Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Andreas Basener

Clusterbildung von Sensordaten aus einem Smarthome zur Aktivitäteninterpretation

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Andreas Basener

Clusterbildung von Sensordaten aus einem Smarthome zur Aktivitäteninterpretation

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Gunter Klemke

Eingereicht am: 9. Dezember 2013

Andreas Basener

Thema der Arbeit
Clusterbildung von Sensordaten aus einem Smarthome zur Aktivitäteninterpretation

Stichworte
Knowledge Discovery in Databases, Datamining, Clusterbildung, Aktivitätenentdeckung, Smarthome, Sensoren

Kurzzusammenfassung
Die Steuerung einer intelligenten Wohnung sollte sich an die Bedürfnisse des Bewohners anpassen. Dazu muss die Steuerung wissen, welche Aktivitäten der Bewohner ausführt. In dieser Masterarbeit werden die Grundlagen für eine Aktivitätenentdeckung erläutert und darauf aufbauend eine Aktivitätenentdeckung entwickelt, implementiert und evaluiert. Zudem wird darauf eingegangen, wie die entdeckten Aktivitäten von anderen Projekten verwendet werden können.

Andreas Basener

Title of the paper
Clustering of sensordata of a smarthome for activity interpretation

Keywords
Knowledge Discovery in Databases, data mining, clustering, activity discovery, smart home, sensors

Abstract
The controller of a smart home should adapt to the needs of its resident. To achieve that, the controller has to know which activities the resident is performing. This thesis explains the basics of activity discovery and describes the design, implementation and evaluation of an activity discovery. Additionally the usage of the discovered activities by other project will be discussed.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Ziele . . . 3
- 1.3 Umfeld . . . 4

2 Analyse **6**
- 2.1 Knowledge Discovery in Databases . . . 6
- 2.2 Datenquellen . . . 13
- 2.3 Related Works . . . 17
  - 2.3.1 BehaviourScope . . . 17
  - 2.3.2 Aware Home . . . 21
  - 2.3.3 CASAS . . . 24
  - 2.3.4 Bewertung der Verfahren . . . 27
- 2.4 Fazit und Zielsetzung . . . 29

3 Design **30**
- 3.1 Infrastruktur . . . 30
- 3.2 Sensoren . . . 32
- 3.3 Datenstrukturen . . . 34
  - 3.3.1 Event . . . 34
  - 3.3.2 Sequenzen . . . 35
  - 3.3.3 Muster . . . 35
  - 3.3.4 Cluster . . . 36
- 3.4 Distanzen . . . 37
- 3.5 Programmablauf . . . 39
  - 3.5.1 Vorverarbeitung . . . 39
  - 3.5.2 Muster entdecken . . . 41
  - 3.5.3 Muster clustern . . . 42
  - 3.5.4 Muster bereitstellen . . . 42
- 3.6 Fazit . . . 43

4 Implementierung **44**
4.1 Daten erfassen . . . 44
4.2 Vorverarbeitung . . . 46
4.3 Functional Spaces . . . 46
4.4 Verwendete Klassen und Datenstrukturen . . . 47
4.4.1 IEvent und Event . . . 48
4.4.2 Sequence . . . 49
4.4.3 Pattern . . . 49
4.4.4 Cluster . . . 50
4.5 Distanzen . . . 52
4.6 Aktivitätenentdeckung . . . 53
4.6.1 Schritt 1: Muster entdecken . . . 54
4.6.2 Schritt 2: Bestes Muster speichern . . . 57
4.6.3 Schritt 3: Eventliste komprimieren . . . 57
4.6.4 Schritt 4: Muster clustern . . . 58
4.6.5 Schritt 5: Cluster zusammenfassen . . . 59
4.6.6 Schritt 6: Cluster bereitstellen . . . 60
4.7 Fazit . . . 63

5 Evaluierung **64**
5.1 Ausgangsdaten . . . 64
5.2 Aktivitätenentdeckung . . . 66
5.3 Ergebnis . . . 67
5.3.1 Versuch 1: alle Sensortypen . . . 68
5.3.2 Versuch 2: nur Positionsdaten . . . 70
5.3.3 Versuch 3: keine Positionsdaten . . . 71
5.3.4 Auswertung der Versuche . . . 73
5.4 Fazit . . . 74

6 Schluss **76**
6.1 Zusammenfassung . . . 76
6.2 Bewertung . . . 77
6.3 Ausblick . . . 78

Literaturverzeichnis **80**

1 Einleitung

1.1 Motivation

Schon seit langem wird versucht, das alltägliche Leben und Wohnen durch Technik angenehmer zu gestalten. In zahlreichen Büchern und Filmen haben sich die Autoren Gedanken gemacht, wie eine intelligente Wohnung aussehen könnte.
Die notwendigen Technologien, um die Ideen umsetzen zu können, wurden im Laufe der Zeit so weit verbessert und entwickelt, dass die Fiktion Realität werden kann. Das stellt die Informatik vor die Aufgabe, Lösungen zu finden, die eine reibungslose Interaktion zwischen Bewohner und Wohnung ermöglichen.

Momentan gibt es zwei Richtungen, in denen das Leben in einer intelligenten Wohnung erforscht wird.
In der ersten wird eine Wohnung mit Steuergeräten, Sensoren und Aktoren ausgestattet, die eine feste Programmierung erhalten. Dadurch können bestimmte Aspekte des täglichen Lebens zentral gesteuert werden. So ist z.B. eine intelligente Belüftung und Klimatisierung der Wohnung möglich, aber auch eine komplexe Lichtsteuerung.
Die andere Richtung ist das Ambient Assisted Living, bzw. Assistenzsysteme für ein selbstbestimmtes Leben. Hier wird versucht, hilfsbedürftigen Menschen ein möglichst selbstständiges Leben in der eigenen Wohnung zu ermöglichen. Dazu werden z.B. die vitalen Parameter des Bewohners überwacht, um Notfälle zu erkennen oder um ihn an die Einnahme von Medikamenten zu erinnern.

Damit die Wohnung auf den Bewohner reagieren kann, muss die Wohnung aber erst einmal wissen, worauf sie reagieren soll. Dazu müssen die Aktivitäten des Bewohners bekannt sein, die die Grundlage für eine intelligente Steuerung der Wohnung bilden.

Dabei stellt sich die Frage, wie diese Aktivitäten definiert werden.
Eine Möglichkeit, dieses Problem anzugehen, ist, im Vorfeld eine genaue Auflistung und Beschreibung der erwarteten Aktivitäten zu erstellen. Diese Variante ist in der Praxis nur schwer umzusetzen. Das Modellieren von Aktivitäten ist sehr aufwendig. Zudem ist es mit dieser Methode schwierig, sich auf Veränderungen und Abweichungen in den Aktivitäten des Benutzers einzustellen. Es müsste jedes Mal das zugrunde liegende Modell manuell angepasst werden. Das führt zu einem enormen Aufwand, der kaum im Verhältnis zu dem gewünschten Nutzen steht.

Besser ist es, die Aktivitäten automatisch zu entdecken. Dazu werden die Sensordaten untersucht, die durch die Interaktion des Bewohners mit der Wohnung entstehen. In diesen Sensordaten werden Muster gesucht, die auf die Aktivitäten des Bewohners schließen lassen.

Für diese Aufgabenstellung gibt es bereits mit der Disziplin *Datamining* und dem ergänzendem *Knowledge Discovery in Databases* (kurz: *KDD*) Verfahren, mit denen bisher unbekannte Muster in Daten entdeckt werden können.
Weiterhin eignet sich die KDD dazu, Muster zu erkennen, ohne dass Vorgaben gemacht werden, wie diese Muster aussehen sollen.

In den nachfolgenden Kapiteln beschreibe ich die Zielsetzung (Kapitel 1.2) für diese Masterarbeit und erkläre, in welchem Umfeld (Kapitel 1.3) sie entstanden ist. Zunächst erläutere ich die Grundlagen, die für diese Masterarbeit notwendig sind (Kapitel 2). Danach stelle ich drei Projekte vor, die sich bereits mit dieser Fragestellung auseinander gesetzt haben, und gehe auf deren Ergebnisse ein (Kapitel 2.3). Anschließend beschreibe ich meinen eigenen Ansatz (Kapitel 3) und wie ich ihn umgesetzt habe (Kapitel 4), um anschließend meine Ergebnisse zu präsentieren (Kapitel 5). Zum Schluss erfolgt eine Bewertung der Ergebnisse und ein Ausblick, was in der Zukunft mit den Ergebnissen möglich ist (Kapitel 6).

1.2 Ziele

Das Ziel dieser Masterarbeit ist, ein Verfahren zu entwickeln, mit dem aus Sensordaten Aktivitäten entdeckt werden können. Dazu werden in den Sensordaten Muster gesucht. Ähnliche Muster werden zu Clustern zusammengefügt, woraus sich dann die entdeckten Aktivitäten ergeben.
Die entdeckten Aktivitäten sollen anschließend annotiert werden und anderen Projekten zur Verfügung stehen, damit diese daraus eine intelligente Steuerung der Wohnung entwickeln können.

Das zu entwickelnde Verfahren muss in der Lage sein, unterschiedliche Sensoren als Datenquelle zu nutzen. Die gewonnenen Daten müssen ausgewertet und für die weitere Verarbeitung vorbereitet werden. In diesen vorbereiteten Daten werden anschließend die Aktivitäten entdeckt.
Die so gewonnenen Aktivitäten werden um zusätzliche Informationen ergänzt und anschließend anderen Projekten zur Verfügung gestellt.

Neben dem Entdecken von Aktivitäten soll zusätzlich herausgefunden werden, ob die Kombination unterschiedlicher Sensoren Einfluss auf die Qualität der Entdeckung hat. Hier stellt sich die Frage, ob die Aktivitätenentdeckung zu besseren Ergebnissen führt, wenn mehr Sensoren verwendet werden, die weitere Aspekte des täglichen Lebens erfassen.
Neben der Quantität der Sensoren ist auch deren Qualität ein wichtiges Merkmal für die Aktivitätenentdeckung. Daher ist es erforderlich herauszufinden, wie sich Fehler in den Sensordaten auf die entdeckten Aktivitäten auswirken.

Neben der Aktivitätenentdeckung in einer intelligenten Wohnung ist auch die Aktivitätenentdeckung von Personen in anderen Zusammenhängen interessant. Daher sollte das Verfahren erweitert werden können, um z.B. Aktivitäten in einem größeren Bereich außerhalb der Wohnung wie der unmittelbare Nachbarschaft oder der ganzen Stadt zu erkennen.

1.3 Umfeld

Diese Masterarbeit ist im Rahmen des Projektes Livingplace der Hochschule für Angewandte Wissenschaften Hamburg entstanden. Das Livingplace ist eine intelligente Wohnung, mit der erforscht wird, wie moderne Technologien das Leben in einer Wohnung angenehmer gestalten und den Bewohner im täglichen Leben unterstützen können [von Luck et al., 2010].
Dabei bietet das Livingplace die Möglichkeit, Experimente in einer realen Umgebung durchzuführen.

Abbildung 1.1: Rendermodell des Livingplace

Im Livingplace sind bereits diverse Projekte realisiert worden oder werden zurzeit erforscht. Einige dieser Projekte eignen sich als Lieferanten von Sensordaten für die Akivitätenentdeckung.

So befindet sich z.B. ein intelligentes Bett [Hardenack, 2011] im Livingplace, dass die Schlafphasen des Bewohners erkennt. Ähnlich dazu ist das Projekt einer kontextsensitiven Couch von Mosawer Nurzai [Nurzai, 2013]. Dieses Projekt liefert unter anderem Informationen darüber, wo sich der Bewohner auf der Couch befindet.
Ein weiteres Projekt, das wichtige Daten für diese Masterarbeit liefert, ist die Installation eines Ubisense-Positionssystems. Damit kann die Position des Bewohners im Livingplace bis auf wenige Zentimeter genau bestimmt werden.

Für den Datenaustausch zwischen den Projekten wird eine Middleware eingesetzt, mit dessen Hilfe Nachrichten ausgetauscht werden können.

Eine genauere Beschreibung der verwendeten Projekte befindet sich in den Kapiteln 3.1 und 3.2.

2 Analyse

In diesem Kapitel erläutere ich die Grundlagen, die für diese Masterarbeit wichtig sind. Dazu beschreibe ich zunächst, was Knowledge Discovery in Databases ist, und welche Schritte dabei durchgeführt werden. Anschließend werden die Grundlagen für die Datenquellen beschrieben, die in dieser Masterarbeit verwendet werden. Danach stelle ich drei Projekte vor, die sich bereits mit Aktivitätenentdeckung beschäftigt haben. Zum Schluss erfolgt eine Beschreibung der Ziele dieser Masterarbeit und wie sie erfüllt werden sollen.

2.1 Knowledge Discovery in Databases

Knowledge Discovery in Databases (kurz KDD) und Datamining werden oft als synonyme Begriffe verwendet. In [Fayyad et al., 1996] wird aber eine wichtige Unterscheidung zwischen beiden Begriffen getroffen. Mit dem Begriff Datamining ist hier die Anwendung von Algorithmen auf Daten und die Extraktion von Mustern aus Daten gemeint und ist ein Teil der KDD. Die KDD enthält aber noch weitere Schritte.

In Abbildung 2.1 sind diese Schritte abgebildet. Dabei handelt es sich um *Fokussieren*, *Vorverarbeitung*, *Transformation*, *Datamining* und *Interpretation* [Ester and Sander, 2000]. Die einzelnen Schritte werden nachfolgend erklärt.

