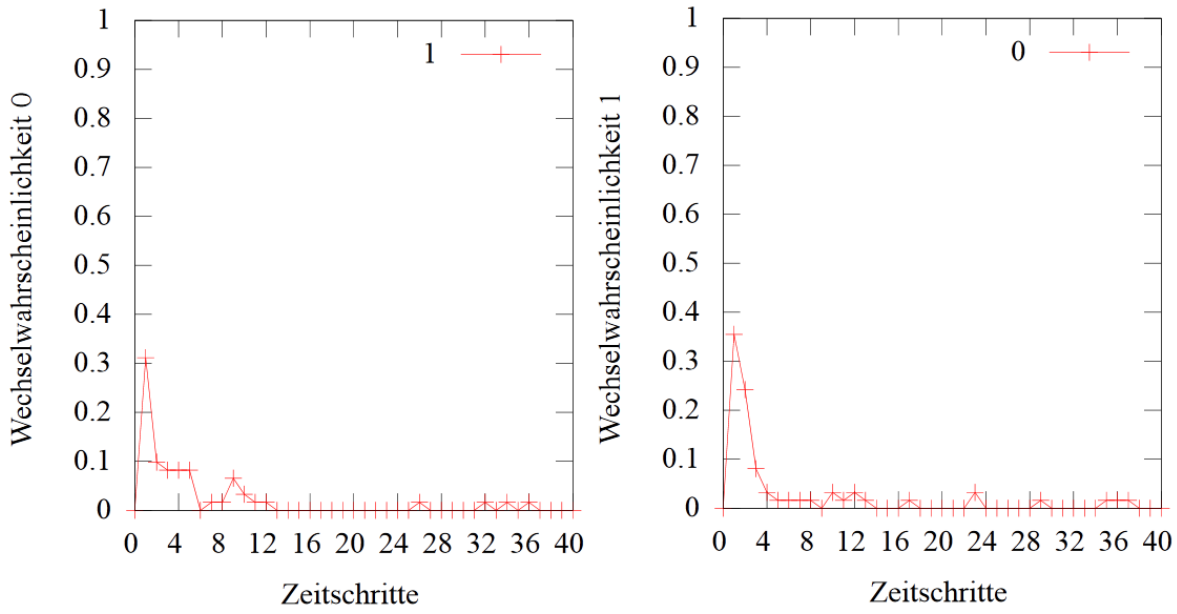

Abbildung 5.12: Wang - Wahrscheinlichkeiten der Übergänge pro Zeitschritt

Prädiktoren:	M Greedy, M Erwartungswert, SM Greedy, SM Plateau ($b = 3$,$Sw = 0,5$)
Threshold:	0
Anzahl der Kontexte in C:	2
Herkunft des Test-Sets:	Wang, "acc_motion_record0925.txt"
Lernen:	links: statisch im Voraus über benutztes Teil-Set rechts: dynamisch, Intervall 150, Faktor 0,5
Zeitschritte:	150×10 Zeitschritte

Tabelle 5.14: Daten zum Versuch 10

Die Auswertung von Versuch 10 bringt folgende, in Abbildung 5.13 dargestellten Ergebnisse:

In der Simulation mit dem statisch gelernten Modell liefern der Semi-Markov Greedy und der Semi-Markov Plateau Prädiktor beide sehr gute Ergebnisse. Der Markov Greedy Prädiktor performed Abschnittsweise sehr gut, zeigt aber ab Zeitschritt $8x150$ deutliche Einbußen bei der Prediction Accuracy. Diese sind darauf zurückzuführen, dass eine lange Serie von Kontext 0 unterbrochen wird und sich wieder mit Kontext 1 abwechselt. Da das gelernte Markov-Modell an dieser Stelle immer die Selbsttransition in 0 vorsieht, liefert der Greedy Markov hier die schlechten Ergebnisse.

