[SHSt07] Johannes Schmitt, Matthias Hollick, Ralf Steinmetz; Der Assistent im Hintergrund: Adaptives Kommunikationsmanagement durch Lernen vom Nutzer; PIK II/2007 - Current Trends in Network and Service Management, April 2007

Der Assistent im Hintergrund: Adaptives Kommunikationsmanagement durch Lernen vom Nutzer

Johannes Schmitt, Matthias Hollick und Ralf Steinmetz
Multimedia Communications (KOM)
Technische Universität Darmstadt
Merckstr. 25, D-64283 Darmstadt
Telefon: +49 (0) 6151 164451
Fax: +49 (0) 6151 166152
Email: {johannes.schmitt,matthias.hollick,ralf.steinmetz}@KOM.tu-darmstadt.de

***Zusammenfassung*— Der Mensch ist von einer Flut an digitalen Informationen umgeben. Die zielgerichtete Auswertung dieser Informationsmenge verspricht Unterstützung im täglichen Leben. Ziel unserer Arbeit ist eine Erweiterung der bisher vorwiegend manuellen Auswertung beim Umgang mit diesen hochdynamisch wechselnden und heterogenen Informationsstrukturen. Hierzu werden in dieser Arbeit die Grundprinzipien für einen Dienst erarbeitet, der es ermöglicht die Regeln zur maschinell-unterstützten Auswertung des Kontextes eines Nutzers anhand des Nutzerverhaltens zu lernen, anzupassen und anzuwenden. Die Umsetzung dieses „Kontext-Dienstes" erfolgt in einem Szenario zur adaptiven Steuerung von Abläufen in einem Kommunikationssystem. In dieser Arbeit wird die entworfene Architektur erläutert. Die Rahmenbedingungen sowie Besonderheiten, die sich bei der Umsetzung der Architektur und der Auswahl der Verfahren zur Regelerstellung ergeben, werden aufgezeigt. Für das ausgewählte Szenario des adaptiven Kommunikationsmanagement werden Verfahren zur Regelerstellung ausgewählt, verglichen und optimiert.**

I. Einleitung

Der Alltag vieler Menschen ist heute geprägt durch Technologien, die es ermöglichen (fast) immer und überall erreichbar zu sein. Zugehörige Endgeräte weisen zudem eine Vielzahl von Funktionalitäten und Konfigurationsmöglichkeiten auf. Diese beiden Eigenschaften sind jedoch für den Menschen nicht immer von Vorteil. Aus der Möglichkeit immer erreichbar zu sein, entsteht immer mehr eine Notwendigkeit. Damit erhöht sich auch die Zahl unerwünschter oder nicht zielführender Anrufe. Ebenso verfügen heutige Technologien über eine Vielzahl von Funktionalitäten, die nie oder nur selten zum Einsatz kommen, da die Handhabung zu kompliziert, die Einrichtung zu aufwendig oder das ständige Aktualisieren der Einstellungen zu umständlich ist.

Auf der anderen Seite umgibt den Menschen heute eine Fülle von digitalen Informationen. Diese Informationen können Aufschluss über die Situation geben, in der er sich gerade befindet (seinen „Kontext" - nach der Definition von Dey [Dey00]). Durch sinnvolle Nutzung vorhandener Informationen können dem Nutzer Entscheidungen abgenommen und Aktionen für ihn ausgeführt werden. Grundlagen zur Kontext-Nutzung wurden hierzu in [GASS04] erarbeitet. Verwandte Arbeiten finden sich unter anderem in [FMR03] und [EHJ02].

Wir haben den Fokus darauf gerichtet, den Menschen in seinen täglichen Arbeiten durch einen „virtuellen Assistenten" zu unterstützen. Der virtuelle Assistent wird in Form eines Dienstes mit der Bezeichnung „Kontext-Dienst" umgesetzt.

