Hochschule für Angewandte Wissenschaften Hamburg

Hamburg University of Applied Sciences

Seminar Ringvorlesung im Masterstudiengang

Jens Ellenberg

Klassifizierung von Kontext in einer intelligenten Wohnung

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Nach der Vision von Mark Weiser [25] sind im 21. Jahrhundert Computer überall und sie unterstützen den Menschen bei seinen Aktivitäten. Ein grundlegendes Konzept bei dieser Vision ist die Fähigkeit der Computer, auf ihre Umgebung zu reagieren. Dafür sind verschiedene Sensoren nötig, welche die relevanten Informationen erfassen. Die erfassten Informationen müssen dann interpretiert werden, damit sie anschließend von den verschiedenen Programmen genutzt werden können. Dieser Teilbereich der Informatik nennt sich „Context Aware Computing“ [21]. Ein Programm, welches sein Verhalten dem Context anpasst, ist beispielsweise der „Wecker 2.0“ [9] [10]. Der Wecker ist ein Programm von vielen, die im „iFlat“ [23] (einer intelligenten Wohnung) die aktuelle Situation (oder den Kontext)[1] nutzen. Für diese Programme fehlt im „iFlat“ bisher diese Kontexterkennung.

1.2 Ziel der Masterarbeit

Das Ziel dieser Masterarbeit ist es, ein Framework zu entwickeln, mit dem der aktuelle Kontext in einer intelligenten Wohnung erkannt werden kann. Dieser Kontext soll für andere Programme nutzbar sein. Es gibt nahezu unendlich viele Möglichkeiten Situationen in einer Wohnung zu definieren. Jede dieser Situationen könnten für ein bestimmtes Programm wichtig sein und sollte daher von dem Framework erkannt werden. Diese Vorgabe ist nicht umsetzbar. Daher dient in dieser Arbeit als Ausgangsbasis das Projekt „Wecker 2.0“ (vgl.: [10]). An den Szenarien des Weckers sollen sich die Ziele in dieser Arbeit orientieren. Die Situationen, die für den „Wecker 2.0“ relevant sind, sollen auch für die Evaluation der Kontexterkennung dienen. Das Framework soll später auch für andere Kontexte einsetzbar sein. Dafür soll es sich einfach für andere Programme erweitern lassen und in diesem Sinne universell nutzbar sein.

In einem ersten Schritt müssen die relevanten Kontexte für diese Arbeit identifiziert werden. Das Erkennen dieser Kontexte soll durch einen Klassifikationsalgorithmus erfolgen. Es muss ein Algorithmus gefunden werden, der den Anforderungen dieser Arbeit genügt. Dieser Algorithmus soll dann für die Wohnung umgesetzt werden. Hierbei sind die verschiedenen Probleme der Klassifikation zu lösen. Zu den Problemen zählen unter anderem Überanpassung der Klassifikation und der Umgang mit falsch erkanntem Kontext.

Als Eingangs-Daten für das Erkennen des Kontext dienen die Sensoren der

[1] Situation und Kontext werden in dieser Arbeit synonym verwandt

Wohnung. Im „iFlat“ gibt es viele verschiedene Sensoren [17]. Jeder dieser Sensoren kann als Quelle für die Kontext-Erkennung dienen [1]. Der Kontext ist nicht in jedem Fall der Wert eines Sensors, sondern der Kontext bildet sich in der Regel auch aus den Werten verschiedener Sensoren. Dabei ist manchmal der Zeitpunkt wichtig oder auch ein Zeitraum in dem der Sensor seine Werte ändert. Kontext kann auch aus anderen Informationsquellen gewonnen werden. Andere Quellen sind beispielsweise Informationsdienste aus dem Internet, die das Wetter zur Verfügung stellen oder Auskunft über die öffentlichen Verkehrsmittel geben. Die Ausgaben der Sensoren müssen in der Regel umgewandelt werden, da sie sonst für eine Klassifikation von Kontext nicht geeignet sind. Die Sensor- und Informationsquellen werden dann zu Merkmalen oder Features[2], aus denen der Kontext anschließend erkannt werden kann. In dieser Arbeit sollen die Möglichkeiten des „iFlat“ identifiziert werden, welche Sensoren zur Verfügung stehen und welche für die Kontext-Erkennung noch fehlen.