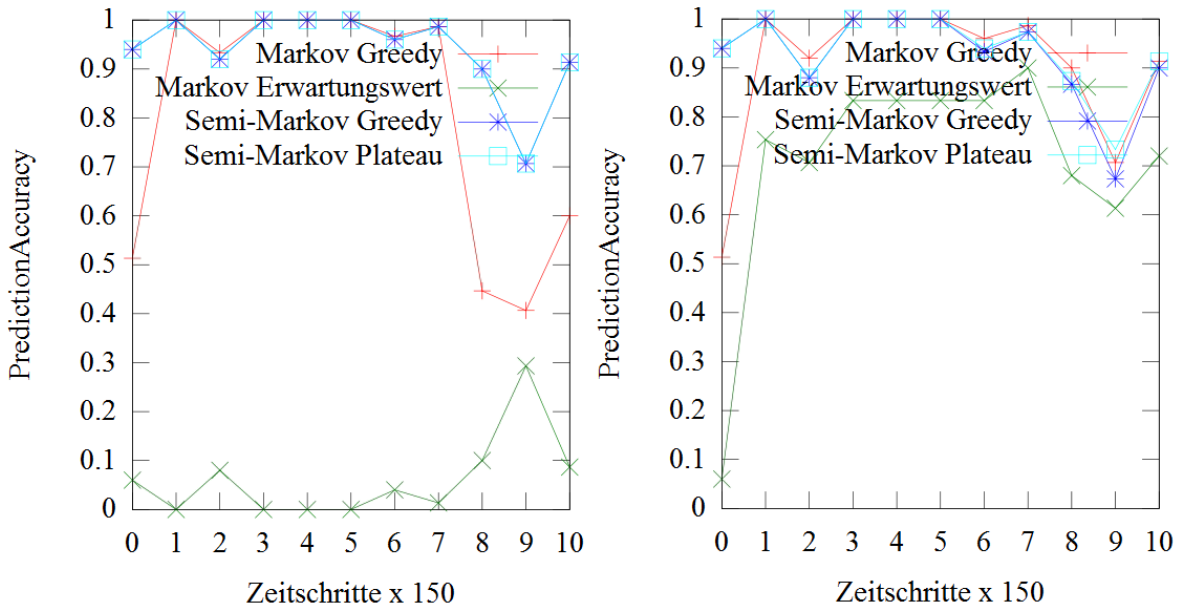

Abbildung 5.13: Wang - Prediction Accuracy für statisches Lernen (links) und dynamisches Lernen(rechts)

Anders stellen sich die Ergebnisse beim dynamischen Lernen dar. Hier liegt der Greedy Markov Prädiktor bis zum Zeitschritt $8x150$ auf den Werten des statisch gelernten Modelles, während der Semi-Markov Greedy und der Semi-Markov Plateau etwas darunter liegen. Durch das dynamische Lernen passt sich der Markov Greedy Prädiktor auch nach dem Zeitschritt $8x150$ noch an und liefert hier damit insgesamt die beste Prediction Accuracy.

Bei einer Toleranz größer als 0 fallen die Kurven für Prediction Accuracy der Markov Greedy, Semi-Markov Greedy und Semi-Markov Plateau Prädiktoren auf eine Kurve zusammen, jedoch lassen sich z.B. mit einer Toleranz von 3 bei annähernd gleicher Prediction Accuracy ein Großteil der Kontext Updates sparen.

6 Zusammenfassung und Ausblick

Das gute Abschneiden des Markov Greedy Prädiktors in den Echtwelt-Daten legt nahe, dass sich der Aufwand zur Berechung der komplizierteren Semi-Markov Prädiktoren zumindest in den untersuchten Fällen nicht lohnt. Dies liegt jedoch an der Struktur der ausgewerteten Datensätze, die kein regelmäßiges, komplexes Zeitverhalten aufweisen.

Da der Semi-Markov Greedy und der Semi-Markov Plateau Prädiktor auch in diesen ungünstigen Fällen etwa gleich gut sind wie der Greedy Markov Prädiktor, ist es denkbar dann einen von diesen Prädiktoren zu wählen, wenn ein regelmäßiges, komplexes Zeitverhalten der Kontexte auftreten könnte.

Grundsätzlich ist die Verwendung des Semi-Markov Greedy Prädiktors erfolgversprechend, wenn der Zeitverlauf der Wahrscheinlichkeiten für den Kontextwechsel einen deutlichen, schmalen Peak aufweist. Der Plateau-Prädiktor sucht nicht nur nach dem Maximum, sondern findet auch breitere Häufungen der Wahrscheinlichkeiten für einen Kontextwechsel.

Die Prädiktoren auf Basis des Erwartungswertes sind nicht zu empfehlen, da sie in den Fällen, in denen sie nicht genau zutreffen, deutlich schlechter als die anderen Prädiktoren sind.

Offen bleibt hier noch die Frage nach dem optimalen Sampling-Intervall, der das Benutzerverhalten bei guter Vorhersagbarkeit möglichst gut wieder gibt. Weiterhin wäre zu erforschen, ob aus vielen Einzelfaktoren zusammengesetzte Kontexte durch die Vorhersage der einzelnen Faktoren eine bessere Voraussagbarkeit erreichen.