In Kapitel II erläutern wir ein typisches Anwendungsszenario für unseren Kontext-Dienst: Das adaptive Kommunikationsmanagement. In Kapitel III erarbeiten wir die Grundprinzipien unseres Kontext-Dienstes. Die zur Realisierung benötigten Feedback-Mechanismen, sowie sich hieraus ableitende Aufgaben und Rahmenbedingungen, werden in Kapitel IV und V identifiziert. Kapitel VI beschreibt die Umsetzung der vorgeschlagenen Architektur im Rahmen unseres Szenarios. Welche Eigenschaften Lernverfahren besitzen müssen, um zur Bestimmung von Kontexten geeignet zu sein, wird dabei untersucht und die notwendigen Anpassungen werden erläutert. Kapitel VII fasst unsere Arbeit zusammen.

II. Anwendungsszenario

Der Kontext-Dienst kann in sehr unterschiedlichen Umgebungen und in verschiedenen Ausprägungen zum Einsatz kommen. Allgemein gesehen dient er zur Auswertung von Informationen in dynamischen und heterogenen Umgebungen. In dieser Arbeit zeigen wir die Architektur und Funktionsweise unseres Ansatzes generisch auf. Zur Veranschaulichung und als Machbarkeitsstudie dient herbei das folgende Anwendungsszenario:

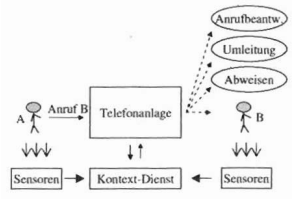

Abb. 1. Einsatzszenario des Kontext-Dienstes

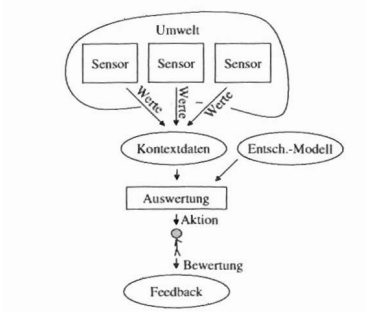

Abb. 2. Phase 1: Auswertung

Adaptives Kommunikationsmanagement. In diesem Szenario soll der Kontext-Dienst in der Lage sein, die Verarbeitung von eingehenden Anrufen und E-Mails zu steuern. Jedoch nicht wie bisher durch Regeln oder Skripte, sondern abhängig von dem Kontext der Anfrage, dem Kontext des Nutzers und den Entscheidungen des Nutzers in vergleichbaren, vorangegangenen Situationen.

Der Kontext-Dienst wurde als Prototyp in eine Siemens Telefonanlage [SIE] integriert und ist dort in der Lage die Kommunikationsdienste der Anlage (wie z.B. Anrufweiterleitung, Anrufabweisung oder Notifikation) zu steuern und dynamisch auf eingehende Anrufe anzuwenden (sie Abbildung 1). Zur Entscheidungsfindung erhält der Kontext-Dienst in diesem Szenario bisher vor allem anrufbezogene Informationen, kann aber auch um zusätzliche Informationsquellen erweitert werden. Um den Kontext einer Person zu bestimmen, sind in dem beschriebenen Szenario Informationen wie der Aufenthaltsort, der Terminkalender oder das Adressbuch von zentralem Interesse. Durch Zugriff auf Daten dieser Art (z.B. von Outlook oder einem Mobiltelefon) lassen sich sehr gute Aussagen treffen.

Eine Schnittstelle über das der Nutzer Feedback an den Kontext-Dienst geben kann wurde in das Web-Frontend der Telefonanlage integriert und kann durch eine Java/J2ME Applikation auch auf mobilen Endgeräten durchgeführt werden.

III. Grundprinzipien des Kontext-Dienstes

Der Kontext-Dienst kann Entscheidungen treffen indem verfügbare Informationen aus der Umwelt (durch Sensoren erfasste Werte $\{x\}$ die die Umwelt X repräsentieren) herangezogen und ausgewertet werden. Die Auswertung wiederum wird durch ein Entscheidungsmodell durchgeführt, welches dynamisch angepasst werden kann.