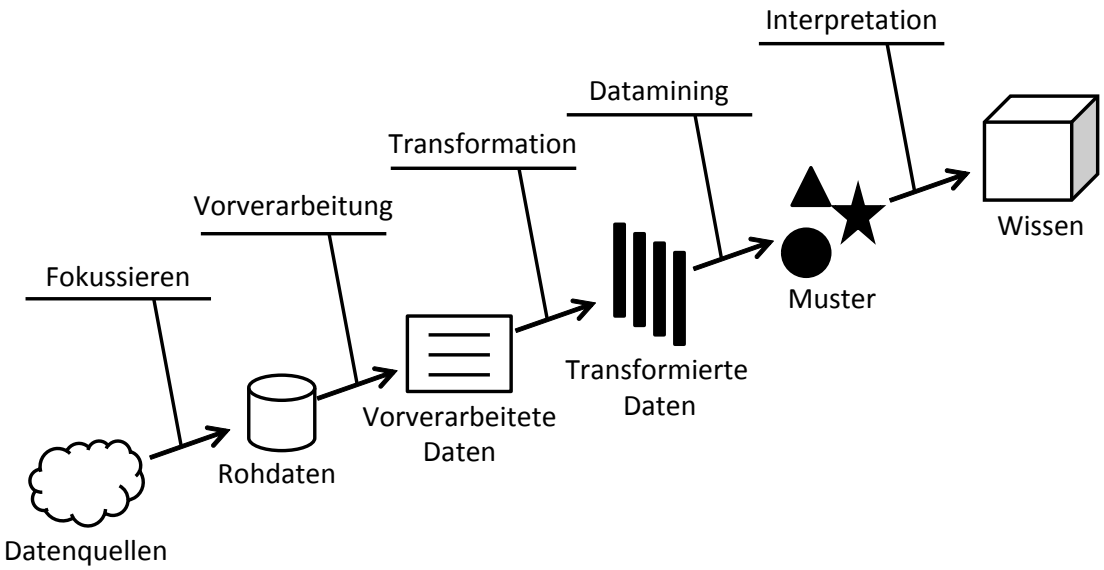


Abbildung 2.1: Schritte der KDD (nach [Fayyad et al., 1996, S. 41])

Fokussieren

Im ersten Schritt müssen die Daten ausgewählt werden, mit denen das KDD betrieben werden soll. Dazu wird zunächst entschieden, welche Datenquellen verwendet werden sollen. Es muss auch festgelegt werden, wie die Daten beschafft werden, z.B. ob sie direkt von den einzelnen Erzeugern der Daten geliefert werden oder ob die Daten in einem Zwischenschritt vorher zentral gebündelt werden.

Sind die Datenquellen ausgewählt, muss nun entschieden werden, wie die Daten gespeichert werden. Dabei ist es nicht nur wichtig, das Datenformat festzulegen, sondern auch wie die Daten gespeichert werden. Die Daten können z.B. in einem vom Menschen lesbaren Format als Textdateien gespeichert werden, als Binärdaten in einer Datenbank usw.
Wie die Daten gespeichert werden, hängt stark davon ab, wie sie beschafft, weiterverarbeitet und muss auf das jeweilige Verfahren angepasst werden.

Vorverarbeitung

Sind die Daten ausgewählt, die verwendet werden sollen, müssen sie nun vorverarbeitet werden. Die Rohdaten enthalten noch Fehler, Inkonsistenzen und liegen evtl. in einem Format vor, das sich für das Datamining nicht eignet.

Fehler in den Rohdaten können durch ungenaue Sensoren und Messfehler entstehen, aber auch durch die Übertragung der Daten. So kann es z.B. vorkommen, dass bei Positionsdaten Werte vorkommen, die außerhalb der Wohnung liegen, oder dass zwei kurz aufeinander folgende Positionen zu weit auseinander liegen, sodass die Geschwindigkeit des Positionswechsels für einen Menschen zu groß wäre.
Solche Fehler werden in der Vorverarbeitung entweder korrigiert, oder die betreffenden Daten werden verworfen.

Sind die Rohdaten unvollständig, müssen die fehlenden Informationen entweder aus den anderen Daten errechnet werden, oder es werden genaue Werte für den Fehler definiert, damit sie beim Datamining berücksichtigt werden können.

Die Vorverarbeitung muss gründlich vollzogen werden. Enthalten die Daten nach der Vorverarbeitung noch Fehler, kann das zu einer fehlerhaften Musterentdeckung führen oder aber auch zu übertrainierten Mustern [Duda et al., 2001, S. 16][Mitchell, 1997, S. 66 ff.]. Diese übertrainierten Muster entsprechen dann nicht mehr einem allgemeinen Muster, sondern nur in dem Fall, bei dem die gleichen Fehler in den Ausgangsdaten enthalten sind.[Fayyad et al., 1996, S.49].

Wie umfangreich die Vorverarbeitung der Daten ausfällt, hängt stark davon ab, wie die Daten zur Verfügung gestellt werden und wie hoch die Qualität der Daten ist.

Transformation

Die vorverarbeiteten Daten müssen nun in eine geeignete Form für das Datamining gebracht werden. Damit der Rechenaufwand für das Datamining möglichst gering gehalten werden kann, sollten die Dimensionen der Daten so klein wie möglich sein. Dazu werden unnötige Werte der Daten entfernt, bzw. mehrere Werte mittels multivariater

Datenanalyse [Backhaus et al., 2006] zu einem Wert zusammengefasst.
Die drei Dimensionen von Positionsdaten können z.B. zu einem Wert zusammengefasst werden, wenn die Koordinaten in Funktionsbereiche umgewandelt werden (s. Kapitel 4.2).

Zusätzlich zu der Verringerung der Dimensionen können die Werte auch in ein diskretes Format gebracht werden. Dazu können z.B. numerische Werte in feste Intervalle aufgeteilt werden, die dann die diskreten Werte darstellen.
Der numerische Wert einer Uhrzeit kann so z.B. in diskrete Zeitabschnitte umgewandelt werden. 7:00 Uhr entspricht dann dem diskreten Wert *morgens* und 15:20 *nachmittags*.

Datamining

Datamining ist der eigentliche Schritt beim KDD, in dem aus den Sensordaten Muster entdeckt werden. Dazu gibt es eine Reihe an Verfahren, die sich teils deutlich unterscheiden. Nachfolgend werden die bekanntesten und verbreitetsten Verfahren und deren wichtigsten Aspekte kurz beschrieben. Detailliertere Beschreibungen finden sich in Ester and Sander [2000], Runkler [2000] und Witten and Frank [2005]. Weitere Verfahren werden im Kapitel 2.3 erläutert.

Man kann die verschiedenen Verfahren grob in zwei Kategorien einteilen: Klassifikation [Oberhauser, 2005] und Clusterbildung Ester and Sander [2000]. Bei der Klassifikation werden Elemente in vorher bekannte Klassen eingeordnet. Dazu werden Regeln festgelegt, wie die Elemente zu interpretieren und auf die Klassen aufzuteilen sind.
Bei der Clusterbildung werden Elemente so zusammengefasst, dass ähnliche Elemente sich in einem Cluster befinden.
In den nachfolgenden Kapiteln wird darauf näher eingegangen.

Clusterbildung

In [Ester and Sander, 2000, S.45] ist das Ziel der Clusterbildung so definiert:

> „Daten (semi-)automatisch so in Kategorien, Klassen oder Gruppen (Cluster) einzuteilen, dass Objekte im gleichen Cluster möglichst ähnlich und Objekte aus verschiedenen Clustern möglichst unähnlich zueinander sind."

Um herauszufinden, wie ähnlich zwei Objekte sind, wird die Distanz zwischen ihnen mittels einer Distanzfunktion berechnet. Je kleiner die Distanzen sind, desto ähnlicher sind sich zwei Objekte. Wie die Distanzfunktion genau aussieht, hängt direkt von den verwendeten Datentypen ab. Bei Koordinaten kann z.B. die euklidische Distanz berechnet werden, bei endlichen Mengen die Anzahl der überlappenden Elemente usw. Die Distanzfunktion kann auch durch eine Distanzmatrix dargestellt werden, in der die Distanzwerte fest vorgegeben werden.

Für die Clusterbildung existieren zahlreiche Varianten, von denen die zwei wichtigsten *k-Means* [Berry and Kogan, 2010, S. 81] und *DBSCAN* [Ester et al., 1996] sind.

k-Means Beim k-Means-Verfahren [Berry and Kogan, 2010, S. 81] werden die Objekte in k Cluster aufgeteilt. Dazu werden im ersten Schritt k Clusterzentren zufällig verteilt. Dann werden die Objekte, anhand einer Distanzfunktion, den Clusterzentren zugeordnet, zu denen sie die geringste Distanz haben. Anschließend werden die Clusterzentren neu berechnet. Dazu wird der Schwerpunkt des jeweiligen Clusters berechnet. Das Verteilen der Objekte auf die Cluster und das Neuberechnen der Clusterzentren wiederholt sich solange, bis entweder die gewünschte Anzahl der Iterationen erreicht ist oder sich die Cluster und Clusterzentren nicht mehr verändern, bzw. die Veränderung einen Schwellwert unterschreitet.

DBSCAN DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [Ester et al., 1996] ist ein dichtebasiertes Clusterverfahren. Dabei werden die Objekte in drei Kategorien unterschieden: Kernobjekte, dichteerreichbare Objekte und Rauschobjekte.

Dazu werden zwei Werte definiert: *minPts* und ϵ. Dabei gibt ϵ an, wie weit zwei Objekte voneinander maximal entfernt sein dürfen, um noch *erreichbar* zu sein. Kernobjekte sind Objekte, die mindestens *minPts* andere Objekte als erreichbare Nachbarn haben. Ein Kernobjekt gilt dann als *dicht*. Mehrere zusammenhängende Kernobjekte bilden einen Cluster.

Dichteerreichbare Objekte sind Objekte, die von einem Kernobjekt eines Clusters erreichbar sind, aber selber nicht dicht sind.

Rauschobjekte sind Sensordaten, die nicht dichte-erreichbar sind und auch selber nicht dicht sind.

Klassifikation

Bei der Klassifikation [Oberhauser, 2005] werden Objekte vorgegebenen Klassen zugeordnet. Dazu werden mit Hilfe von Trainingsdaten Funktionen gebildet, mit denen Objekte den Klassen zugeordnet werden können.

Bayes-Klassifikatoren Bayes-Klassifikatoren [Larose, 2006, S. 204 ff.] basieren auf dem Bayes-Theorem [Behrends, 2013, S. 115 ff.] und bilden die Klassen dadurch, dass die Wahrscheinlichkeit berechnet wird, mit dem ein Objekt zu einer Klasse gehört. Um die Wahrscheinlichkeit zu berechnen, werden die einzelnen Attribute eines Objektes untersucht, ob sie zu einer Klasse gehören. Bei einem naiven Bayes-Klassifikator wird dabei angenommen, dass die Attribute voneinander unabhängig sind.

Entscheidungsbäume Mit Entscheidungsbäumen [Ester and Sander, 2000, S. 663 ff.] werden Objekte in mehreren Stufen, die der Baumtiefe entsprechen, in die Zielkassen aufgeteilt. Dazu werden die Objekte hierarchisch an jedem Knoten auf jeweils eine bestimmte Eigenschaft hin überprüft. So kann z.B. ein Sensorwert zuerst darauf untersucht werden, wo im Raum er erzeugt wurde, wann er erzeugt wurde und anschließend, welchen Wert er aufweist. Die Reihenfolge und die Bedingungen, die ein Sensorwert

erfüllen muss, um einer Klasse zugeordnet zu werden, hängt davon ab, wie die Klassen definiert sind.

Data dredging

Datamining ist bei Statistikern in Verruf geraten. Wenn in Daten lange genug gesucht wird, werden früher oder später Muster entdeckt, die statistisch relevant erscheinen, es aber nicht sind [Fayyad et al., 1996, S.40]. Das fehlerhafte Anwenden von Datamining wird auch als *Data dredging* bezeichnet [Fayyad et al., 1996][Smith, 2002][Ioannidis, 2005]. Daher ist es wichtig, die Algorithmen sorgfältig auszuwählen und die Ergebnisse zu hinterfragen, die durch das Datamining entstehen.

Dazu muss nach dem Datamining das Ergebnis sorgfältig geprüft werden. Die entstandenen Cluster bzw. Klassen müssen klar voneinander abgetrennt und die Objekte korrekt darauf verteilt sein.

Interpretation

Der letzte Schritt der KDD ist die Interpretation [Duda et al., 2001, S. 15] der entdeckten Muster.
Dazu werden die entdeckten Muster daraufhin untersucht, wie gut sie die zugrundeliegenden Daten darstellen und ob sie für eine zukünftige Mustererkennung geeignet sind. Bekannte Stichproben werden mit den entdeckten Mustern verglichen und die Abweichung berechnet. Je mehr Stichproben in den Mustern wiedergefunden werden können und je kleiner die Abweichungen sind, desto besser ist das Ergebnis der KDD.

Sollte das Ergebnis nicht den Erwartungen entsprechen, kann die KDD wiederholt und das Ergebnis verfeinert werden oder die KDD wird mit neuen Parametern neu gestartet.

Ist das Ergebnis zufriedenstellend, werden die entdeckten Muster nun als neues Wissen zur Verfügung gestellt.

Dazu können die Muster in grafischer Form für den Menschen übersichtlich dargestellt werden, als neue Datenbasis für eine weitere Verwendung in anderen Projekten verwendet werden.

2.2 Datenquellen

Eine der wichtigsten Bestandteile dieser Masterarbeit sind die Datenquellen. Um die Daten für die Aktivitätenentdeckung verwenden zu können, werden bestimmte Anforderungen an den Informationsgehalt dieser Datenquellen gestellt [Hildebrand et al., 2011, S. 26ff].
Für diese Masterarbeit müssen die Sensordaten folgende Qualitäten erfüllen:

Verfügbarkeit Die Datenquellen müssen innerhalb des Livingplace verfügbar sein. D.h., dass die Sensordaten von anderen Projekten erzeugt und zur Verfügung gestellt werden müssen.

Präzision Die Präzision der Sensordaten sollte so hoch wie möglich sein. Je höher die Abweichungen der Sensordaten sind, desto schwieriger wird es, in ihnen zuverlässige Muster zu entdecken.

Zuverlässigkeit Die Sensordaten müssen zuverlässig erfasst werden können. Das bedeutet einerseits, dass die Sensoren in einem definierten und kontrollierten Bereich arbeiten, aber auch, dass Aktivitäten des Bewohners tatsächlich entsprechende Sensordaten erzeugen.

Geringe Latenz Der Zeitraum von der Entstehung bis zur Erfassung eines Sensorwertes sollte möglichst gering sein, genauso wie die Erkennung, Verarbeitung und Übertragung der Sensordaten. Ansonsten kann es passieren, dass Sensordaten in einer verfälschten Reihenfolge für die Aktivitätenentdeckung verwendet werden oder, dass der Sensorwert veraltet ist.

Informationsgehalt Ein einzelner Sensorwert sollte einen möglichst hohen Informationsgehalt aufweisen. Die übertragenen Werte sollten daher einen hohen Anteil an Nutzdaten haben. Zusätzlich übertragene Daten der Sensordaten sollten entweder weiter dazu dienen, die Sensordaten besser auswerten zu können, oder auf ein Minimum reduziert werden.

Wiederholbarkeit Die Sensordaten müssen so erzeugt werden, dass Experimente wiederholt werden können. Veränderungen in der Aktivitätenentdeckung können nur dann als Verbesserungen erkannt werden, wenn die zugrundeliegenden Sensordaten immer gleich sind.