Für jeden einzelnen Kontext ist wahrscheinlich nur eine Teilmenge aller Features relevant. Je mehr Features bei der Klassifikation beachtet werden müssen, desto komplexer wird das gesamte System. Daher ist es wahrscheinlich sinnvoll, für jeden Kontext eine Teilmenge der Features zu bestimmen. In dieser Arbeit soll daher die gesamte Menge der Features auf einen Feature-Vektor reduziert werden, der nur aus relevanten Features besteht und nicht aus weiteren Features, die für die Klassifikation keine Bedeutung haben. Genauso sollten redundante Features in dem Feature-Vektor reduziert werden.

2 Überblick über das Thema

2.1 Vergleich mit anderen Arbeiten

Kontext Aware Computing findet sich in sehr unterschiedlichen Bereichen der Informatik wieder. Beispielsweise wird Kontext eingesetzt, um die verfügbaren Informationen den aktuellen Umständen anzupassen (vgl. [14] und [4]) oder um die Daten zu reduzieren, die in der entsprechenden Situation benötigt werden (vgl. [2]). Beim „Wecker 2.0“ wird der Kontext eingesetzt, um die Funktionen des Programms der aktuellen Situation in der Wohnung anzupassen. Kontext wird so auf viele verschiedene Weisen genutzt. Allen Arbeiten haben gemeinsam, dass eine bestimmte Menge reduziert werden soll. Bei dieser Menge kann es sich um Daten, Funktionen oder Informationen handeln. Die Umgebung, in der der Kontext genutzt wird kann sehr unterschiedlich sein [6]. In den oben genannten Beispielen

[2] Merkmale und Feature werden hier synonym verwendet

sind es eine Einkaufsstraße, eine Wohnung, ein Museum oder ein Konferenzraum. Die Umgebung für diese Arbeit soll sich auf eine Wohnung beschränken. Es ist geplant, dass auch Informationen von außerhalb der Wohnung für die Kontexterkennung zur Verfügung stehen.

Es stellt sich auch die Frage nach der Darstellung des Kontextes. In [3] wurden verschiedene Arten der Notation gegenübergestellt. Dabei wurden die Kategorien Dimension, Repräsentationsmerkmal und Verwaltung/Nutzung unterschieden und bewertet. Für das „iFlat“ hat sich gezeigt, dass sich ein ontologiebasiertes Kontextmodell am besten eignet [8]. Dieses Modell wird in [26] genauer beschrieben. Ein ontologiebasiertes Modell hat neben den verschiedenen Vorteilen auch den Nachteil, dass es sehr komplex ist. Daher muss für diese Arbeit erneut überprüft werden, ob es sich weiterhin für die Darstellung des Kontext eignet und ob weiterhin das 5W1H Modell [16] als Grundlage für die Dimensionen der Ontologie eingesetzt werden soll.

2.2 Abgrenzung der Arbeit

In dieser Arbeit soll Kontext aus verschiedenen Datenquellen erkannt werden. Die Datenquellen sollen dabei nicht zum Inhalt dieser Arbeit gehören. Die Arbeit bettet sich im Forschungskontext „iFlat“ der HAW Hamburg [20] ein. Hier sollen die vorhandenen Sensoren des „iFlat“ genutzt werden. Sensoren die noch nicht fertiggestellt sind oder die sich noch in der Planung befinden, werden für diese Arbeit simuliert. Andere Informationsquellen (z.B. das Internet) werden genutzt, wenn diese verfügbar sind, ansonsten werden sie simuliert.

In dieser Arbeit soll kein Programm entstehen, dass jeden möglichen Kontext erkennt. Das ist aufgrund der Vielfalt von möglichen Kontexten nicht möglich. Daher soll in dieser Arbeit ein Framework entstehen, das mit geringem Aufwand an neue Situationen angepasst werden kann. Es sollen neue Features hinzugefügt werden können, bestehende Features sollen entfernbar sein. Genauso soll es möglich sein, einen neuen Kontext zu definieren und diesen durch das Framework erkennen zu lassen. Beispielhaft sollen einige Szenarien vom „Wecker 2.0“ umgesetzt werden.