Die Auswertung von Informationen geschieht meist durch Regelwerke oder Skripte (siehe auch [GASS04]). Solche manuell zu erstellen ist eine hoch komplexe Aufgabe und daher für den normalen Anwender oft nicht geeignet. Insbesondere wenn die Menge der zu verarbeitenden Informationen dynamisch variiert, ist die Verarbeitung durch statische Modelle nicht sinnvoll. Der Kontext-Dienst erstellt daher das Entscheidungsmodell automatisch [WF01], indem er vom Nutzer lernt bzw. ihn beobachtet und bestimmt, von welchen Informationen das Verhalten des Nutzers abhängig ist.

Die Funktionen des Dienstes lassen sich in zwei Phasen unterteilen: Die Auswertungsphase und die Adaptionsphase.

Phase 1 - Auswertung: In dieser Phase wird der Kontext-Dienst genutzt um eine Entscheidung zu treffen (siehe Abbildung 2). Hierzu erhält der Kontext-Dienst die Menge an auszuwertenden Informationen X und das Entscheidungsmodell $f()$, welches das Verhalten bzw. den Wunsch des Nutzers $g()$ näherungsweise abbildet. Das bedeutet, dass das Modell nicht nur $f(X) = g(X)$ in möglichst vielen Fällen als Ergebnis erhalten soll, sondern auch $f(X') = g(X')$ für neue und bisher unbekannte Situationen X'. Das Ergebnis dieser Auswertung kann zur Steuerung bzw. zum Auslösen von Aktionen genutzt werden. Die Reaktion des Nutzers auf die gewählte Aktion (sein „Feedback") entspricht $g(X)$ für die zu einer Situation gesammelten Informationen X. Sie kann anschließend genutzt werden, um das Entscheidungsmodell anzupassen. Je nach Anwendungsgebiet des Kontext-Dienstes kann das Feedback implizit (durch Beobachten der Reaktion des Nutzers) oder explizit erfolgen (indem der Nutzer das

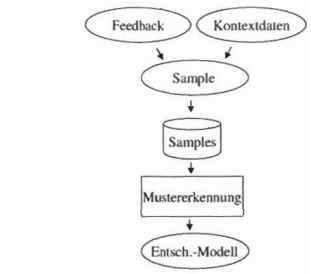

Abb. 3. Phase 2: Adaption

korrigierte Ergebnis selbst auswählt - siehe Kapitel IV).

Phase 2 - Adaption: Wird zu einer Auswertung aus Phase 1 ein Feedback vom Nutzer erhalten, so kann in dieser Phase eine Anpassung des Entscheidungsmodells erfolgen (siehe hierzu Abbildung 3). Der Wunsch des Nutzers zu einer bestimmten Situation lässt sich abbilden durch die Menge der in dieser Situation gesammelten Informationen X in Kombination mit dem Feedback des Nutzers $g(X)$. Das Tupel $s = (g(X), X)$ welches sich daraus ergibt, repräsentiert diesen Wunsch des Nutzers und wird als „Sample" bezeichnet. Die Menge der Samples eines Nutzers spiegelt das Wissen des Dienstes über den Nutzer wider. In Phase 1 muss die Auswertung größerer Mengen von Samples in eingeschränktem Zeitrahmen durchgeführt werden können. Dazu ist es notwendig, im Voraus die Abhängigkeiten, die durch die Samples beschrieben werden zu finden und zu extrahieren. Hierzu wird anschließend ein Lernverfahren angewendet, welches auf diesen Daten $\{s\} = S$ eine Mustererkennung durchführt und die gefundenen Abhängigkeiten in Form eines neuen Modells $f'()$ darstellt. Das neu erstellte Modell wird schließlich als Grundlage für die nächste Auswertung in Phase 1 verwendet.