Sensoren

In [Schiessle, 1992, S. 14] sind Sensoren folgendermaßen definiert:

> „Der *Sensor* ist ein technisches Bauteil, das aus einem physikalischen nichtelektrischen Meßsignal ein eindeutiges elektrisches Signal erzeugt [...]“.

Sensoren werden dazu verwendet, physikalische Größen wie Temperatur und Helligkeit zu messen, und für eine weitere Verarbeitung qualitativ und quantitativ festzuhalten [Schiessle, 1992] [Hering and Schönfelder, 2012].

Im Livingplace sind bereits verschiedene Projekte realisiert, die mit Sensoren ausgestattet sind, die sich für eine Aktivitätenentdeckung eignen. Dazu gehören das Ubisensesystem für die Positionsbestimmung von Personen und Objekten, das intelligente Bett, welches von Frank Hardenack während seiner Masterarbeit [Hardenack, 2011] entwickelt wurde, und eine intelligente Couch von Mosawer Nurzai [Nurzai, 2013].

Diese Projekte sind bereits eine gute Grundlage für die Aktivitätenentdeckung. Um aber eine detailliertere Beschreibung der Aktivitäten zu erhalten, sind weitere Sensortypen notwendig.
Diese Sensortypen können dann beispielsweise angeben, welche Wasserhähne benutzt werden, wann eine Tür oder ein Schrank geöffnet oder geschlossen wird, uvm.

Sensoren bilden die Datenbasis für die Aktivitätenentdeckung.

Scriptsimulator

Der Scriptsimulator ist in den Vorstudien meines Masterstudiums entstanden [Basener, 2011c] [Basener, 2012]. Der Grund für die Entwicklung des Scriptsimulators ist, dass es im Livingplace sehr schwierig ist, ausreichend reale Sensordaten zu produzieren, die ein typisches Wohnszenario widerspiegeln. Um die gewünschten Sensordaten zu produzieren, müssten zunächst einmal alle benötigten Sensortypen im Livingplace installiert sein. Weiterhin ist es notwendig, dass eine Person für eine längere Zeit im Livingplace wohnt und dort verschiedene Aktivitäten wiederholt ausführt.
Da aber im Livingplace Kommilitonen ständig damit beschäftigt sind, ihre Projekte und Experimente durchzuführen, ist das nicht möglich.

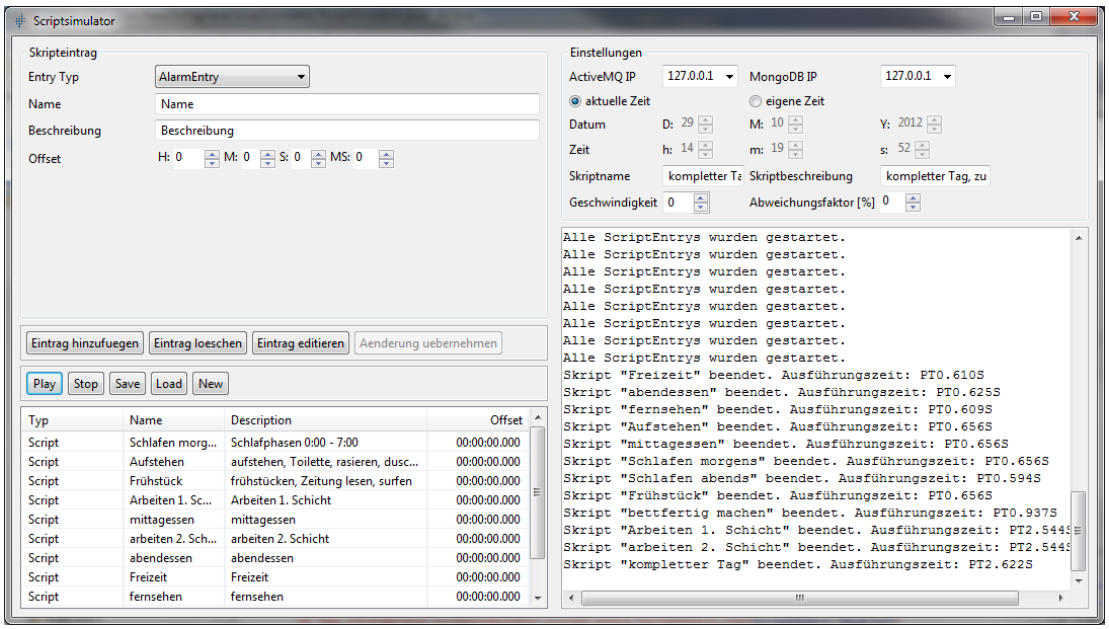


Abbildung 2.2: GUI des Scriptsimulators

Der Scriptsimulator wurde entwickelt, Szenarien für das Livingplace als Skripte zu schreiben, die beim Abspielen die benötigten Sensordaten erzeugen.
Jedes Szenario ist ein einem Skript festgehalten, das, ähnlich wie Regieanweisungen in einem Theater, nacheinander verschiedene Einträge ausführt. Ein Eintrag entspricht dabei einem oder mehreren, logisch zusammenhängenden, Sensorwerten. Z.B. kann

ein Eintrag den einzelnen Sensorwert für das Öffnen einer Tür erzeugen, aber auch die aufeinanderfolgenden Positionsdaten, die beim Durchqueren der Wohnung entstehen.

Im Scriptsimulator sind bereits einige Projekte des Livingplace umgesetzt, sodass deren Sensordaten erzeugt werden können. Zusätzlich erlaubt der Scriptsimulator auch, Sensoren zu simulieren, die kein Gegenstück im Livingplace haben.
Somit lassen sich z.B. Positionsdaten des installierten Ubisensesystems simulieren, sowie auch nicht vorhandene Sensoren für den Wasserverbrauch.

Die Skripte sind so implementiert, dass kleinere Szenarien als Skripte erstellt werden können, die konkreten Aktivitäten entsprechen, z.B. Kaffeekochen oder Schlafen. Diese kleinen Skripte können dann zu größeren Skripten zusammengefasst werden, um größere Zeiträume, wie Tage, Wochen oder Monate, im Livingplace zu simulieren.

Damit die Skripte komfortabel verwendet werden können, bietet der Scriptsimulator die Optionen, sowohl Datum und Uhrzeit, als auch die Abspielgeschwindigkeit des Skriptes zu verändern. Möchte man z.B. ein Szenario abspielen, das in der Nacht stattfindet, kann es zu jedem beliebigem Zeitpunkt gestartet werden, der unabhängig von der simulierten Zeit des Skriptes ist.
Weiterhin kann es sinnvoll sein, längere Skripte mit einer erhöhten Geschwindigkeit abzuspielen. Der Scriptsimulator stellt diese Funktion zur Verfügung und ermöglicht es auch, ein Skript ohne irgendwelche Verzögerungen zwischen den Einträgen abzuspielen. Die Abspieldauer hängt dann nur noch von der Leistungsfähigkeit des verwendeten Computers ab.

Die Genauigkeit der erzeugten Sensordaten kann beliebig eingestellt werden. Dazu kann eine Fehlerrate angegeben werden, die sowohl in die Sensorwerte selbst eingerechnet wird, als auch den Zeitpunkt beeinflusst, wann der Sensorwert erzeugt wird.
Dadurch ist es möglich, Sensorwerte ohne jegliche Fehlerrate zu erzeugen, um immer präzise Ausgangsdaten zu erzeugen, aber auch fehlerhafte Sensoren zu simulieren.

Eine Simulation [Hartmann, 2005] kann die Realität nie exakt wiedergeben, sondern kann sich dieser nur möglichst genau annähern. Dabei ist es wichtig festzulegen, welche Aspekte der Realität simuliert werden sollen und welche weggelassen werden.

2.3 Related Works

Mit dem Auffinden von Aktivitäten in Sensordaten haben sich bereits mehrere Projekte beschäftigt. Nachfolgend sind drei dieser Projekte aufgeführt, und es werden die unterschiedlichen Herangehensweisen erklärt. Mit diesen Projekten habe ich mich bereits im Vorfeld dieser Masterarbeit auseinandergesetzt. Der folgende Abschnitt ist aus meinem Seminarbericht [Basener, 2011b] entnommen und wurde mit einer Bewertung der Verfahren erweitert.

2.3.1 BehaviourScope

Das Projekt BehaviourScope[1] der Universität Yale in den USA wurde im Zeitraum 2006 bis 2009 im Enelab[2] durchgeführt. An diesem Projekt waren u.a. Athanasios Bamis, Jia Fang, Andreas Savvides und Dimitrios Lymberopoulos beteiligt, die sich mit Verhaltensmustern in Sensornetzwerken beschäftigt haben.

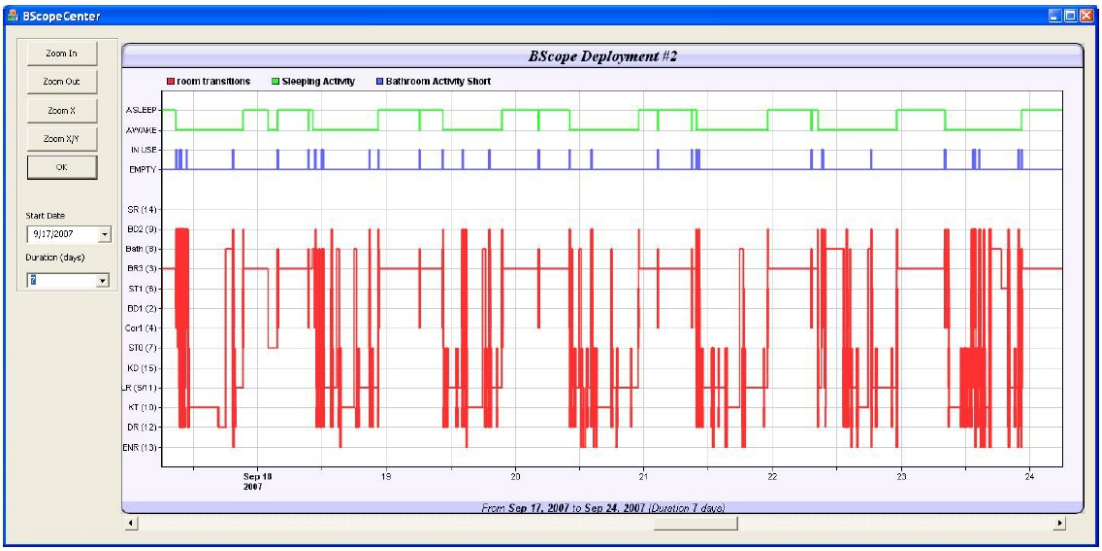


Abbildung 2.3: mehrere Eventstreams [Lymberopoulos et al., 2008]

[1] http://bscope.eng.yale.edu/
[2] http://enalab.eng.yale.edu/

In diesem Projekt wurde eine Versuchswohnung mit PIR-Sensoren (Bewegungsmelder) bestückt, die von einer älteren Person bewohnt wurde. Durch die Bewegungsmelder konnte aufgezeichnet werden, wann sich die Person zu welchem Zeitpunkt in welchem Teil der Wohnung aufhielt [Lymberopoulos et al., 2008].
Die einzelnen Positionsdaten werden als Events chronologisch zu einen Eventstream aneinander gereiht. Aus diesem Eventstream werden dann die Aktivitäten des Bewohners erkannt.

Event und Eventstream

Ein Event ist als ein Tupel definiert [Lymberopoulos et al., 2008, S. 3].

$$\text{Eventtupel} = \{P, T, D\}$$

P ist die ID des Sensors, T der Zeitstempel und D die Dauer des Events in Minuten. Ein einzelnes Event enthält somit die Informationen welcher Bewegungssensor aktiviert wurde, und in welchem Zeitraum dies geschah.
Ein Eventstream ist wie folgt definiert:

$$E(T_{\text{start}}, T_{\text{stop}}) = \{(P_j^i, T_j^i, D_j^i) | (P_j^i, T_j^i, D_j^i) \in S^i, T_{\text{start}} \leq T_j^i \leq T_{\text{stop}} \forall i, j\}$$

T_{start} und T_{stop} geben die obere bzw. untere Schranke für das Zeitintervall des Eventstreams an. Eventtupel, deren Start- und Stopzeiten innerhalb dieses Intervalls liegen, gehören zu diesem Eventstream.
Ein Beispiel, wie ein Intervall ausehen kann ist folgendes:

$$E(\text{10pm}, \text{10am}) =< \{\text{Bed}, \text{11:00pm}, 300\}, \{\text{Bath}, \text{4:00am}, 5\}, \{\text{Bed}, \text{4:05am}, 300\},$$

$$\{\text{Bath}, \text{9:05pm}, 10\}, \{\text{Kitchen}, \text{09:15pm}, 30\} >$$

Durch die Struktur der Eventstreams kann sehr gut nachvollzogen werden, wo sich der Bewohner in der Wohnung aufgehalten hat und wie lange er das getan hat.

zeitliche Abstraktion

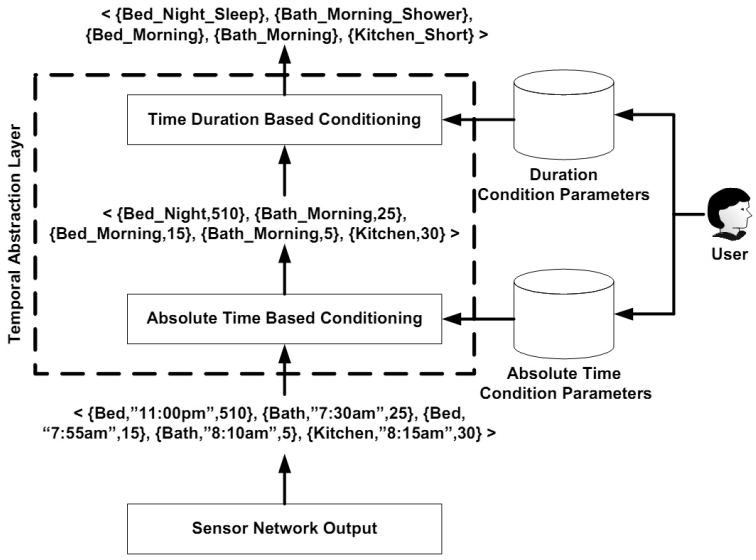


Abbildung 2.4: zeitliche Abstraktion [Lymberopoulos et al., 2008]

Da es unwahrscheinlich ist, dass der Bewohner die gleichen Aktionen zu exakt der gleichen Zeit bzw. im exakt gleichen Zeitraum ausführt, wird ein Mechanismus benötigt, um Abweichungen auszugleichen. Hierzu werden die Eventtupel auf fest definierte Aktionen abstrahiert, die eine einfache zeitliche Einordnung ermöglichen (s. Abbildung 2.4).