Nach [13] teilt sich Kontext in internen und externen Kontext. Externer Kontext ist, nach dem 5w1h Modell, das „wer, wo, was, wie und wann“ oder all das, was durch Sensoren direkt messbar ist. Interner Kontext bezieht sich auf das „warum“, die Intention, die hinter der vorliegenden Aktion steckt. Beispielsweise steht der Bewohner um 3:00 Uhr Nachts aus seinem Bett auf. Dieses könnte als „Aufstehen“ von den Sensoren in der Wohnung erkannt werden und ist ein externer Kontext. Der interne Kontext ist der Gedanke, der den Bewohner dazu bewegt. Hier könnte

es sein, dass der Bewohner ins Bad möchte oder in die Küche um etwas zu trinken. In dieser Arbeit soll sich auf den externen Kontext beschränkt werden. Interner Kontext ist ein weites Feld, dem man sich nur statistisch annähern kann, da es bisher keine Möglichkeit gibt, die Gedanken eines Menschen direkt zu bestimmen. Daher soll er nicht Bestandteil der Arbeit sein.

3 Aufbau der Arbeit

3.1 Architektur

In dieser Arbeit soll aus primitiven Sensordaten komplexe Kontextinformation generiert werden. Dabei ist die Anzahl der Sensoren, der Weg, wie die der Kontext erkannt werden soll und die Anzahl der verschiedenen Kontexte nicht festgelegt. Das Ziel ist es, ein Framework zu schaffen, in dem neue Daten in den Prozess hinzugefügt oder bestehende Daten entfernt werden können. Außerdem soll ein neu definierter Kontext integrierbar sein, wenn er von einem neuen Modul benötigt wird.

Die Komponenten für die Kontexterkennung teilen sich in die Bereiche Daten-Generator, Feature-Generator, Feature-Selection und Kontext-Classification auf (siehe: Abbildung: 1). Der Daten-Generator besteht aus den Sensoren des iFlat (vgl.: [22]) und externen Informationsquellen. Aus diesen Daten werden die Features anschließend generiert. Dieses geschieht im „Feature Generator“. Dabei kann ein Feature auch aus mehreren Sensoren erzeugt werden oder ein Sensor wird über einen Zeitraum interpretiert, um die erste oder zweite Ableitung der Sensorwerte zu erhalten. Verschiedene Features zusammengefasst ergeben einen Feature-Vektor. Aufgrund dieses Feature-Vektors wird mit Klassifikationsalgorithmen die aktuelle Situation der Wohnung erkannt. In [1] wird Kontext so definiert:

> „Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves“

Nach dieser Definition können alle Sensor-Daten, alle Informationen aus den Zwischenschritten und die Kontextinformationen selbst, die aus dem Feature-Generator oder der Kontext-Klassifikation gewonnen werden, für eine Anwendung von Interesse sein. Daher muss die Architektur so gewählt werden, dass die Kommunikation zwischen den Komponenten dynamisch ist und jederzeit zusätzliche Komponenten in das System integriert werden können. Um diese Dynamik zu gewährleisten, soll in dieser Arbeit die Blackboard Architektur [11] eingesetzt werden.

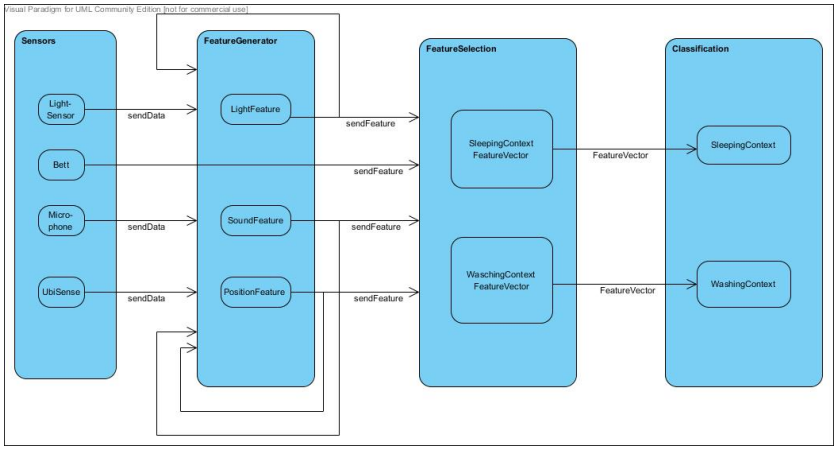

Abbildung 1: Kontext-Erzeugung

Das Blackboard ist eine Kommunikationsschnittstelle zwischen Komponenten mit der die Komponenten gleichberechtigt und entkoppelt miteinander kommunizieren können. Auf dem Blackboard werden alle Informationen (Sensordaten, Feature, Kontext) für alle anderen Komponenten zur Verfügung gestellt. Die einzelnen Komponenten melden sich für genau die Informationen am Blackboard an, die sie benötigen. So können verschiedene Erzeuger oder Abonnenten von Informationen dem System hinzugefügt oder entfernt werden. Durch diese Entkopplung besteht die Gefahr, dass es für eine Information keinen Erzeuger oder keinen Abonnenten gibt. Dieses muss bei der Entwicklung der Programme berücksichtigt werden.