Beide Phasen zusammengenommen ergeben wiederum einen Kreislauf. Durch Nutzung des Dienstes und Feedback des Nutzers kann dem Dienst das gewünschte Verhalten antrainiert werden.

Das vorgestellte Prinzip erlaubt das Steuern des Kontext-Dienstes durch einfaches Feedback des Nutzers. Statt das Modell $f()$ manuell zu erzeugen, reicht Feedback aus, um komplexe Verhaltensweisen nachzubilden. Dabei besteht ein Feedback aus dem korrigierten Ergebnis $g(X)$, also in der Regel nur der Spezifikation des Wertes, welcher in der entsprechenden Situation als Ergebnis vom Nutzer erwünscht ist (beispielsweise Kontext=„Büro" oder Aktion=„Anrufweiterleitung").

IV. Feedback

Die Lernverfahren, die für diesen Anwendungsbereich in Frage kommen, gehören zur Klasse der „überwachten" Lernverfahren. Das bedeutet, dass Feedback notwendig ist, um das Verfahren zu trainieren. Dieses Feedback bildet dabei das einzige Element, über das der Nutzer in Kontakt mit dem Kontext-Dienst tritt, wobei Feedback primär das korrigierte Ergebnis beinhaltet.

Die Umsetzung der Methode, über die der Kontext-Dienst Feedback erhält, ist vom Einsatzszenario abhängig. Dabei kann zwischen zwei Typen von Feedback unterschieden werden:

Implizites Feedback - Erfolgt als direkte Reaktion des Nutzers auf die vom Kontext-Dienst getroffene Entscheidung. In den meisten Szenarien sind dabei jedoch nur begrenzt Eingabemöglichkeiten für das Feedback vorhanden. In einem Szenario, in welchem der Dienst beispielsweise genutzt wird, um ein Gerät zu steuern, indem er es an/aus schalten kann, kann das manuelle Betätigen des Schalters als Feedback genutzt werden. Bei dem Telefonie-Szenario können auch Reaktionen wie „Anruf-Abweisen" oder „Nicht-Annehmen" als Feedback verwendet werden.

Explizites Feedback - Diese Variante ermöglicht die Erstellung von Feedback über eine „Historie". Die Historie gibt Einblick in zurückliegende Entscheidungen des Systems, und der Nutzer kann nachträglich zu diesen Entscheidungen Feedback geben. Diese Art von Feedback erlaubt einen größeren Funktionsumfang, erfordert aber auch, dass der Nutzer über den Zugriff auf entsprechende Geräte verfügt, um die dafür notwendigen Funktionalitäten umzusetzen (beispielsweise grafische Oberfläche oder Spracherkennung).

V. Aufgaben und Rahmenbedingungen

Für die Umsetzung des vorgestellten Dienstes ergeben sich eine Reihe von Rahmenbedingungen:

- *Minimale Interaktion mit dem Nutzer.* Der Nutzer soll bei seinen Arbeiten unterstützt werden. Das bedeutet durch den Einsatz des Dienstes soll der Aufwand seitens des Nutzers reduziert werden.
- *Korrektheit der Ergebnisse.* Falsche Entscheidungen erzeugen unerwünschte Aktionen. Das Modell muss so nahe wie möglich das Verhalten des Nutzers abbilden.
- *Dynamik der Informationsquellen.* Neue Informationsquellen und deren Formate müssen von dem System gefunden und angewendet werden können. Ebenso muss mit nicht verfügbaren Informationsquellen gerechnet werden.
- *Konzept-Drift.* Veränderungen des Verhaltens eines Nutzers müssen schnell erkannt und vom Modell adaptiert werden (siehe Kapitel VI-C).
- *Falsche Angaben.* Bei der Interaktion mit Menschen („Human-in-the-Loop" Systemen) muss mit falschen Feedback-Angaben gerechnet werden. Ebenso können Informationsquellen falsche, veraltete oder widersprüchliche Daten liefern.
- *Aufwand - Auswertung.* Es ist damit zu rechnen, dass eine Auswertung eines Modells vergleichsweise häufiger vorkommt, als dessen Neuerstellung. Daher sollte der Aufwand in den Bereich Modellerstellung verlagert werden. Je nach Anwendungsgebiet des Kontext-Dienstes ist die Auswertung des Modells mehr oder weniger zeitkritisch. Im Falle des beschriebenen Kommunikationszenarios gelten hier sogar Echtzeitkriterien. Zudem muss zur Auswertung des Modells auch der Aufwand für die Informationsgewinnung gezählt werden.
- *Aufwand - Erstellung.* Der Aufwand bei der Erstellung ist weniger zeitkritisch, da keine direkte Interaktion mit dem Nutzer erfolgt. Auf der anderen Seite ist es möglich, dass ein Nutzer nach dem Geben eines Feedback testet, ob sich das System adaptiert hat.