Literaturverzeichnis

[BCW90] T. C. Bell, J. G. Cleary, I. H. Witten. Text Compression. Technical report, Prentice Hall Advanced Reference Series, 1990. (Zitiert auf Seite 10)

[BEYY04] R. Begleiter, R. El-Yaniv, G. Yona. On Prediction Using Variable Order Markov Models. *Journal of Artificial Inteligence Research*, 22:385–421, 2004. URL http://www.aaai.org/Papers/JAIR/Vol22/JAIR-2212.pdf. (Zitiert auf Seite 10)

[CW84] J. Cleary, I. Witten. Data compression using adaptive coding and partial string matching. In *IEEE Transactions on Communications*. 1984. (Zitiert auf Seite 10)

[DAS99] A. K. Dey, G. D. Abowd, D. Salber. A context-based infrastructure for smart environments. In *Managing interactions in smart environments: 1st International Workshop on Managing Interactions in Smart Environments, Dublin, December 1999*, pp. 167–168. 1st International Workshop on Managing Interactions in Smart Environments, Dublin, 1999. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.4814&rep=rep1&type=pdf. (Zitiert auf Seite 9)

[Kato6] J.-P. Katoen. *Discrete-time Markov chains, Lecture 2 of Probabilistic Models for Concurrency*. RWTH Aachen, 2006. URL http://www-i2.informatik.rwth-aachen.de/Teaching/Course/PMC/2006/pmc_lec2.pdf. (Zitiert auf Seite 17)

[KM09] D. Katsaros, Y. Manolopoulos. Prediction in Wireless Networks by Markov Chains. Technical report, Computer & Communication Engineering Department, University of Thessaly, Volos, Greece Informatics Department, Aristotle University, Thessaloniki, Greece, 2009. URL http://delab.csd.auth.gr/papers/IEEEWComm09km.pdf. (Zitiert auf Seite 10)

[May05] R. Mayrhofer. Context Prediction based on Context Histories: Expected Benefits, Issues and Current State-of-the-Art. Technical report, University Linz, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.5252&rep=rep1&type=pdf. (Zitiert auf Seite 11)

[MPF+10] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, A. Campbell. Supporting Energy-Efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones. In P. Floréen, A. Krüger, M. Spasojevic, editors, *Pervasive Computing*, volume 6030 of *Lecture Notes in Computer Science*, pp. 355–372. Springer Berlin

/ Heidelberg, 2010. URL http://dx.doi.org/10.1007/978-3-642-12654-3_21. 10.1007/978-3-642-12654-321. (*Zitiert auf den Seiten* 10 *und* 44)

[MRF04] R. Mayrhofer, H. Radi, A. Ferscha. Recognizing and predicting context by learning from user behavior. In *special issue on Advances in Mobile Computing*. 2004. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.2388&rep=rep1&type=pdf. (Zitiert auf den Seiten 9 und 13)

[RL00] K. Rothermel, A. Leonhardi. A Comparison of Protocols for Updating Location Information. Technical report, IPVR, Universität Stuttgart, 2000. (Zitiert auf den Seiten 9, 11 und 15)

[RST96] D. Ron, Y. Singer, N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. Technical report, Institute of Computer Science, Hebrew University Jerusalem, 1996. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.1845&rep=rep1&type=pdf. (Zitiert auf Seite 10)

[SKJH03] L. Song, D. Kotz, R. Jain, X. He. Evaluating location predictors with extensive Wi-Fi mobility data. Technical report, Dartmouth College, DoCoMo USA Labs, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.11.848&rep=rep1&type=pdf. (Zitiert auf Seite 11)

[WKZA10] Y. Wang, B. Krishnamachari, Q. Zhao, M. Annavaram. Markov-optimal Sensing Policy for User State Estimation in Mobile Devices. Technical report, Department of Electrical Engineering, University of Southern California, Los Angeles, USA, 2010. URL http://anrg.usc.edu/www/index.php/Downloads. (Zitiert auf Seite 47)

[WS95] F. Willems, Y. Shtarkov. The context-tree weighting method: Basic properties. In *IEEE Information Theory*. 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.352&rep=rep1&type=pdf. (Zitiert auf Seite 10)

[WSCY99] O. Wolfson, A. P. Sistla, S. Chamberlain, Y. Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7:257–387, 1999. URL http://dx.doi.org/10.1023/A:1008782710752. 10.1023/A:1008782710752. (Zitiert auf Seite 10)

[ZL78] J. Ziv, A. Lempel. Compression of individual sequences via variable-rate coding. In *IEEE TRANSACTIONS ON INFORMATION THEORY*, volume 24. 1978. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.2892&rep=rep1&type=pdf. (Zitiert auf Seite 10)

Alle URLs wurden zuletzt am 24.08.2011 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Tobias Fink)