Zuerst wird der Zeitstempel eines Sensortupels T_0 auf eine absolute Zeitangabe abgebildet. Z.b. wird die Zeitangabe *11:00pm* aus dem Tupel $T_0 = \{\text{Bed}, 11{:}00\text{pm}, 510\}$ mit der ID *Bed* zu *Bed_Night* verknüpft. Daraus ergibt sich ein neues Tupel $T_1 = \{\text{Bed\_Night}, 510\}$.

Anschließend wird mittels der Dauer auf die eigentliche Aktion geschlossen. Aus dem Tupel T_1 entsteht das endgültige Tupel $T_2 = \{\text{Bed\_Night\_Sleep}\}$

Wie die zeitlichen Parameter zu interpretieren sind, wird vom Benutzer manuell vorgegeben.

Apriori-Algorithmus

Um wiederkehrende Verhaltensmuster in den Eventstreams zu erkennen, wird im BehaviourScope-Projekt der Apriori-Algorithmus verwendet. In Listing 2.1 ist der Apriori-Algorithmus aufgeführt.

```
1  Apriori(D)
2
3  Eingabe: Datenbasis D
4  Ausgabe: Menge heaufiger Itemmengen
5
6  L_1 := {häufige 1-Itemmenge}
7  k := 2
8  while L_{k-1} ≠ ∅ do
9    C_k := AprioriGen(L_{k-1})            //neue Kandidatenmenge
10   for all Transaktionen t ∈ D do
11     C_t := {c ∈ C_k | c ⊆ t}             //in t enthaltene Kandidatenmenge
12     for all Kandidaten c ∈ C_t do
13       c.count := c.count +1
14     end for
15   end for
16   L_k := {c ∈ C_k | c.count ≥ |D| * minsupp
17   k := k + 1
18 end while
19 return ⋃_k L_k
```

Listing 2.1: Apriori-Algorithmus (Beierle and Kern-Isberner [2006], S. 148)

Durch den Apriori-Algorithmus werden die am häufigsten vorkommenden Eventsequenzen gefunden. Dabei wird mit einem einzelnen Event begonnen und dieser iterativ um weitere Events erweitert. Ob eine erweiterte Sequenz übernommen wird, hängt davon ab, ob die in dieser Sequenz enthaltenen Subsequenzen eine Mindesthäufigkeit aufweisen. Eine detaillierte Beschreibung des Apriori-Algorithmus ist in [Beierle and Kern-Isberner, 2006, S. 147 f.] zu finden.

Die so gefundenen Sequenzen bilden die Basis für die Aktivitätenerkennung in neuen Sensordaten.

2.3.2 Aware Home

Das Aware-Home-Projekt[3] des Georgia Institute of Technology in den USA beschäftigt sich seit 1999 mit der Verknüpfung von moderner Informationstechnologie mit dem alltäglichen Wohnen.
In diesem Projekt sind u.a. Brian Jones, Nachiketas Ramanujam, Lana Yarosh und Taejung Yun aktiv, die sich mit dem Erlernen von Verhaltensmustern beschäftigt haben.

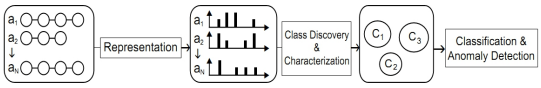


Abbildung 2.5: Framework [Hamid, 2008]

Die Verhaltensmustererkennung basiert in diesem Projekt auf Verhaltensklassen und deren Verknüpfung in Graphen.
In diesem Verfahren wird davon ausgegangen, dass einzelne Aktivitäten klar als solche angeben sind. Eine Aktivität besteht dabei aus mehreren diskreten Events. Diese Aktivitäten werden nun auf deren Eigenschaften hin untersucht. Ähnliche Aktivitäten werden zu Aktivitätsklassen zusammengefasst. Mit diesen Klassen können dann neue Aktivitäten eingeordnet werden.
Der genaue Ablauf des in Abbildung 2.5 dargestellte Frameworks wird in den nachfolgenden Abschnitten beschrieben:

n-Grams

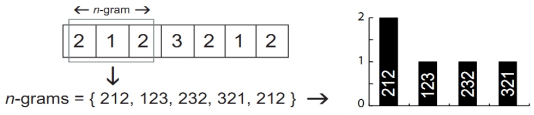


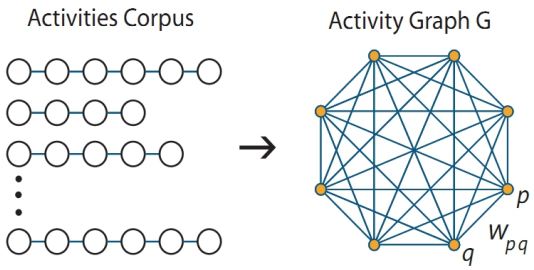
Abbildung 2.6: n-grams [Hamid, 2008]

Die Aktivitäten stellen Sequenzen von diskreten Events dar. Über diese Sequenzen können Fenster mit der Größe n geschoben werden, die n-Grams genannt werden. Es

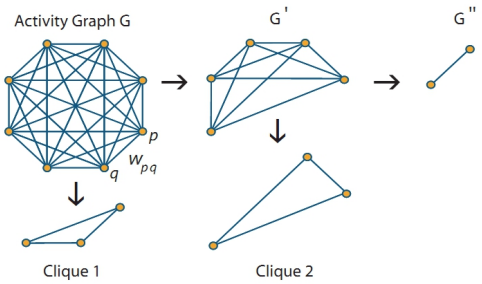
[3] http://awarehome.imtc.gatech.edu/

werden sämtliche n-Grams einer Aktivität erstellt. Aus den gefundenen n-Grams wird dann ein Histogramm erstellt, das angibt, welches n-Gram wie häufig vorkommt. In Abbildung 2.6 ist zu sehen, wie ein n-Gram und das dazugehörige Histogramm für die Größe 3 aussehen können.

Aktivitätsgraph



(a) Aktivitätsgraph



(b) Cliquen/Aktivitätsklassen

Abbildung 2.7: [Hamid, 2008]

Mit Hilfe der n-Grams und der daraus berechneten Histogramme wird nun ein ungerichteter und gewichteter Aktivitätsgraph erstellt. Dazu wird für alle Aktivitäten berechnet, wie sehr sie den anderen Aktivitäten ähneln.
Für die Ähnlichkeit zweier Aktivitäten wird untersucht, ob die n-Grams nur in einer Aktivität vorkommen oder in beiden. Kommt ein n-Gram nur in einer Aktivität vor, ergibt sich daraus ein Unterschied. Kommt jedoch ein n-Gram in beiden Aktivitäten vor, wird untersucht, wie häufig das n-Gram jeweils vorkommt. Die genaue Formel ist in

[Hamid, 2008, S. 31] in der Gleichung (1) angegeben.
Aus den ermittelten Ähnlichkeiten der Aktivitäten kann ein Aktivitätsgraph erstellt werden. Abbildung 2.7(a) veranschaulicht dies. Die Knoten sind hierbei die Aktivitäten, die Kanten geben die Ähnlichkeit zwischen den einzelnen Aktivitäten wieder.

Aktivitätsklassen

Im anschließenden Schritt wird der ermittelte Aktivitätsgraph reduziert. Hierbei werden ähnliche Knoten zu Aktivitätsklassen zusammengefasst. Dazu werden in dem Aktivitätsgraphen iterativ maximale Cliquen gesucht. Um den Rechenaufwand klein zu halten, werden hierfür dominante Sets verwendet. Die genaue Berechnung ist in [Hamid, 2008, S. 33ff.] zu finden.

Jetzt können neue Aktivitätsinstanzen den berechneten Aktivitätsklassen zugeordnet werden. Dazu wird die neue Aktivität mit allen in einer Aktivitätsklasse enthaltenen Aktivitäten verglichen und die Ähnlichkeiten berechnet. Die aufsummierten und gewichteten Ähnlichkeiten zu den einzelnen Aktivitäten einer Klasse geben dann die Ähnlichkeit der neuen Aktivität zu der gesamten Aktivitätsklasse an (Abb. 2.7(b)).

2.3.3 CASAS

Das CASAS-Smart-Home-Projekt[4] der Washington State University in den USA besteht seit 2005. Diane J. Cook, Lawrence Holder, Behrooz Shirazi, Maureen Schmitter-Edgecombe und Teddy Yap beschäftigen sich im Rahmen dieses Projektes mit dem Erkennen und Erlernen von Verhaltensmustern in Sensordaten.

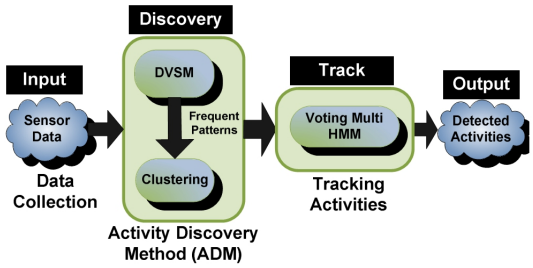


Abbildung 2.8: Ablauf der Aktivitätenentdeckung [Rashidi et al., 2011]

Dort werden die Sensordaten mit Hilfe der Activity Discovery Method (ADM) untersucht. In der ADM werden häufig wiederkehrende Muster in den Sensordaten erkannt und abgespeichert. Damit das anschließende Erkennen von Mustern besser funktioniert, werden die erkannten Muster zu Clustern zusammengefügt. Mit diesen Clustern wird dann die eigentliche Mustererkennung für neue Sensordaten durchgeführt.

Activity Discovery Method

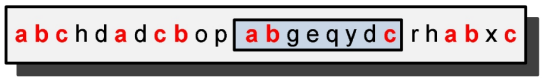


Abbildung 2.9: Eventstreams [Rashidi et al., 2011]

Die Activity Discovery Method (ADM) besteht aus zwei Teilen. Im ersten Teil werden die eingehenden Sensordaten auf Muster untersucht. Dies geschieht mit Hilfe des Discontinuous Varied-order Sequential Miners (DVSM). Im zweiten Teil werden ähnliche Verhaltensmuster zu Clustern zusammengefasst.

[4] http://ailab.wsu.edu/casas/

Der DVSM arbeitet folgendermaßen: Zuerst werden alle Einzelevents eines Datensatzes D auf ihre Häufigkeit hin geprüft. Die häufigsten Events werden in einem neuen reduzierten Datensatz D_r zusammengefasst. Welche die häufigsten Events sind, wird von dem Benutzer durch den Wert α vorgegeben. Über D_r wird nun ein Fenster der Größe 2 geschoben, um so Muster mit der Länge 2 zu erhalten. Anschließend werden die erkannten Muster um ihre Prä- und Suffixe erweitert.
Für jedes erkannte Muster werden zusätzlich häufig vorkommende Variationen gesucht. Diese Variationen werden mit Hilfe der Levenshtein-Distanz berechnet [Merkl and Waack, 2005, S. 93ff.]. Wie in Abbildung 2.9 zu sehen ist, kann ein Muster $\{a, b, c\}$ neben seiner einfachen Form noch in anderen Varianten auftauchen. So können einzelne Events vertauscht sein ($\{a, c, b\}$), oder aber fremde Events tauchen innerhalb einer Variante auf ($\{a, b, e, c\}$). Zudem kann auch beides gleichzeitig eintreten ($\{a, c, e, b\}$). Details sind in [Rashidi et al., 2011, S. 4f.] in den Formeln (1) bis (9) beschrieben.

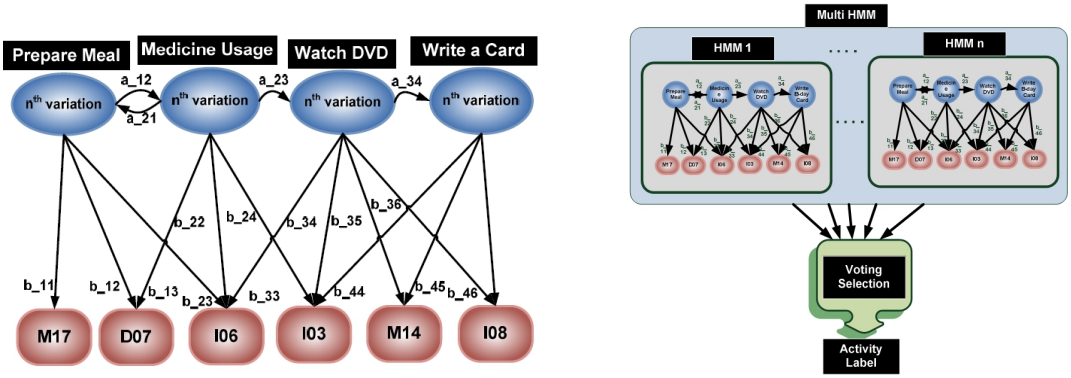
Clustering

Im nächsten Schritt der ADM werden die entdeckten Muster zu Clustern zusammengefasst. Zu diesem Zweck werden die einzelnen Events um zusätzliche Informationen erweitert. Diese Zusatzinformationen können der Typ eines Sensors oder die Dauer eines Events sein. Diese erweiterten Events bilden Zustände, die für das Clustering verwendet werden. Dazu wird, ähnlich wie in der DVSM, die Distanz zwischen den Zuständen berechnet [Rashidi et al., 2011, S. 5f.].
Zustände, die sich ähneln, werden dann zu Clustern zusammengefasst. Der Zustand mit den geringsten Distanzen zu den anderen Zuständen eines Clusters wird als Clusterzentrum verwendet. Wie viele Cluster erzeugt werden, wird vom Benutzer vorgegeben. Die erzeugten Cluster spiegeln die Aktivitäten wieder, die zukünftig erkannt werden können.

Tracking

Nachdem aus den Sensordaten entsprechende Verhaltensmuster gefunden wurden, wird im Tracking das erlernte Modell auf neue Sensordaten angewendet. Um herauszufin-

den, zu welchen Aktivitäten die neuen Sensordaten gehören, werden Hidden-Markov-Modelle (HMM) [Fink, 2003] verwendet.