Für den Einsatz eines solchen Dienstes müssen diese Bedingungen je nach Szenario unterschiedlich gewichtet werden, und es können noch weitere Bedingungen hinzukommen. Die meisten dieser Bedingungen betreffen das eingesetzte Verfahren zur Erstellung des Entscheidungsmodells bzw. die Eigenschaften des Modells an sich (siehe VI-B). Durch die Auswahl und den Einsatz eines geeigneten Lernverfahrens können diese Bedingungen eingehalten werden.

VI. Umsetzung des Kontext-Dienstes

Die zuvor beschriebene Architektur des Kontext-Dienstes zur Auswertung von Informationen, wurde wie in Abbildung 4 gezeigt umgesetzt.

Von außen ist der Kontext-Dienst als Web-Service über zwei Schnittstellen ansprechbar. Eine Schnittstelle implementiert die Funktionen für die Anfrage zur Auswertung von Informationen. Eine weitere Schnittstelle stellt dem Nutzer die Feedback-Funktionalitäten zur Verfügung, mit welchen der Dienst gesteuert werden kann. Bei der Umsetzung des Dienstes ergeben sich eine Reihe weiterer interessanter Aspekte, die in den folgenden Punkten erläutert werden.

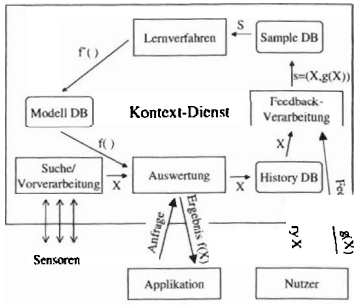

Abb. 4. Umsetzung des Kontext-Dienstes

A. Sensor Suche und Vorverarbeitung

Korrekte Entscheidungen können nur dann getroffen werden, wenn (mindestens) eine relevante bzw. ausschlaggebende Informationsquelle verfügbar ist. Dazu ist es hilfreich, auf möglichst viele Informationsquellen Zugriff zu haben. Es ist notwendig, dass die Informationsquellen Meta-Informationen anbieten, um zu bestimmen, welche Informationen relevant sind (d.h. welche in Relation mit dem Nutzer stehen) und wie sie verwendet werden können. Gerade in dynamisch wechselnden Umgebungen ist die Suche nach relevanten Sensoren keine triviale Aufgabe.

Beim Einsatz von Lernverfahren spielt die Vorverarbeitung der Daten eine wichtige Rolle:

- *Werte vergleichbar machen.* Um Entscheidungsmodelle auszuwerten, müssen die darauf angewendeten Werte mit denjenigen vergleichbar sein, mit denen das Modell erstellt wurde. An dieser Stelle gilt es anhand der Meta-Information zu bestimmen, welche Werte vergleichbar sind oder welche durch Umformatierung vergleichbar gemacht werden können.
- *Aggregieren von Werten.* Durch Aggregation von Informationsquellen können höherwertige Aussagen getroffen werden (umfangreiche Arbeiten hierzu sind in [Man05] zu finden).
- *Ungültige Werte filtern.* Um bessere Ergebnisse zu erzielen sollten veraltete, ungenaue oder ungültige Werte gefiltert werden.
- *Nicht vorhandene Sensorwerte ergänzen.* Je nach eingesetztem Typ von Entscheidungsmodell können Probleme entstehen, wenn für Entscheidungen heranzuziehende Informationen nicht verfügbar sind. Eine mögliche Methode um gegen dieses Problem anzugehen, ist das Interpolieren der fehlenden Werte durch vergleichbare Werte aus anderen Samples.