3.2 Vorgehen

In dieser Arbeit soll das Framework nach dem „Top-Down“ Verfahren entwickelt werden. Somit teilt sich die Arbeit in die folgenden vier Schritte auf:

Kontext-Klassifikation: Verschiedene Kontexte müssen als erstes definiert werden. Der Kontext soll dann durch ein Klassifikationsverfahren aus einem Feature-Vector erkannt werden. Zu klären ist, welches Klassifikationsverfahren am besten für die Kontexterkennung eingesetzt werden soll.

Feature: Die Features sind vorverarbeitete Sensordaten oder Informationen aus externen Quellen. Es muss ermittelt werden, welche Features sich aus den Sensorendaten generieren lassen und welche externen Quellen zur Verfügung stehen. Auch hier stellt sich die Frage, was notwendig ist, um das Erkennen von Kontext zu gewährleisten.

Data Generator: Es müssen die verschiedenen Sensoren und Informationsquellen identifiziert werden, die für die Erzeugung der Features notwendig sind. Gegebenenfalls müssen verschiedene Sensoren simuliert werden, wenn sie nicht zur Verfügung stehen.

Feature Vector: Für einen bestimmten Kontext werden wahrscheinlich nicht alle Features aus der Wohnung benötigt, um eine Klassifikation zu gewährleisten. Außerdem kann es sein, dass verschiedene Features redundant zueinander sind. Daher wird ein Verfahren benötigt, dass aus den vorhandenen Features eine Teilmenge extrahiert.

3.2.1 Kontext-Klassifikation

Der erste Schritt dieser Arbeit ist es, die relevanten Kontexte zu bestimmen, die erkannt werden sollen. Kontext kann nahezu jede denkbare Situation sein. Daher ist es notwendig, die Möglichkeiten zu beschränken. Als Basis sollen die Szenarien des Wecker 2.0 dienen. Für diese Szenarien sollen verschiedenen Kontexte definiert werden. Hierbei sind die Übergänge von einem Kontext zum nächsten ein Problem. Es muss die Frage geklärt werden, was zu einem Kontext dazugehört und was nicht. Beispielsweise könnte das Liegen im Bett schon der „Schlafen"-Kontext sein. Genauso ist es auch denkbar, dass erst der Schlaf-Zustand selbst zu diesem Kontext gehört und sonst nichts. Dann gibt es noch das Problem der Übergänge von einem nicht aktivem Kontext zum aktivem Kontext. Wenn sich beispielsweise ein Bewohner auf die Grenzlinie zwischen dem Wohnbereich und dem Küchenbereich aufhält, weil er noch in der Küche sein will, aber gleichzeitig den TV im Wohnbereich beobachtet. Der Kontext „Küchenarbeit" könnte für diese Situation nicht klar zugeordnet werden, da der Bewohner mal in der Küche ist und mal nicht. Hier könnte man die Bereiche anders voneinander trennen oder den Kontext anders definieren. Das Problem würde aber an einer neuen Grenze wieder auftreten. Oder bei einem neu definierten Kontext gibt es andere Grenzen, auf denen es Zweideutigkeit gibt. Daher sollte hier eine einfache Lösung gefunden werden, die den gegebenen Szenarien genügt.

Nachdem klar ist, welcher Kontext erkannt werden soll, müssen Klassifikationsverfahren gefunden werden, die den Anforderungen in dieser Arbeit entsprechen. Diese Verfahren gilt es dann umzusetzen. Da zu diesem Zeitpunkt noch keine realen Daten für die Klassifikation vorliegen, müssen entsprechende Testdaten erzeugt werden. Die Testdaten sollen nicht die Sensoren aus dem iFlat direkt simulieren, sondern sie dienen der Verifikation der Klassifikation. Trotzdem muss bei diesen Daten darauf geachtet werden, dass sie repräsentativ für die Möglichkeiten der Sensoren sind, da sie später durch echte Daten ersetzt werden sollen.