(a) Hidden-Markov-Modell (b) Voting

Abbildung 2.10: [Rashidi et al., 2011]

Die Aktivitäten bilden hierbei die versteckten Zustände des HMM.
Damit nun die richtige Aktivität erkannt werden kann, werden für alle n Variationen n HMM aufgestellt. Neue Sensordaten werden in alle HMM eingegeben und jeweils die wahrscheinlichste Aktivität wird berechnet. Danach wird die Aktivität ausgewählt, die in allen HMM am häufigsten als aktuell wahrscheinlichste Aktivität berechnet wurde [Rashidi et al., 2011, S. 6f.].

2.3.4 Bewertung der Verfahren

Jedes der vorgestellten Verfahren hat seine Vor- und Nachteile, die nachfolgend erläutert werden. Zudem gibt es Unterschiede bei den verwendeten Sensoren, deren Verarbeitung und der Aktivitätenentdeckung.

Im Projekt Behaviorscope werden lediglich Bewegungsmelder als Sensoren verwendet, mit denen herausgefunden werden kann, wann ein Bewohner einen Raum betritt und wie lange er sich dort aufhält.
Im Casas Projekt geben die Sensoren Zustände an. Als Sensoren werden hier auch Bewegungsmelder verwendet, sowie Sensoren, die angeben, ob Türen geöffnet bzw. geschlossen werden. Zusätzlich ist ein Sessel mit einem Sensor ausgestattet, der erkennt, wenn sich der Bewohner hinsetzt oder wieder aufsteht.
Das AwareHome Projekt untersucht zwei unterschiedliche Szenarien mit unterschiedlichen Sensoren. Im ersten Szenario wurde die Ladebucht eines Geschäftes mit zwei Kameras ausgestattet, die bestimmte Bewegungen der Lieferanten und Mitarbeiter erkennen. Im zweiten Szenario wurde eine Wohnung mit Matten ausgestattet, die erkennen konnten, wann der Bewohner über sie rüber läuft und so die Position des Bewohners in der Wohnung ermittelt.

In meiner Masterarbeit werde ich ein deutlich größeres Angebot an unterschiedlichen Sensoren verwenden. Ich möchte damit herausfinden, ob mit einer größeren Anzahl verschiedener Sensoren eine bessere Aktivitätenentdeckung erzielt werden kann.

Bei der Aktivitätenentdeckung verwenden die Projekte Behaviourscope und CASAS zunächst den Apriori-Algorithmus. Im Bahaviourscope Projekt werden die entdeckten Muster mit vordefinierten zeitlichen Abstraktionen verknüpft, um so die Aktivitäten des Bewohners zu erstellen.
Im CASAS Projekt werden die gefundenen Muster zu Clustern zusammengefügt, um so Muster zusammenzufassen, die die gleiche Aktivität repräsentieren, aber in ihrem Aufbau leichte Unterschiede aufweisen.
Im AwareHome Projekt wird davon ausgegangen, dass bekannt ist, mit welchem Event eine Aktivität beginnt und mit welchem sie aufhört. Hier besteht die Aktivitätenentde-

ckung daraus, mit dem bekannten Vorwissen, die unterschiedlichen Ausführungen der Aktivitäten zu erfassen und mit Hilfe ihrer Histogramme Klassen zu bilden.

In den drei Verfahren werden unterschiedliche Annahmen getroffen, die vor der Aktivitätenentdeckung bekannt sein müssen, z.B. die Anzahl der zu entdeckenden Aktivitäten oder ihr Start und Ende. In meiner Masterarbeit möchte ich ein Verfahren entwickeln, dass ohne diese Vorgaben auskommt.
Am geeignetsten erscheint mir das Verfahren des CASAS Projektes. Hier werden als Vorgaben die Anzahl der zu entdeckenden Aktivitäten und die Mindesthäufigkeiten von Mustern gefordert. Zudem ist es hier von vornherein vorgesehen, auch Varianten von Aktivitäten zu entdecken.

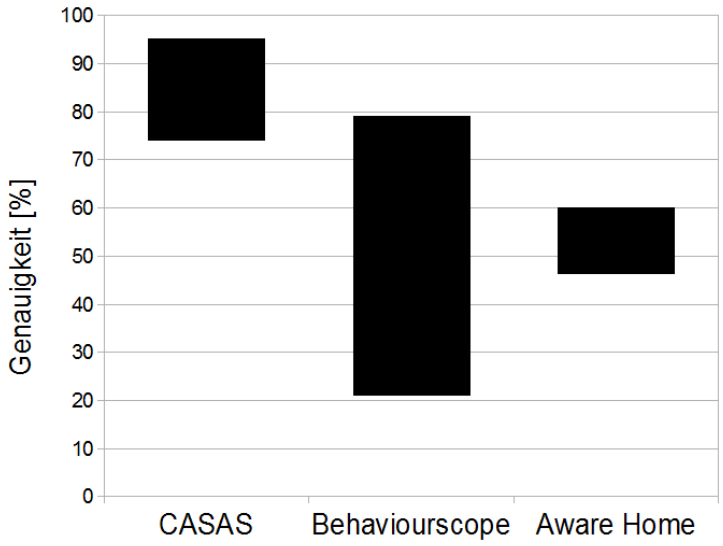


Abbildung 2.11: Genauigkeit und Schwankung der vergleichenden Arbeiten

Die Genauigkeiten der drei Verfahren unterscheiden sich stark voneinander und schwanken je nach Qualität der Sensordaten. Im Behaviorscope-Projekt liegt die Genauigkeit der entdeckten Muster zwischen 21,0% und 79,0% [Lymberopoulos et al., 2008, S. 6 ff.], im AwareHome-Projekt liegt sie bei 46,2% und 60,1% [Hamid, 2008, S. 62 ff.] und im CASAS-Projekt bei 73,8 und 95,2% [Rashidi et al., 2011, S. 9 ff.].

2.4 Fazit und Zielsetzung

In diesem Kapitel habe ich die Grundlagen erläutert, die für eine Aktivitätenentdeckung notwendig sind. Weiterhin habe ich drei Projekte vorgestellt, die bereits erfolgreich eine Aktivitätenentdeckung implementiert haben.

Unter Berücksichtigung dieser Grundlagen können für diese Masterarbeit folgende Ziele definiert werden:

Für die Aktivitätenentdeckung wird Knowledge Discovery in Databases verwendet. Zunächst muss eine Datenbasis geschaffen werden. Dazu müssen die zu verwendenden Sensoren ausgewählt werden und es muss festgelegt werden, wie die Sensordaten gesammelt und für die Aktivitätenentdeckung vorverarbeitet werden.
Anschließend muss die eigentliche Aktivitätenentdeckung entworfen und implementiert werden. Da die Aktivitäten ohne Vorgaben entdeckt werden sollen, wird für das Datamining der k-Means-Algorithmus für eine Clusterbildung eingesetzt.
Zum Schluss der KDD werden die entdeckten Aktivitäten mit zusätzlichen Informationen erweitert und über die Infrastruktur des Livingplace bereitgestellt.

Die Ergebnisse der Aktivitätenentdeckung müssen sich mit den Ergebnissen aus den vergleichenden Arbeiten messen lassen. Dabei sind vor allem das CASAS-Projekt und das BehaviourScope-Projekt interessant, da deren Ergebnisse hohe Werte erreichen.
Weiterhin ist es sinnvoll herauszufinden, wie der Aufbau der entdeckten Aktivitäten ist und wie deren Komplexität Einfluss auf das Ergebnis hat. Darunter fällt auch, wie lang Aktivitäten sein können, damit sie zuverlässig erkannt werden, bzw. wie sich die Zusammensetzung unterschiedlicher Sensoren auswirkt.

Das Verfahren soll sich in die bestehende Infrastruktur des Livingplace integrieren. Das bedeutet, dass die vorhandenen Ressourcen, wie Sensoren, genutzt werden und dass die entstandenen Ergebnisse innerhalb des Livingplace zur Verfügung gestellt werden.

Ein weiteres Ziel der Masterarbeit ist, dass sich das entwickelte Verfahren erweitern lässt. Dazu zählt, dass weitere Sensoren zu der Aktivitätenentdeckung hinzugefügt werden können, aber auch, dass das Verfahren für andere Aufgabenstellungen verwendet werden kann, z.B. für die Aktivitätenentdeckung von Personen in Städten.

3 Design

In diesem Kapitel stelle ich mich den Herausforderungen, die in der Diskussion im vorhergehenden Kapitel 2 herausgearbeitet wurden.

Zunächst beschreibe ich die Infrastruktur und die Sensoren, die ich im Rahmen dieser Masterarbeit verwendet habe, anschließend werden die Datenstrukturen und der Programmablauf beschrieben.

3.1 Infrastruktur

Im Rahmen meiner Masterarbeit verwende ich die Infrastruktur des Livingplace, um Sensordaten zu erzeugen, zu verschicken und zu empfangen und um die entdeckten Muster bereit zu stellen.
Sören Voßkuhl und Kjel Otto haben in ihrem Projekt [Otto and Voskuhl] maßgeblich dazu beigetragen, die Infrastruktur für das Livingplace zu definieren und aufzubauen. Dazu haben sie mit ActiveMQ [Snyder et al., 2011] eine Middleware bereitgestellt, mit dessen Hilfe die verschiedenen Projekte im Livingplace miteinander kommunizieren können.

Das ActiveMQ bietet zwei Möglichkeiten, wie Nachrichten verschickt werden können: Über Queues [Knuth, 2002, S. 238 ff.] oder über einen Publish-Subscribe-Mechanismus [Gamma, 2001, S. 287 ff.].
Queues dienen dazu, dass ein Empfänger Nachrichten von mehreren Sendern empfangen kann. Dazu werden die Nachrichten von den Sendern an die Queue gesendet und dort, in der Reihenfolge des Eintreffens, gespeichert. Der Empfänger kann nun die

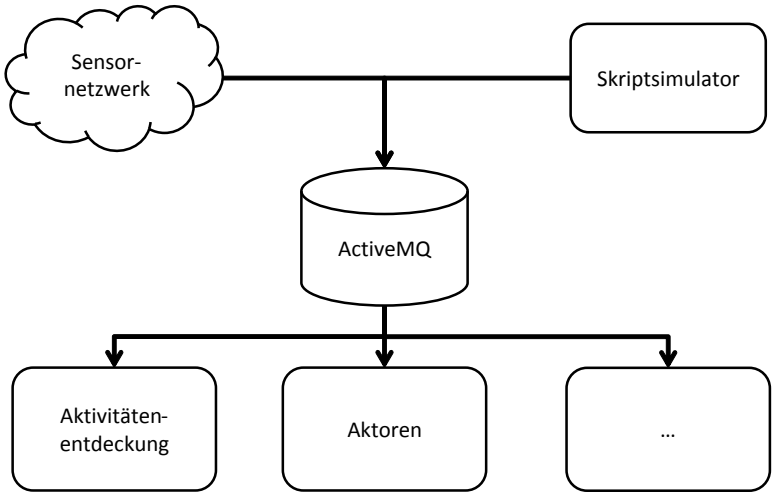


Abbildung 3.1: Datenfluss und Infrastruktur im Livingplace

eingegangenen Nachrichten der Reihe nach abarbeiten.
Mit dem Publish-Subscribe-Muster kann ein Empfänger automatisch Nachrichten von mehreren Sendern empfangen. Dazu benachrichtigt der Empfänger alle Sender, von denen er Nachrichten erhalten möchte, dass sie ihm die Nachrichten automatisch zuschicken, sobald welche erzeugt werden.
Das Publish-Subscribe-Muster bietet sich für den Austausch von Sensorwerten an. Die Sensoren müssen nicht ständig abgefragt werden, ob sie einen neuen Wert erzeugt haben. Stattdessen wird der Empfänger automatisch benachrichtigt.

ActiveMQ verwende ich dazu, die erzeugten Sensordaten zu versenden und zu empfangen. Nach der Aktivitätenentdeckung werden die Aktivitäten über ActiveMQ anderen Projekten zur Verfügung gestellt.

Für den Austausch von Nachrichten wird das JSON-Format [Rieber, 2009, S. 658 ff.] verwendet. JSON ist ein textbasiertes Datenformat, in dem gängige Datenformate wie numerische Werte und Zeichenketten ohne großen Overhead gespeichert werden können. JSON ermöglicht es, die Daten in einem sowohl menschen- als auch maschinenlesbaren Format zu versenden. Dazu werden die Werte als Attribut-Werte-Paare gespeichert.
JSON ist weit verbreitet und wird überwiegend für die Kommunikation in Netzwerken verwendet. Wie bei ActiveMQ existieren Implementationen für zahlreiche Programmier-

sprachen. Durch den relativ einfachen Aufbau der JSON-Nachrichten ist das Erzeugen von Nachrichten und das Parsen der Nachrichten auch ohne gesonderte Bibliotheken möglich.
Für diese Masterarbeit wird JSON dazu verwendet, die Sensordaten zu versenden und am Ende der Aktivitätenentdeckung die Aktivitäten bereitzustellen.

Eine weitere Komponente, die allen Projekten zur Verfügung steht, ist die MongoDB Datenbank [HOL, 2012] [BOE, 2010]. Mit dieser Datenbank können alle Nachrichten, die über das ActiveMQ versendet werden, persistent gespeichert werden. Die MongoDB wird für diese Masterarbeit nicht verwendet. Durch den Einsatz des Scriptsimulators und die daraus entstehende Simulation von Wohnsituationen würden die Zeitstempel der Datenbank nicht mit denen der Simulation übereinstimmen. Eine Anpassung meines Programmes an eine Verwendung der MongoDB, würde nur einen unnötigen Aufwand bedeuten, der in keinem Verhältnis zum Nutzen steht.
Stattdessen wird eine dateibasierte Ablage verwendet. Details dazu sind in den Kapiteln 3.2 und 4.1 beschrieben.

Die Verwendung von ActiveMQ und JSON ist für die Funktionalität der Aktivitätenentdeckung nicht notwendig. Um aber eine reibungslose Interoperabilität mit anderen Projekten zu gewährleisten und so den Nutzen der Aktivitätenentdeckung zu steigern, lohnt sich der Mehraufwand.

3.2 Sensoren

Für die Aktivitätenentdeckung werden Daten benötigt, die in dieser Masterarbeit aus Sensoren gewonnen werden. Die Sensoren sind dabei so ausgewählt, dass sie Aufschluss über die Aktivitäten des Bewohners geben. Jeder Sensorwert ist daher das Ergebnis einer direkten Aktion des Bewohners. Ereignisse und Daten, die keine Rückschlüsse auf die Aktivitäten des Bewohners geben können, z.B. die Außentemperatur der Wohnung, fließen nicht in die Aktivitätenentdeckung ein.