B. Eingesetzte Lernverfahren und Modelle

Bei der Umsetzung des Kontext-Dienstes stehen eine Reihe von Lernverfahren zur Auswahl. Jeder Typ von Lernverfahren bringt sein eigenes Modell mit sich. Die bekanntesten Lernverfahren sind diejenigen zur Erstellung von Support-Vektor-Maschinen, Neuronalen Netzen, Bayesschen Netze, Naive Bayes oder Entscheidungsbäumen [Das02]. Jedes Lernverfahren und dessen Modell hat andere Eigenschaften im Bezug auf Aufwand zur Modellerstellung bzw. Modellauswertung, Robustheit, Speicherbedarf oder der verwendbaren Datenformate. Welches Verfahren am besten geeignet ist, ist stark abhängig von den Rahmenbedingungen des jeweiligen Einsatzszenarios. Es existieren noch eine Reihe spezieller Eigenschaften, welche gerade in der Umsetzung dieses Dienstes von Bedeutung sind:

- *Erweiterbarkeit*. Manche Modelle (z.B. Naive Bayes) können um ein Sample erweitert werden, ohne dass das gesamte Modell neu berechnet werden muss.
- *Backtracing*. Diese Eigenschaft eines Modells (z.B. Entscheidungsbäume) erlaubt das Nachvollziehen, von welchen Sensoren eine Entscheidung abhängig war. Diese Sensoren können anschließend genutzt werden um bei einer Zustandsänderung aktiv eine neue Überprüfung des aktuellen Kontextes anzustoßen und davon abhängig Aktionen auszulösen.
- *Online – Offline*. Die Lernverfahren können in die Klassen „Online" und „Offline" eingeteilt werden (näheres hierzu in [Sar04]). Klassische Lernverfahren sind „Offline" Lernverfahren. Das bedeutet, dass alle Samples, unabhängig von ihrem Alter gleichwertig behandelt werden. Solange auf ein stabiles Konzept $g()$ hingearbeitet werden soll (z.B. bei Spam-Filtern), ist dies auch ausreichend. Wenn das Konzept (in diesem Fall das Verhalten des Nutzers) Änderungen unterworfen ist (z.B. Wechsel der Gewohnheiten, Arbeitsplatzwechsel), findet ein so genannter „Konzept-Drift" statt und der Dienst muss sich schnell dem neuen Verhalten anpassen können (näheres hierzu in Kapitel VI-C).

C. Konzept-Drift Verarbeitung

Wenn sich das Ziel-Konzept ändert ($g() \rightarrow g'()$) entsteht ein so genannter „Konzept-Drift" [Tsy04]. In diesem Fall muss das angewendete Verfahren möglichst schnell in der Lage sein, sich an das neue Konzept anzupassen.

Die Samples in der Sample-Datenbank beschreiben das Wissen des Kontext-Dienstes über den Nutzer. Das grundlegende Problem besteht darin, dass nach einem Konzept-Drift zwar neue Samples s' erstellt werden, die dem neuen Konzept $s' = (X, g'(X))$ entsprechen. Jedoch befinden sich in der Menge der Samples S noch „ungültig" gewordene Samples, die das alte Konzept $g()$ beschreiben. Das bedeutet, ohne die Verarbeitung von Konzept-Drift entscheidet der Dienst solange zugunsten des Konzeptes $g()$, bis mehr Samples des neuen Konzeptes vorhanden sind ($|s'| > |s|$) und entsprechend viele Feedback-Aktionen des Nutzers durchgeführt wurden. Offline-Verfahren unterstützen die Behandlung von Konzept-Drift nicht von sich aus.