3.2.2 Feature-Generator

Für die Klassifikation müssen Informationen zur Verfügung gestellt werden. Diese Informationen stammen aus den Sensoren der intelligenten Wohnung und aus anderen Quellen, wie beispielsweise dem Internet. Die Sensoren liefern in der Regel sehr viele Daten pro Sekunde. Die Daten der verschiedenen Sensoren sind heterogen und für eine direkte Nutzung meistens nicht geeignet. Sie liefern meistens Zahlen in einem individuellen Wertebereich ohne Einheit. Beispielsweise werden aus dem Ortungssystem „UbiSense“ zwei Vektor-Werte geliefert, aus denen die Position im Raum berechnet werden kann [24]. In diesem Schritt wird auch mit der Vorverarbeitung von den Sensoren abstrahiert. Dadurch kann das System leichter an neue Situationen angepasst werden. Des Weiteren sind nicht nur die direkten Sensordaten für die Klassifikation von Interesse, sondern auch Daten verschiedener Sensoren in der Kombination oder die Daten eines Sensors in der Ableitung über die Zeit. Beispielsweise könnte das UbiSence [24] dazu genutzt werden, die Geschwindigkeit der Person zu bestimmen oder die Position eines Stuhls kombiniert mit der Position einer Person, könnte zu einem „sitzen“ Feature werden. Eine weitere Herausforderung ist die Anzahl der Datenpackete pro Sekunde, die die Sensoren liefern. Diese müssen im besten Fall in nahezu Echtzeit umgewandelt und weitergeleitet werden. Hier könnte das Tool ESPER [12] helfen. Es eignet sich um Streams zu bearbeiten und Entscheidungsbäume auf Streams zu erstellen [7].

3.2.3 Data Generator

Der Begriff „Daten-Generator“ bezieht sich hier auf alles, was Daten erzeugen kann und in der Lage ist diese auf dem Blackboard zur Verfügung zu stellen. Dieses können primitive Sensoren sein, die beispielsweise die Intensität des Lichtes messen und die Werte über eine Schnittstelle weiterleiten, bis hin zu komplexen Systemen, wie dem „intelligenten Bett“ [15], welches in der Lage ist, komplexe Daten zu erzeugen. Des Weiteren sind hier auch Schnittstellen gemeint, die Daten aus fremden Systemen beziehen und für den „Living Place“ aufbereiten. Hier kann als Beispiel ein Programm genannt werden, das aus dem Internet das aktuelle Wetter holt und es auf dem Blackboard zur Verfügung stellt. Für diese Arbeit sollen die bestehenden Sensoren und Programme des „iFlat“ genutzt werden, soweit diese einsatzbereit sind. Alle weiteren Sensoren, die für die Kontext-Erkennung benötigt werden, könnten auch simuliert werden.

Die Sensoren lassen sich auf verschiedenen Ebenen betrachten:

Bereiche: Die Bereiche, in denen die Sensoren aktiv sind (z.B. Schlafbereich, Wohnbereich oder Badezimmer).

Funktion: Der Typ der Daten, die erfasst werden (z.B. Position-Koordinaten oder Helligkeitswerte).

Typ: Die Art des Sensors, der die Daten liefert (z.B. Kamera oder UbiSence).

Für die Kontext-Erkennung sind die Anzahl und die Art der Sensoren nicht wichtig. Es kommt auf die Art der Daten an, die die Sensoren liefern beziehungsweise den Service, den die Sensoren zur Verfügung stellen. Beispielsweise ist für verschiedene Kontexte die Position des Bewohners interessant. Diese Position könnte über eine Kamera erfasst werden, durch induktive Sensoren oder durch das UbiSence-System. Die Position des Sensors ist nicht direkt wichtig, aber deren Aktionsbereich, wenn dieser sich nicht über die gesamte Wohnung erstreckt. Es muss die geforderte Funktion für jeden relevanten Bereich der Wohnung zur Verfügung stehen. Eventuell ist es notwendig, verschiedene Sensoren für die gleiche Funktion zu nutzen. Die Art des Sensors spielt in dieser Arbeit eine untergeordnete Rolle. Bei der Entwicklung der Features muss darauf geachtet werden, dass die Sensoren auch existieren und die Daten liefern. Hier bietet sich eine Zusammenarbeit mit dem Projekt der Kommunikationsinfrastruktur an [17]. Hier soll eine Übersicht über die verschiedenen Funktionen der Sensoren im Living Place zur Verfügung gestellt werden.