Da das Livingplace, auf Grund seines experimentellen Charakters, keine praktikable Möglichkeit bietet, reale Daten zu erzeugen, die aus einer echten Wohnsituation entstan-

den sind, verwende ich den Scriptsimulator als Datenquelle. Der Scriptsimulator ist im Kapitel 2.2 und in [Basener, 2011c] und [Basener, 2012] näher erläutert.

Die Sensoren sind die einzige Datenquelle für die Aktivitätenentdeckung und müssen daher ausreichend Informationen enthalten, damit die Aktivitäten korrekt entdeckt werden können. Die Sensordaten, die vom Scriptsimulator erzeugt werden, enthalten alle Daten, wie sie von den Projekten des Livingplace erzeugt werden. Sensortypen, die nur simuliert werden und kein reales Gegenstück im Livingplace haben, sind so implementiert, dass sie realistische Sensorwerte erzeugen.
Neben nützlichen Daten enthalten die Sensorwerte auch überflüssige Informationen, die für die Aktivitätenentdeckung nicht benötigt werden. Diese Daten müssen dann in der Vorverarbeitung herausgefiltert werden.

Die Sensordaten werden im JSON-Format über das ActiveMQ gesendet. Jeder Sensorwert wird mit einem Zeitstempel versehen, der den Zeitpunkt angibt, an dem der Sensorwert erzeugt wurde. Dabei handelt es sich um den Zeitpunkt innerhalb des simulierten Szenarios, nicht um den realen Zeitpunkt bei der Ausführung des Szenarios. Der Zeitstempel entspricht der Unixzeit [Matthews] mit einer Genauigkeit von einer Millisekunde.

Die Sensoren senden ihre Daten auf jeweils einem eigenen Topic über das ActiveMQ. Die Sensordaten werden in zeitlicher Reihenfolge empfangen und gespeichert.
Dazu liest ein Empfangsprogramm auf sämtlichen Topics mit und speichert die empfangenen Werte in einer Textdatei ab. In der Textdatei werden die einzelnen Sensordaten jeweils in eine eigene Zeile geschrieben und die Werte der Sensordaten werden durch ein Semikolon abgetrennt. Jede Zeile wird von dem Namen des Sensors angeführt, danach folgt ein Zeitstempel und anschließend der eigentliche Wert des Sensors.

$$[\text{Name des Senors}], [\text{Zeitstempel}], [\text{Wert1}], [Wert2]...$$

Beispiel für Positionsdaten:

$$\text{Ubisense}; 1369770766660; 9.73; 5.46$$

Die Einheiten der Sensorwerte werden als bekannt vorausgesetzt und nicht mitgesendet.
Die eingelesenen Sensordaten werden im Weiteren auch als Datenstrom bezeichnet.

3.3 Datenstrukturen

3.3.1 Event

Jeder einzelne Sensorwert, der empfangen wird, stellt ein Event dar. Ein Event ist somit ein diskretes Vorkommen von Sensordaten und stellt die kleinste Einheit für die Aktivitätenentdeckung dar [Hamid, 2008, S. 8][Lymberopoulos et al., 2008, S. 2] [Rashidi et al., 2011, S. 3].

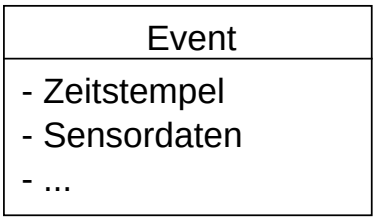


Abbildung 3.2: Event

Ein Event wird mindestens durch einen Zeitstempel und die Werte des jeweiligen Sensors beschrieben. Je nach Sensortyp können zusätzliche Informationen übertragen werden, die für die Aktivitätenentdeckung unwichtig sind. Diese Informationen werden bei der Entdeckung ignoriert und später für die Annotation der Aktivitäten verwendet (s. Kapitel 4.6.6).

Beim Erzeugen der Events werden zugrunde liegende Sensordaten vorverarbeitet. Welche Maßnahmen vorgenommen werden, hängt dabei vom jeweiligen Sensortyp ab. Bei numerischen Sensordaten werden die aufeinanderfolgenden Werte auf Plausibilität hin überprüft, z.B., ob sie innerhalb der erlaubten Wertebereiche liegen, oder ob die Werte Ausreißer bzw. Messfehler beinhalten. Bei den Positionsdaten werden z.B. Sensorwerte ignoriert, die außerhalb des Functional Spaces des Livingplace liegen, oder die innerhalb kurzer Zeit eine ungewöhnlich große Distanz zu der letzten Position aufweisen.

Details, wie Events in der Aktivitätenentdeckung verwendet werden, finden sich im Kapitel 4.

3.3.2 Sequenzen

Eine Sequenz besteht aus einem oder mehreren Events [Hamid, 2008, S. 8][Lymberopoulos et al., 2008, S. 2] [Rashidi et al., 2011, S. 3]. Dabei kann die Sequenz eine beliebige Länge annehmen und aus unterschiedlichen Eventtypen zusammengesetzt sein.

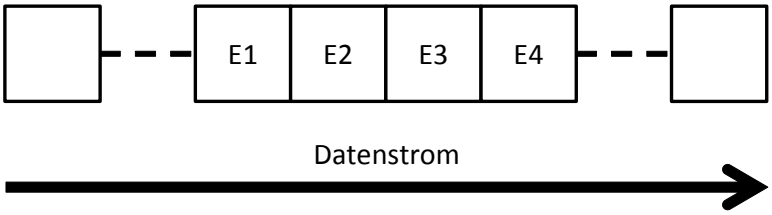


Abbildung 3.3: Datenstrom und Sequenz

Die Sequenz setzt sich immer aus zeitlich aufeinanderfolgenden Events zusammen, die ohne Lücken aus dem ursprünglichen Datenstrom gebildet werden. D.h., dass jede Sequenz einem Ausschnitt aus dem Datenstrom entspricht.

Sequenzen bilden die Grundlage für die Muster, die die entdeckten Aktivitäten beschreiben.

Neben den einzelnen Events enthält eine Sequenz auch den Index im Datenstrom, wo das erste Event der Sequenz zum ersten Mal auftaucht.

3.3.3 Muster

Ein Muster enthält eine Sequenz von Events und ergänzt diese um weitere Informationen.
Zu diesen Informationen zählen die Anzahl, wie häufig diese Sequenz im Datenstrom vorkommt, und eine Liste der Indizes, wo die Sequenz im Datenstrom vorkommt. Zusätzlich werden der Wert des Musters berechnet, die durchschnittliche Zeit aller Zeitstempel der enthaltenen Events und die Standardabweichung von dieser Durchschnittszeit [Rashidi et al., 2011, S. 5].

Für die Berechnung der Durchschnittszeit werden einfach alle Zeitstempel der Events addiert und durch die Anzahl der Events in der Sequenz geteilt.

$$t_{\text{Durchschnitt}} = \frac{\sum \text{Zeitstempel}}{\text{Anzahl der Events}}$$

Die Standardabweichung der Zeit berechnet sich folgendermaßen:

$$t_{\text{Standardabweichung}} = \sqrt{\frac{\sum(\text{Zeitstempel} - t_{\text{Durchschnitt}})^2}{\text{Anzahl der Events}}}$$

Für den Wert eines Musters gilt folgendes:

$$\begin{aligned} N &= \text{Anzahl der Events im gesamten Datenstrom} \\ C &= \text{Häufigkeit der Sequenz} \\ L &= \text{Länge der Sequenz} \\ \text{Wert} &= \frac{N}{N - (C * L) + C + L} \end{aligned}$$

Der Wert eines Musters ist größer, je länger die Sequenz ist und je häufiger sie vorkommt.

3.3.4 Cluster

Ein Cluster [Ester and Sander, 2000] besteht aus einem oder mehreren Mustern. Für jedes enthaltene Muster, in einem Cluster, werden die Distanzen zu allen anderen Mustern in dem Cluster, berechnet. Das Muster, welches die geringsten Distanzen zu allen anderen Mustern aufweist, wird als Clusterzentrum bezeichnet.
Um ein Muster zu einem Cluster hinzuzufügen, wird die Distanz zu dem Clusterzentrum berechnet. Ist die Distanz geringer als ein bestimmter Wert, wird das Muster zu dem Cluster hinzugefügt. In diesem Fall wird das Clusterzentrum neu berechnet.

In einem Cluster sind alle Muster enthalten, die ähnlich zueinander sind.
Am Ende der Aktivitätenentdeckung werden die Cluster mit zusätzlichen Informationen angereichert und stellen dann die Aktivitäten dar.

Ein einfaches Beispiel für ein Cluster sieht folgendermaßen aus: Gegeben sind drei Muster A, B und C mit den Sequenzen:

$$A = [a, b, c, d]$$
$$B = [a, c, c, d]$$
$$C = [a, b, d]$$

Die Distanzen der Muster zueinander sind:

	A	**B**	**C**
A	-	1	1
B	1	-	2
C	1	2	-

Tabelle 3.1: Distanzen der Muster untereinander

Wie die Distanzen berechnet werden, wird im nächsten Kapitel beschrieben.
Wenn alle Distanzen für ein Muster addiert werden, erhält man als Ergebnis für Muster A einen Wert von 2, für Muster B und C hingegen 3. Damit hat Muster A die geringste Distanz zu allen anderen Mustern in diesem Cluster und bildet somit das Clusterzentrum.

3.4 Distanzen

Eine der wichtigsten Eigenschaften für die Bildung von Clustern ist, wie sehr sich die Muster unterscheiden bzw. ähneln. Um herauszufinden, wie ähnlich sich zwei Muster sind, wird die Distanz [Ester and Sander, 2000, S. 47 f.] zwischen ihnen berechnet. Je

geringer die Distanz zwischen zwei Mustern ist, desto ähnlicher sind sie sich. Je größer die Distanz, desto unterschiedlicher sind sie.
Unterschreitet die Distanz zweier Muster einen bestimmten Wert, gehören sie zu dem selben Cluster.

Die Berechnung der Distanzen zählt zu den schwierigsten Aufgaben bei der Clusterbildung. Bei numerischen Sensordaten ist die Distanz noch recht einfach zu berechnen. Hier kann einfach die Differenz berechnet werden, oder die euklidische Distanz bei mehrdimensionalen Werten.
Bei beschreibenden Werten oder Zuständen ist es aber bereits nicht mehr so einfach, eine Distanz zu berechnen. Wie groß ist die Distanz zwischen dem angeschalteten Lichtschalter A und dem ausgeschalteten Lichtschalter B? Hier müssen für die Distanzen Werte festgelegt werden, die für die Clusterbildung sinnvoll sind.
Eine weitere Frage ist, wie die Distanzen zwischen Werten unterschiedlicher Sensortypen berechnet werden sollen. Wie groß ist die Distanz zwischen einem geöffneten Kühlschrank und einem eingeschalteten Lichtschalter?

Im Rahmen dieser Masterarbeit wird die gewichtete Levenshtein-Distanz [Levenshtein, 1966] verwendet. Dabei wird errechnet, durch wie viele Änderungsoperationen (Hinzufügen, Entfernen, Vertauschen...) ein Muster in ein anderes Muster gewandelt werden kann.
Wenn wir zwei Sequenzen $S1 = [a, b, c, d]$ und $S2 = [a, b, c]$ haben, kann die Sequenz $S2$, durch ein Hinzufügen des Events d ans Ende, in die Sequenz $S1$ umgewandelt werden. Umgekehrt kann die Sequenz $S1$ durch das Löschen des Events d in die Sequenz $S2$ umgewandelt werden. Dadurch beträgt die Distanz zwischen der Sequenz $S1$ und der Sequenz $S2$ 1.
Die Distanz zwischen zwei Sequenzen ist immer symmetrisch. Es ist also egal, ob die Distanz von der Sequenz $S1$ zu der Sequenz $S2$ berechnet wird oder umgekehrt die Distanz von $S2$ zu $S1$.

Die verschiedenen Operationen können unterschiedlich gewichtet werden. So kann z.B. das Hinzufügen von Events stärker gewichtet werden als das Vertauschen zweier Events. Zudem können für die einzelnen Sensortypen unterschiedliche Gewichtungen für die jeweiligen Operationen bestimmt werden.

Die genauen Werte für die Berechnungsgrundlage der Distanzen müssen durch sorgfältige und umfangreiche Versuche herausgefunden werden.

3.5 Programmablauf

Um von den Sensordaten zu den entdeckten Aktivitäten zu gelangen, werden die Sensordaten in mehreren Schritten verarbeitet.

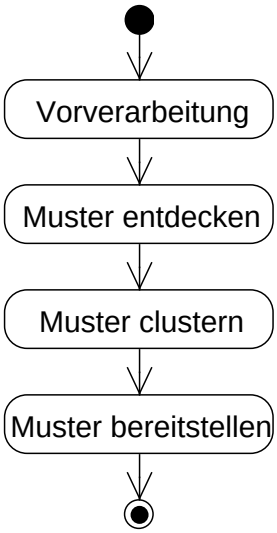


Abbildung 3.4: Programmablauf

In Abbildung 3.4 sind die einzelnen Schritte aufgeführt, um aus Sensordaten Aktivitäten zu entdecken. Die Schritte lehnen sich an das Kapitel 2.1 an.
Für diese Masterarbeit werden die Schritte *Vorverarbeitung*, *Muster entdecken*, *Muster clustern* und schließlich *Muster bereitstellen* verwendet.

3.5.1 Vorverarbeitung

Damit die Sensordaten in den späteren Schritten verwendet werden können, müssen sie vorverarbeitet werden. Dadurch können fehlerhafte Daten aussortiert und die Daten auf Plausibilität überprüft werden.
Bei den Positionsdaten ist z.B. darauf zu achten, dass sich die Position nicht sprunghaft

über große Distanzen verändert, sondern nur in dem Rahmen, wie es für einen Menschen möglich ist.

Die Sensordaten liegen in einer dateibasierten Ablage auf der Festplatte des Computers vor, der die Aktivitätenentdeckung durchführt. In Kapitel 3.2 wurde bereits erklärt, wie die Daten über das ActiveMQ empfangen werden und in welchem Datenformat sie in einer Textdatei gespeichert werden.
Die Sensordaten liegen bereits in zeitlich aufsteigender Reihenfolge vor. Das erspart ein nachträgliches Sortieren der Daten. Die einzelnen Sensordaten werden bei der Vorverarbeitung in Events umgewandelt, die wiederum in einer Eventliste gespeichert werden.