Meta-Verfahren - Bestehenden Ansätze in dem Bereich sehen vor, die Offline Verfahren in so genannte Meta-Verfahren zu kapseln (MetaL - Verfahren). Das Meta-Verfahren ergänzt dabei das eigentliche Verfahren um Fenster- oder Gewichtungstechniken [Wid97], [MM00]. Fenstertechniken lassen alte Samples ausscheiden, da nur Samples innerhalb eines Fensters zur Modellerstellung herangezogen werden. Erweiterungen bei den Fenstertechniken erlauben die Variation der Fenstergröße abhängig von der Stabilität des Konzeptes. Gewichtungstechniken erlauben das „Altern" von Werten. Ältere Werte werden dabei schwächer gewichtet als neuere. Die Gewichte werden anschließend bei der Modellerstellung berücksichtigt. Beide Techniken lassen sich auch kombinieren, führen aber nur bedingt zum Ziel. Da alte Samples durchaus noch gültig und relevant sein können, ist mit diesen Techniken das eigentliche Problem, ungültige Samples zu entfernen, noch nicht gelöst.

Online-Lernverfahren - Lernverfahren dieser Klasse wurden mit dem Fokus auf die Verarbeitung von sich dynamisch erweiternden bzw. ändernden Konzepten entworfen. Die bekanntesten Verfahren sind STAGGER, DARLING und die FLORA Familie. Online-Lernverfahren sind bisher nur selten im Einsatz - daher sind Implementierungen und Erfahrungswerte aus realistischen Einsatzszenarien rar. Wieso sich FLORA für die Umsetzung des Dienstes in dem in Kapitel II vorgestellten Anwendungsszenarios sehr gut eignet, wird im folgenden Teil erläutert.

FLORA - Von FLORA wurden bisher vier Versionen (FLORA 1-4) vorgestellt und mit der letzten Version FLORA 4 am weitesten entwickelt. Der Name FLORA steht als Akronym für FLOating Rough Approximation und basiert auf dem von Miroslav Kubat im Jahre 1989 entworfenen System [WK96]. Das Lernverfahren basiert auf logischen Regeln, welche dynamisch zwischen Regelmengen unterschiedlicher Aussagen verschoben werden können. Zudem verfügt FLORA über die in Kapitel VI-B beschriebenen Eigenschaften der Erweiterbarkeit und des Backtracings. Durch Verschieben einzelner Regeln kann das Verhalten des Modells schnell angepasst werden.

Ein weiteres Problem von Konzept-Drift ist, dass es in der Anfangsphase nur schwer von einem „falschen" Feedback unterschieden werden kann. Das „Langzeitgedächtnis" von FLORA 4 kann dieses Problems lösen. Eine neue Aussage des Nutzers wird vom Modell sofort übernommen, das alte Konzept bleibt aber als Backup gespeichert. Sollten nachfolgende Aussagen besser dem alten Konzept entsprechen, kann zu diesem zurückgesprungen werden.

Der in der Literatur vorhandene Ansatz von FLORA verfügt nur über die Funktionalität, binär zu klassifizieren und diskrete Werte zu verarbeiten. Im Zuge der Umsetzung des Dienstes wurde der Ansatz von FLORA erweitert und als WEKA-kompatibles [Wai] Lernverfahren mit der Bezeichnung „Flora-MC" (FLORA-Multiclassification) implementiert. FLORA-MC verfügt als Erweiterung sowohl über eine zusätzliche Strategie der dynamischen Fensterverwaltung, als auch über die Möglichkeit, kontinuierliche Wertebereiche zu verarbeiten und Konzepte mit beliebig vielen Zielsituationen abzubilden.