3.2.4 Feature-Vector

In der Theorie ist die Frage nach einem Feature-Subset unwichtig. Ein Bayes-Klassifikator beispielsweise wird theoretisch immer besser, je mehr Features beachtet werden. Unter praktischen Gesichtspunkten gibt es unter anderem zwei Probleme in der Klassifikation. Erstens kann es bei zu vielen Feature-Vektoren zu einer Überanpassung kommen. Zweitens ist die Optimierung der Klassifikation sehr komplex. Beispielsweise ist das Problem, den perfekten Entscheidungsbaum zu finden, NP-Hart und in neuronalen Netzen ist das Problem noch komplexer [18]. Daher sollte aus dem gesamten Feature-Set für jeden Kontext ein Feature-Subset gesucht werden [19]. Diese Suche soll durch eine automatische Suche geschehen. In einer Wohnung kann es für verschiedene Zustände innerhalb eines Jahres verschiedene Auswirkungen geben. Ein offenes Fenster hat im Winter eine andere Wirkung auf die Wohnung als im Sommer. Weil es im Rahmen der Arbeit nicht möglich ist, ausreichend Testdaten über Jahre zu sammeln, muss die Suche mit Expertenwissen eingegrenzt werden. Es ist geplant, die Wrapper-Methode aus [18] umzusetzen, da die Voraussetzungen hier passend erscheinen.

3.3 Herausforderungen

Eine Herausforderungen der Masterarbeit ist die Abhängigkeit von verschiedenen anderen Projekten. Es werden Sensordaten benötigt, um das System insgesamt testen zu können. Außerdem werden Testdaten benötigt, um die Klassifikation zu trainieren. Wenn die Sensoren in der Wohnung nicht rechtzeitig einsatzbereit sind, wäre es eine Möglichkeit, diese Daten zu simulieren. Hier besteht aber die Gefahr, dass die Daten nicht repräsentativ für die Sensoren der Wohnung sind.

Sensoren haben einen Arbeitsbereich, in dem ihre Ausgaben schwanken. Das bedeutet, dass die Features variable Daten liefern. Dieses kann gerade bei der Grenz-Problematik zu Schwierigkeiten führen. Vielleicht könnte durch eine Glättung der Werte dieses Problem abgemildert werden.

Des Weiteren bringt die Arbeit mit Kontext verschiedene Risiken. Es muss die Frage geklärt werden, wie verschiedene Kontexte sich von einander trennen lassen. Beispielsweise stellt sich die Frage, ob der Kontext „Schlafen“ mit dem liegen im Bett beginnt oder ob dieser Kontext erst mit einem Schlafenden-Zustand startet. Möglicherweise ist die Position Bett für den Kontext „Schlafen“ auch nicht notwendig, aber hinreichend, da der Bewohner auch im Wohnzimmer schlafen kann. Dann muss geklärt werden, wie fein der Kontext definiert werden soll. Beispielsweise könnte „Schlafen“ mit dem Ausziehen anfangen und mit einer angezogenen Person am nächsten Morgen enden. Oder die Aktionen „Ausziehen“, „Ins Bett gehen“, „Lesen“, „Licht ausschalten“, „Einschlafen“ u.s.w. sind eigene Kontexte. Hier könnte jeder Kontext noch beliebig verfeinert werden.

Wie oben schon erwähnt, können Sensoren ausfallen, aber auch ein Klassifikationsalgorithmus liegt beim Klassifizieren nicht immer richtig. Daher gibt es bei dieser Arbeit auch falsche positive und falsche negative Treffer. Dies muss in der Evaluierungsphase ausreichend getestet werden, wie gut das System unter „realen“ Bedingungen funktioniert. Eine Möglichkeit, diese Herausforderung zu umgehen, ist es, sie auf die Anwendungen zu übertragen, die mit dem Kontext arbeiten.