Beim Speichern in die Eventliste können die eingelesenen Sensordaten bereits auf einfache Fehler hin überprüft werden. zum Beispiel, ob die Werte innerhalb der Toleranzgrenzen liegen, vollständig sind.
Eine weitere wichtige Maßnahme in diesem Schritt ist die Reduzierung der Dimensionen der Sensorwerte. Überflüssige Informationen, die keine Rolle für die Entdeckung von Aktivitäten spielen, können ignoriert werden. Sofern es möglich ist, können auch mehrere Werte zu einem einzelnen Wert zusammengefasst werden. Ein Beispiel ist hier die Position im Raum. Der Sensor liefert drei Koordinaten, X, Y und Z. Die Z-Koordinate ist für die Aktivitätserkennung überflüssig, da für diese Masterarbeit von einem zweidimensionalen Bodenplan für das Livingplace ausgegangen wird. Die X- und Y-Koordinaten können mit Hilfe der Functional Spaces [Karstaedt, 2011] zu einem eindimensionalen Wert umgewandelt werden. Für die Aktivitätenentdeckung ist es nicht notwendig, die Position des Bewohners zentimetergenau bestimmen zu können. Es reicht vollkommen aus zu wissen, wo im Livingplace sich der Bewohner im welchem Zusammenhang aufhält. Dazu werden Funktional Spaces definiert, die angeben, ob sich der Bewohner z.B. im Bett, in der Nähe der Kochinsel oder am Esstisch befindet. Aus den Koordinaten $[3.5; 6.0; 1.0]$ wird dann ein einzelner Wert *KITCHEN_FRIDGE*. Details dazu sind in Kapitel 4.3 beschrieben.

Sind die Sensordaten komplett eingelesen, können nun aufwändigere Überprüfungen durchgeführt werden. Bei den Positionsdaten wird jetzt überprüft, ob sich der Bewohner mit einer plausiblen Geschwindigkeit durch die Wohnung bewegt. Sollte die Beschleunigung von einer Position zur nächsten zu hoch sein, wird das Event wieder aus der Eventliste gelöscht.

3.5.2 Muster entdecken

In den aufbereiteten Events können nun Muster entdeckt werden. Ich verwende dazu das Verfahren, das im CASAS-Projekt [Rashidi et al., 2011] entwickelt wurde und passe es für die spezifischen Gegebenheiten des Livingplace an.
Dazu wird die Eventliste L_{event} Event für Event durchgegangen und für jedes Event festgehalten, wie oft es sich in dieser Liste wiederholt. Events, die sich oft genug wiederholen, werden in einer weiteren Liste L_{repeat} gespeichert und um die vorstehenden und nachfolgenden Events zu Sequenzen erweitert. Für jede dieser erweiterten Sequenzen wird wieder berechnet, wie oft sie sich in der Eventliste L_{event} wiederholt und in die Liste L_{repeat} übernommen, wenn sie häufig genug vorkommen. Die Sequenzen werden solange erweitert, wie es möglich ist, bzw. keine Sequenzen mehr gefunden werden, die häufig genug vorkommen.

Die so entdeckten sich wiederholenden Sequenzen werden in Muster umgewandelt. Jedes dieser Muster wird nun evaluiert. Dazu werden sein Wert, die Durchschnittszeit und die Standardabweichung des Zeitstempels errechnet, wie in Kapitel 3.3.3 beschrieben. Mit Hilfe der errechneten Werte wird nun das Muster ausgewählt, welches den höchsten Wert aufweist. Dieses Muster wird in eine separate Liste L_{best} für gefundene Muster gespeichert. Die Sequenz des besten Musters wird in der Eventliste L_{event} an allen auftauchenden Stellen durch sein erstes Event aus der Sequenz ersetzt. Dadurch entsteht eine komprimierte Eventliste L_{compr}.

Nun wird wieder von vorne angefangen und in der komprimierten Eventliste L_{compr} wird für die Events berechnet, wie häufig sie vorkommen usw. Dies wird solange wiederholt, bis eine maximale Anzahl an Iterationen erreicht wurde, oder bis sich die Eventliste nicht mehr komprimieren lässt.

Am Ende des Schritts *Muster entdecken* stehen zwei Listen, L_{repeat} und L_{best}, zur Verfügung. Die Liste L_{repeat} enthält sämtliche entdeckten Muster aus allen Iterationen, und die Liste L_{best} enthält die Muster mit dem jeweils höchsten Wert aus jeder Iteration.

3.5.3 Muster clustern

Aus den entdeckten Mustern werden nun Cluster gebildet. Hierzu werden die Muster miteinander verglichen und ähnliche Muster zu einem Cluster zusammengefasst.
Dazu werden die Techniken angewandt, die in den Kapiteln 3.3.4 und 3.4 vorgestellt wurden.

Da zu Anfang noch keine Cluster existieren, wird das erste Muster aus der Liste der entdeckten Muster genommen und in ein neues, leeres Cluster eingefügt. Dadurch ist dieses Muster automatisch das Zentrum des neuen Clusters. Nun wird das nächste Muster aus der Liste genommen und die Distanz zu dem Clusterzentrum berechnet. Liegt die Distanz unter einem Schwellwert, wird das Muster dem Cluster hinzugefügt. In diesem Fall wird das Zentrum des Cluster neu berechnet. Ansonsten wird ein neues Cluster mit diesem Muster erzeugt.
So wird mit der gesamten Liste der entdeckten Muster verfahren. Die Muster werden entweder einem bestehendem Cluster oder einem neuen Cluster hinzugefügt.

Am Ende dieses Schritts ist eine Liste mit Clustern $L_{cluster}$ entstanden, in der jedes Muster einem der enthaltenen Cluster zugeordnet wurde.

3.5.4 Muster bereitstellen

Im letzten Schritt werden die entdeckten Cluster anderen Projekten des Livingplace zur Verfügung gestellt. Damit diese Projekte die Cluster sinnvoll verwenden können, werden sie mit zusätzlichen Informationen angereichert.

Bei diesen zusätzlichen Informationen handelt es sich unter anderem um die in der Aktivität enthaltenen Functional Spaces und Sensortypen.
Diese Informationen können dazu verwendet werden, die Aktivitäten zu filtern und zu sortieren.

Ist auch dieser Schritt abgeschlossen, ist die Aktivitätenentdeckung beendet.

3.6 Fazit

In diesem Kapitel habe ich ein Design vorgestellt, mit dem ich die Ziele aus Kapitel 1.2 erreichen kann und die Anforderungen aus Kapitel 2.4 erfüllt werden.
Das Design greift auf die verfügbaren Ressourcen des Livingplace zu. Es verwendet ActiveMQ als Middleware für die Übertragung von Sensorwerten und für die Verbreitung der entdeckten Aktivitäten. Die Aktivitätenentdeckung ist unabhängig von der tatsächlichen Quelle der Sensordaten. Neben der Verwendung von realen Daten ist es möglich, auf simulierte Daten des Scriptsimulators zurückzugreifen.
Mit *Events*, *Sequenzen*, *Mustern* und *Clustern* wurden Datenstrukturen vorgestellt, mit denen es möglich ist, die Sensordaten systematisch zu verarbeiten und in ihnen Aktivitäten zu entdecken.
In Kapitel 3.5 wurden die einzelnen Schritte der Aktivitätenentdeckung vorgestellt und erläutert, welche Funktion sie erfüllen.

Das Design ist darauf ausgelegt, jederzeit ohne großen Aufwand erweitert werden zu können. Sollten in der Zukunft neue Sensortypen im Livingplace installiert werden, können diese ohne großen Aufwand in der Aktivitätenentdeckung berücksichtigt werden. Weiterhin kann das Design so erweitert werden, dass die Aufgabenstellung nicht nur das Livingplace umfasst, sondern auch dessen Umgebung. Es ist auch möglich, komplett andere Umgebungen zu erfassen, z.B. andere Wohnungen.

4 Implementierung

Im vorangegangenen Kapitel 3 habe ich die grundsätzliche Struktur der Aktivitätenentdeckung erläutert. In diesem Kapitel gehe ich auf die Details ein und werde bei den einzelnen Komponenten zeigen, wie ich sie in der Praxis für das Livingplace umgesetzt habe.

4.1 Daten erfassen

Für die Entdeckung der Aktivitäten werden die Sensordaten aus Tabelle 4.1 verwendet.

Typ	Beschreibung	Beispiel
Alarm	Wecker wird ausgelöst	alarm; wecker
Bed	Schlafphasen des Bewohners	REM
Couch	Wo auf der Couch sitzt der Bewohner	c10
Door	Welche Tür wurde geöffnet/geschlossen	Haustuer; oeffnen
Position	Position des Bewohners	x:10.5; y:9.4
Power	elektrischer Verbraucher wird an-/ausgeschaltet	HERD; ON
Storage	Schrank wird geöffnet/geschlossen, bzw. Objekt wird entnommen/hinzugefügt	BATH_SHELVE; OPENED
Water	Wasserhahn wird geöffnet/geschlossen	BATH_SINK_HOT; ON

Tabelle 4.1: Verwendete Sensortypen und Beschreibung

Die Sensordaten werden vom Scriptsimulator erzeugt und über das ActiveMQ versendet. Gleichzeitig werden die Sensordaten vom Scriptsimulator in eine Datei geschrieben, um die Daten persistent für die Aktivitätenentdeckung vorzuhalten.
In Listing 4.1 ist ein Ausschnitt aus einer solchen Datei angegeben.

```
1 Alarm;1369782076660;Telefon;Arbeitskontakt 1
2 Ubisense;1369783276660;1.33;1.18
3 Power;1369783306660;KAFFEMASCHINE;ON
4 Storage;1369783316660;KITCHEN_SHELF;OPENED
5 Storage;1369783321660;KITCHEN_SHELF;OBJECT_TAKEN
6 Storage;1369783326660;KITCHEN_SHELF;CLOSED
7 Alarm;1369783576660;Kaffemaschine;Kaffe fertig
8 Ubisense;1369783606660;1.33;1.18
9 Ubisense;1369785076660;3.37;1.31
10 Water;1369785406660;BATH_TOILET_COLD;ON
```

Listing 4.1: Ausschnitt Rohdaten

Pro Zeile ist immer ein Sensorevent angegeben. Die einzelnen Informationen eines Events sind durch Semikolons getrennt.
Der erste Wert einer Zeile gibt an, um welchen Sensortypen es sich handelt. Der zweite Wert ist der Zeitstempel, wann der Sensorwert erzeugt wurde. Der Zeitstempel entspricht der UNIX-Zeit mit einer Genauigkeit von einer Millisekunde.
Nach dem Zeitstempel folgt der eigentliche Wert des Sensors. Dabei kann es sich um einen einzelnen Wert, aber auch um mehrere handeln. Wie diese Werte zu interpretieren sind, hängt vom jeweiligen Sensortyp ab.

Die eingelesenen Sensordaten werden als Events (s. Kapitel 4.4.1) in der globalen Liste `eventList` gespeichert.

4.2 Vorverarbeitung

Wie im Kapitel 3.5.1 bereits beschrieben, werden die Sensordaten vor der Aktivitätenentdeckung bearbeitet, um u.a. Fehler zu bereinigen.

Die ersten Korrekturen finden beim Erzeugen von Events aus den Sensordaten statt.
Der erste Schritt ist, die eingelesenen Daten auf ihre Korrektheit hin zu untersuchen. Hier wird untersucht, ob die Werte im gültigen Wertebereich liegen, vollständig sind und korrekt übertragen wurden.
Sensordaten, die diese erste Prüfung nicht bestehen, werden verworfen und für die nächsten Schritte nicht verwendet.

Ist die Korrektheit der Sensordaten überprüft worden, werden nun ihre Dimensionen reduziert, um eine einfachere Berechnung zu ermöglichen. Dazu werden mehrere Werte eines Sensors zu einem Wert aggregiert.
Bei den Positionsdaten können z.B. die Werte für die Koordinaten in X- und Y-Richtung zu einem Wert zusammengefasst werden, den Functional Spaces (s. Kapitel 4.3).

Sind alle Events erzeugt worden, können nun Fehler gesucht und beseitigt werden, die sich aus dem Zusammenhang mehrerer Sensordaten ergeben.
Am Beispiel der Positionsdaten kann überprüft werden, ob aufeinanderfolgende Positionen plausibel von einem Menschen erzeugt werden können. Wenn eine Position zu schnell und zu weit von einer vorherigen Position auftaucht, wird sie für die Aktivitätenentdeckung verworfen.

4.3 Functional Spaces

Die Functional Spaces dienen dazu, Positionsangaben mit einer Funktion zu verknüpfen. Für die Aktivitätenentdeckung ist es nicht notwendig, die Position des Bewohners zentimetergenau bestimmen zu können. Wichtiger ist zu wissen, in welchem Zusammenhang die Position steht. Bastian Karstaedt hat in seiner Masterarbeit [Karstaedt, 2011] bereits beschrieben, wie Functional Spaces für das Livingplace aussehen können.

Ich habe für meine Masterarbeit die Functional Spaces definiert, wie sie in Abbildung 4.1 zu sehen sind.

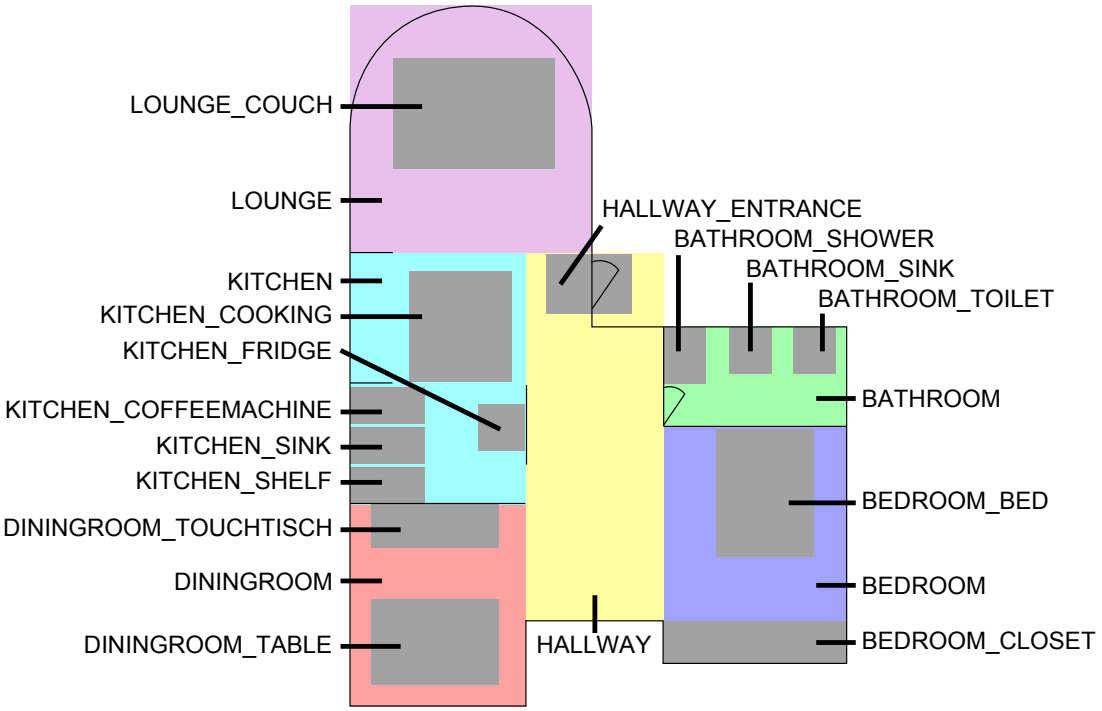


Abbildung 4.1: Functional Spaces im Livingplace

Die farblich markierten Flächen entsprechen den Räumen der Wohnung, z.B. Küche, Bad, etc. Die grauen Flächen innerhalb der Räume bestimmten Bereiche, in denen der Bewohner mit Gegenständen im Livingplace interagieren kann, z.B. Herd, Dusche etc.