VII. Zusammenfassung und Ausblick

Es wurden die Grundprinzipien, die Architektur und die Rahmenbedingungen für einen Kontext-Dienst aufgezeigt, der es ermöglicht die Auswertung des Kontextes eines Nutzers an dessen Verhalten anzupassen. Ebenso wurde die Umsetzung dieses Ansatzes und deren Besonderheiten in einem Szenario zur adaptiven Steuerung von Abläufen in einem Kommunikationssystem durchgeführt und erläutert. Im Rahmen dieser Arbeit wurde eine Reihe von unterschiedlichen Lernverfahren durch Simulationen unter realistischen Bedingungen auf die angesprochenen Rahmenbedingungen getestet. Im Bereich der Performance und Genauigkeit erzielten die meisten Verfahren akzeptable Ergebnisse, so dass hier durchaus eine Auswahl an verschiedenen Lernverfahren besteht.

Der Kontext-Dienst wird aktuell im Zusammenspiel mit der Siemens HiPath [SIE] Kommunikationsplattform getestet, wobei die Feedback-Funktionalitäten dem Nutzer als zusätzliches Web-Frontend im Siemens-ComAssistant zur Verfügung gestellt werden. Der Kontext-Dienst kann dort als Alternative zur manuellen Regel-Erstellung genutzt werden und hat das Potenzial auch weitere Aufgaben im Bereich des Kommunikationsmanagements zu übernehmen.

Weitere Arbeiten in diesem Bereich sehen die Nutzung von Backtracing-Eigenschaften zur aktiven Steuerung von anderen Diensten vor. Als weiterer Aspekt zur Ergänzung des Gesamtsystems wird auch die Suche nach geeigneten Sensoren bzw. Informationsquellen in dynamischen und heterogen Umgebungen im Fokus weiterer Arbeiten stehen. Um zum Gesamtziel - dem virtuellen Assistenten - zu gelangen, wird der Kontext-Dienst auch zur Steuerung in weiteren Anwendungsbereichen und im Zusammenspiel mit anderen Applikationen herangezogen und getestet werden.

Literaturverzeichnis

[Das02] Belur V. Dasarathy. Handbook of Data Mining and Knowledge Discovery, 2002.

[Dey00] Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, 2000.

[EHJ02] Carl M. Kadie Eric Horvitz, Paul Koch and Andy Jacobs. Probabilistic Forecasting of Presence and Availability, 2002.

[FMR03] Alois Ferscha, Rene Mayrhofer, and Harald Radi. Recognizing and predicting context by learning from user behavior. In *Proceedings of the 1st International Conference on Advances in Mobile Multimedia*, volume 171, pages 25–35, 2003.

[GASS04] Manuel Goertz, Ralf Ackermann, Johannes Schmitt, and Ralf Steinmetz. Context-aware communication services: A framework for building enhanced ip telephony services. In *International Conference on Computer Communications and Networks ICCCN 2004*, pages 535–540, October 2004.

[Man05] **Manuel** Görtz. *Effiziente Echtzeit-Kommunikationsdienste durch Einbeziehung von Kontexten*. PhD thesis, Technische Universität Darmstadt, 2005.

[MM00] Marcus A. Maloof and Ryszard S. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41(1):27–52, 2000.

[Sar04] Warren Sarle. What are batch, incremental, on-line, offline, June 2004.

[SIE] Siemens HiPath-Kommunikationsplattform. `http://www.siemens.de/hipath`.

[Tsy04] Alexey Tsymbal. The problem of concept drift: Definitions and related work, April 2004.

[Wai] Waikato University. Weka data mining software. `http://www.cs.waikato.ac.nz/~ml/weka/`.

[WF01] Ian H. Witten and Eibe Frank. *Data Mining - Praktische Werkzeuge und Techniken für das maschinelle Lernen*. Carl Hanser Verlag, München, Wien, 2001.

[Wid97] Gerhard Widmer. Tracking context changes through meta-learning. *Machine Learning*, 27(3):259–286, June 1997.

[WK96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, April 1996.