4 Zusammenfassung

4.1 Zusammenfassung

In dieser Arbeit soll das Erkennen von Kontext in einer intelligenten Wohnung umgesetzt werden. Dafür muss als erstes der Kontext festgelegt werden, der erkannt werden soll. Das Erkennen des Kontext fängt bei den Daten an, die es im „iFlat“ (einer intelligenten Wohnung) gibt. Die Daten müssen für die Nutzung der Klassifikationsalgorithmen vorbereitet werden, da sie heterogen und dynamisch sind,

außerdem sind sie in der Regel mit einer Toleranz behaftet. Anschließend werden bestimmte Features aus allen Daten ausgewählt, um mit ihnen den Kontext zu erkennen. Die Auswahl der Features soll durch einen Wrapper vorgenommen werden, der nach dem optimalen Feature-Subset sucht. Das System soll auf einer Blackboard-Architektur aufbauen, da die Struktur der Kommunikation der Komponenten untereinander nicht festgelegt ist und es später möglich sein soll, weitere Komponenten hinzuzufügen oder bestehende aus dem System zu entfernen.

4.2 Ausblick

Das Verfahren sucht nach Situationen in der Wohnung. Es ist denkbar, während des Erkennens von Kontext den aktuellen Kontext aus der Wohnung zu nutzen. Die Sensoren könnten sich dem Kontext anpassen, um bessere Ergebnisse zu liefern und die Klassifikationsverfahren könnten je nach Kontext die Feature-Vektoren anders auswählen. Dieser Ansatz wurde auch in [5] verfolgt.

In dieser Arbeit wird der interne Kontext nicht weiter verfolgt. Diese können anschließend in das Framework integriert werden, um eine Vorhersage der folgenden Aktionen leichter treffen zu können. Beispielsweise könnte nach dem Aufstehen mitten in der Nacht das Licht im Bad angeschaltet werden, wenn als Grund des Aufstehens der Weg ins Bad erkannt würde.

Andere Anwendungen könnten auf Grundlage dieser Arbeit entwickelt werden. Kontext kann andere Programme beeinflussen oder diese optimieren, wenn erst klar ist, welche Möglichkeiten es gibt. Dafür müsste dann aber die Erkennung von Kontext um neue Kontexte erweitert werden, damit diese auch durch die anderen Programme genutzt werden können.

Literatur

[1] ABOWD, G. D. ; DEY, A. K. ; BROWN, P. J. ; DAVIES, N. ; SMITH, M. ; STEGGLES, P. : Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. Springer-Verlag (HUC '99). – ISBN 3–540–66550–1, 304–307

[2] BOLCHINI, C. ; CURINO, C. ; SCHREIBER, F. A. ; TANCA, L. : Context Integration for Mobile Data Tailoring. In: *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2526–1, S. 5

[3] BOLCHINI, C. ; CURINO, C. A. ; QUINTARELLI, E. ; SCHREIBER, F. A. ; TANCA, L. : A Data-oriented Survey of Context Models. In: *ACM SIGMOD Record* 36 (2007), Dec, Nr. 4, S. 19–26. http://dx.doi.org/10.1145/1361348.1361353. – DOI 10.1145/1361348.1361353

[4] CANO, J.-C. ; MANZONI, P. ; TOH, C. K.: UbiqMuseum: A Bluetooth and Java Based Context-Aware System for Ubiquitous Computing. In: *Wirel. Pers. Commun.* 38 (2006), Nr. 2, S. 187–202. http://dx.doi.org/10.1007/s11277-005-9001-x. – DOI 10.1007/s11277–005–9001–x. – ISSN 0929–6212

[5] DAI, P. ; XU, G. : Context-aware computing for assistive meeting system. In: *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*. New York, NY, USA : ACM, 2008. – ISBN 978–1–60558–067–8, S. 1–7

[6] ELLENBERG, J. : Context Awareness im Vergleich / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-aw2/Ellenberg/bericht.pdf. – Forschungsbericht

[7] ELLENBERG, J. ; MÜLLER, V. V. D. (Hrsg.): *Event Stream Processing mit ESPER: unter Einsatz von Datamining Verfahren*. VDM Verlag Dr. Müller, 2010

[8] ELLENBERG, J. : Vorarbeiten für den Wecker 2.0 / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/ellenberg.pdf. – Forschungsbericht

[9] ELLENBERG, J. : Ein Wecker in einem ubicom Haus / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/Ellenberg/bericht.pdf. HAW Hamburg. – Forschungsbericht

[10] ELLENBERG, J. : Entwicklung des Wecker 2.0 / HAW Hamburg. Version: 2011. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-proj2/ellenberg.pdf. – Forschungsbericht

[11] ERMAN, L. D. ; HAYES-ROTH, F. ; LESSER, V. R. ; REDDY, D. R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. In: *ACM Comput. Surv.* 12 (1980), June, 213–253. http://dx.doi.org/http://doi.acm.org/10.1145/356810.356816. – DOI http://doi.acm.org/10.1145/356810.356816. – ISSN 0360–0300

[12] ESPERTECH-INC: *Esper: Event Processing for Java*. http://www.espertech.com/products/esper.php. Version: 2008

[13] GREENBERG, S. : Context as a dynamic construct. In: *Hum.-Comput. Interact.* 16 (2001), Nr. 2, S. 257–268. http://dx.doi.org/10.1207/S15327051HCI16234_09. – DOI 10.1207/S15327051HCI16234_09. – ISSN 0737–0024

[14] GUI, F. ; GUILLEN, M. ; RISHE, N. ; BARRETO, A. ; ANDRIAN, J. ; ADJOUADI, M. : A Client-Server Architecture for Context-Aware Search Application. In: *Network-Based Information Systems, International Conference on* 0 (2009), S. 539–546. http://dx.doi.org/10.1109/NBiS.2009.75. – DOI 10.1109/NBiS.2009.75. ISBN 978–0–7695–3767–2

[15] HARDENACK, F. : Das intelligente Bett - Semantische Interpretation auf Basis kapazitiver Sensoren / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/hardenack/bericht.pdf. – Forschungsbericht

[16] JANG, S. ; WOO, W. : 5W1H: Unified user-centric context. In: *Proceedings of the 7th International Conference on Ubiquitous Computing* Citeseer, 2005

[17] JOHANNSEN, B. : Generische Modelbasierte Kommunikationsinfrastruktur / HAW Hamburg. Version: 2010. http://users.

informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/johannsen/bericht.pdf. – Forschungsbericht

[18] KOHAVI, R. ; JOHN, G. H.: Wrappers for feature subset selection. In: *Artif. Intell.* 97 (1997), December, 273–324. http://dx.doi.org/10.1016/S0004-3702(97)00043-X. – DOI 10.1016/S0004–3702(97)00043–X. – ISSN 0004–3702

[19] LIU, H. ; YU, L. : Toward Integrating Feature Selection Algorithms for Classification and Clustering. In: *IEEE Trans. on Knowl. and Data Eng.* 17 (2005), April, 491–502. http://dx.doi.org/http://dx.doi.org/10.1109/TKDE.2005.66. – DOI http://dx.doi.org/10.1109/TKDE.2005.66. – ISSN 1041–4347

[20] LUCK, P. D. K. ; KLEMKE, P. D. G. ; GREGOR, S. ; RAHIMI, M. A. ; VOGT, M. : Living Place Hamburg – A place for concepts of IT based modern living / Hamburg University of Applied Sciences. Version: Mai 2010. http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf. – Forschungsbericht

[21] *Kapitel* 4. In: NARDI, B. A.: *Studying context: A comparison of activity theory, situated action models, and distributed cognition*. The MIT Press, 35–52

[22] PAUTZ, A. : Kabelloses Sensornetzwerk im Living Place Hamburg / HAW Hamburg. Version: 2010. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/pautz/bericht.pdf. – Forschungsbericht

[23] STEGELMEIER, S. : iFlat - Eine dienstorientierte Architektur für intelligente Raeume / HAW Hamburg. Version: 2009. http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/aal2009.pdf. VDE2. – Forschungsbericht

[24] TENNSTEDT, S. : Projektbericht: Emotional Tent Ambient Awareness / HAW Hamburg. Version: 2009. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master08-09-proj/tennstedt/bericht.pdf. – Forschungsbericht

[25] WEISER, M. : The computer for the 21st century. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (1999), Nr. 3, S. 3–11. http://dx.doi.org/10.

1145/329124.329126. – DOI 10.1145/329124.329126. – ISSN 1559–1662

[26] WINKLER, R. : Entwicklung eines ontologiebasierten Kontextmodells für kollaborative Web-Anwendungen / Technischen Universitaet Dresden. 2007. – Forschungsbericht

Bildquellen

Die Quellen der Bilder sind:

- Abbildung 1 wurde mit Visual Paradigm for UML 7.2 erstellt.