4.4 Verwendete Klassen und Datenstrukturen

In diesem Abschnitt gebe ich einen kurzen Überblick über die wichtigsten Klassen und Datenstrukturen, die ich in meinem Programm verwendet habe. Es werden nur solche Klassen und Methoden aufgeführt, die für die Erläuterung der Aktivitätenentdeckung notwendig sind. Hilfsklassen und Getter- und Settermethoden o.ä. werden nicht erwähnt,

um die Übersichtlichkeit zu erhalten.
Dieses Kapitel setzt die in Kapitel 3.3 beschriebenen Datenstrukturen um.

4.4.1 IEvent und Event

Das Interface `IEvent` bildet die Basis für die Aktivitätenentdeckung. Von diesem Interface ist die Klasse `Event` abgeleitet, die grundlegende Funktionen implementiert, die von allen Eventtypen genutzt werden. Für jeden Sensortyp wird eine eigene Eventklasse erzeugt, die von der Klasse `Event` erbt. Für den Sensortyp *Alarm* wäre das z.B. die Klasse `AlarmEvent`.
Mit dem Interface `IEvent` wird die Methode `distanceTo(IEvent)` deklariert. Diese Methode dient dazu, die Distanz zwischen diesem Eventobjekt und dem als Parameter übergebenen Eventobjekt zu berechnen (s. Kapitel 4.5).

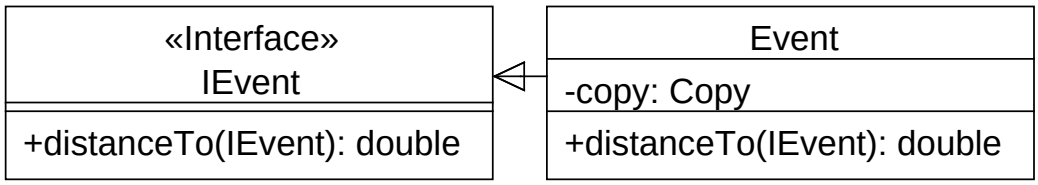


Abbildung 4.2: IEvent und Event UML-Klassendiagramm

Die abgeleiteten Eventklassen für die einzelnen Sensortypen müssen die Methode `distanceTo(IEvent)` implementieren. Zusätzlich muss ein passender Konstruktor implementiert werden, dem die notwendigen Sensordaten übergeben werden können. In diesem Kontruktor finden die in Kapitel 4.2 beschriebenen Überprüfungen der Sensordaten statt.
Die Eventklassen müssen alle das Feld `copy` enthalten, welches ein Enumerateobjekt vom Typ `Copy` ist. Für `Copy` sind die Zustände `FALSE`, `TRUE`, `NEW` und `PREDEFINED` definiert. Dieses Feld wird für die Komprimierung der globalen Eventliste verwendet, wie in Kapitel 4.6.3 beschrieben.

4.4.2 Sequence

Mit der Klasse `Sequence` werden Eventsequenzen beschrieben.
In dem Feld `sequence` sind die Events in chronologisch aufsteigender Reihenfolge gespeichert.

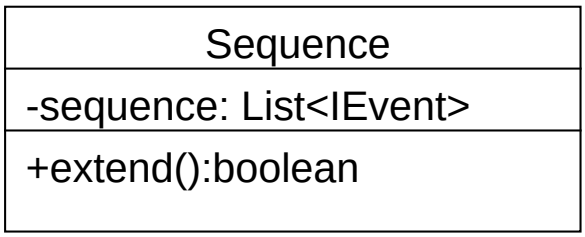


Abbildung 4.3: Sequenz UML-Klassendiagramm

Die Methode `extend()` dient dazu, die Sequenz zu erweitern. Dazu wird der Liste `sequence` das Event hinten angehängt, welches sich in der globalen Eventliste hinter der Sequenz befindet. Konnte die Sequenz erweitert werden, liefert die Methode den Wert `true` zurück. Scheitert das Erweitern, z.B. weil die Sequenz am Ende der globalen Eventliste steht, liefert die Methode den Wert `false` zurück.

4.4.3 Pattern

Mit der `Pattern`-Klasse werden die Muster für die Aktivitätenentdeckung verwaltet. Das Feld `sequence` gibt die Sequenz an, die dieses Muster repräsentiert. In der Liste `instances` sind die Indices gespeichert, an denen die Sequenz in der globalen Eventliste vorkommt. Mit dem Feld `count` wird festgehalten, wie häufig die Sequenz in der globalen Eventliste vorkommt.
Die Felder `value`, `meantime` und `stddevtime` geben den Wert, die Durchschnittszeit und die Standardabweichung der Zeitstempel wieder.

Mit der Methode `evaluate(int)` werden die Werte für die Felder `value`, `meantime` und `stddevtime` berechnet. Der Integerwert, der dieser Methode übergeben wird,

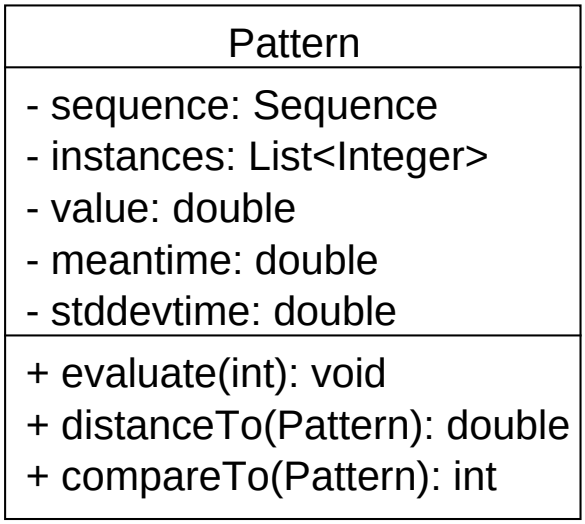


Abbildung 4.4: Pattern als UML Klassendiagramm

entspricht der Anzahl aller Events in der globalen Eventliste. Wie die Werte berechnet werden ist in Kapitel 3.3.3 beschrieben.
Die Methode `distanceTo(Pattern)` berechnet die Distanz zwischen einem Patternobjekt und dem übergebenen Patternobjekt (s. Kapitel 3.4).
Um Pattern sortieren zu können, wird die Methode `compareTo(Pattern): int` des Java `Comparable`-Interfaces implementiert. Dadurch können Patternobjekte anhand des Wertes in dem Feld `value` sortiert werden. Das wird für die Bestimmung des besten Musters benötigt (s. Kapitel 4.6.2).

4.4.4 Cluster

Die Klasse `Cluster` dient dazu, Muster bei der Aktivitätenentdeckung zu Clustern zusammenzufügen.
Das Feld `centroid` repräsentiert das Clusterzentrum. Im Feld `patternList` sind alle Muster enthalten, die zu diesem Cluster gehören.

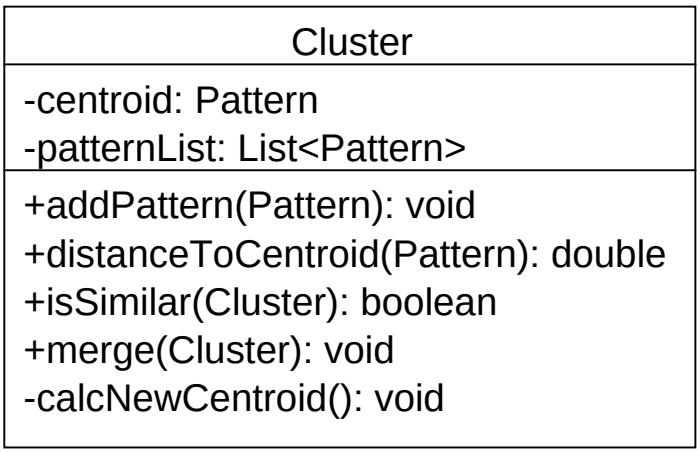


Abbildung 4.5: Cluster UML-Klassendiagramm

Um herauszufinden, ob ein Pattern zu einem Cluster gehört, wird das Pattern der Methode `distanceToCentroid(Pattern)` übergeben. Diese Methode berechnet die Distanz zwischen dem übergebenen Pattern und dem Clusterzentrum.
Ob sich zwei Cluster ähneln, wird mit der Methode `isSimilar(Cluster)` berechnet. Dazu wird die Distanz zwischen den beiden Clusterzentren berechnet. Unterschreitet der Wert einen bestimmten Wert, liefert die Methode `true` zurück, ansonsten liefert sie `false`. Soll ein Pattern dem Cluster hinzugefügt werden, wird die Methode `addPattern(Pattern)` verwendet. Das übergebene Pattern wird der Liste `patternList` hinzugefügt. Es wird automatisch ein neues Clusterzentrum mit Hilfe der Methode `calcNewCentroid()` berechnet.
Mit der Methode `merge(Cluster)` werden die Pattern aus dem übergebenen Cluster der `patternList` dieses Clusters hinzugefügt und anschließend automatisch ein neues Clusterzentrum berechnet.

4.5 Distanzen

Die Distanz zwischen Events und Mustern ist das wichtigste Maß, um Muster zu clustern. Für die Berechnung der Distanzen wird gewichtete Levenshtein-Distanz [Levenshtein, 1966] verwendet, die in Kapitel 3.4 genauer erklärt ist.

Die Distanzen können einen Wert zwischen 0 und 1 annehmen. 0 bedeutet, dass zwei Events miteinander übereinstimmen, 1 bedeutet, dass sie komplett verschieden sind. Werte zwischen 0 und 1 können nur zwischen Events entstehen, die vom gleichen Eventtyp sind. Unterschiedliche Eventtypen haben untereinander immer eine Distanz von 1.

Wie die Distanzen zwischen zwei Events desselben Eventtyps bestimmt werden, hängt von dem jeweiligen Typ ab.
Bei Positionsevents wird zunächst überprüft, ob beide Events den gleichen Functional Space aufweisen. Ist das der Fall, beträgt die Distanz 0. Sind die Functional Spaces unterschiedlich, wird die euklidische Distanz zwischen den beiden Punkten berechnet. Für Eventtypen, die durch diskrete Werte, wie z.B. Zustände, beschrieben werden, werden vordefinierte Tabellen erstellt. Diese Tabellen enthalten für alle möglichen Kombinationen der diskreten Werte einen fest definierten Distanzwert.

Für die Distanzen zwischen zwei Mustern wird zunächst die Differenz der beiden enthaltenen Sequenzlängen berechnet. Dadurch wird festgestellt, wie viele Events hinzugefügt bzw. entfernt werden müssen, um die Länge des jeweils anderen Musters zu erreichen.
Anschließend werden die Sequenzen der Muster direkt miteinander verglichen und die Distanzen der einzelnen Events aufaddiert.
Die beiden Werte ergeben addiert die Distanz zwischen den beiden Mustern.

In Abbildung 4.6 ist ein Beispiel abgebildet.
Gegeben sind zwei Sequenzen, *Sequenz 1* mit einer Länge von 2 und *Sequenz 2* mit einer Länge von 3. Die Differenz der Sequenzlängen beträgt 1. Die Differenz der jeweils ersten Events beträgt 0.5 und der zweiten 0. Dass die erste Sequenz keinen Gegenpart

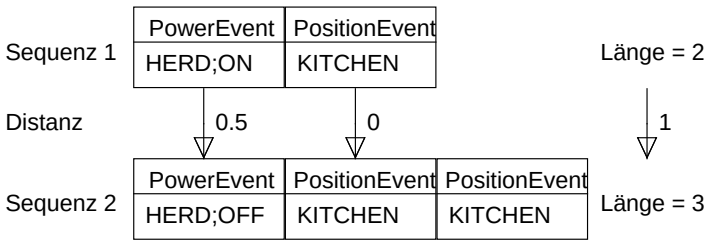


Abbildung 4.6: Beispiel zur Distanzberechnung

für das dritte Event in der zweiten Sequenz hat, ist bereits in der Differenz der Sequenzlängen berücksichtigt. Die aufaddierten Distanzen der Events ergeben somit 0.5.
Addiert man nun die Summe der Eventdifferenzen (0.5) mit der Differenz der Sequenzlänge (1), erhält man die Distanz zwischen den beiden Sequenzen von 1.5.
Da jedes Muster immer nur eine Sequenz enthält, entspricht die Distanz der Sequenzen auch immer der Distanz der Muster.

4.6 Aktivitätenentdeckung

Der Kern dieser Masterarbeit ist das Entdecken von Aktivitätsmustern aus Sensordaten. In den vorhergehenden Kapiteln habe ich erklärt, wie die Daten erzeugt, erfasst und vorbereitet werden. In diesem Kapitel erkläre ich nun, wie aus den erzeugten Events Aktivitäten entdeckt werden, und wie sie zu Clustern zusammengefasst werden.

In Abbildung 4.7 ist der Ablaufplan der Aktivitätenentdeckung zu sehen. Es werden mehrere Schritte nacheinander ausgeführt: *1. Muster entdecken*, *2.bestes Muster speichern 3. Eventliste komprimieren*, *4. Muster clustern*, *5. Cluster zusammenfassen* und *6. Aktivitäten bereitstellen*.

Meine Implementierung basiert auf dem CASAS-Projekt (Kapitel 2.3.3) und ist auf die Gegebenheiten des Livingplace erweitert und zugeschnitten.

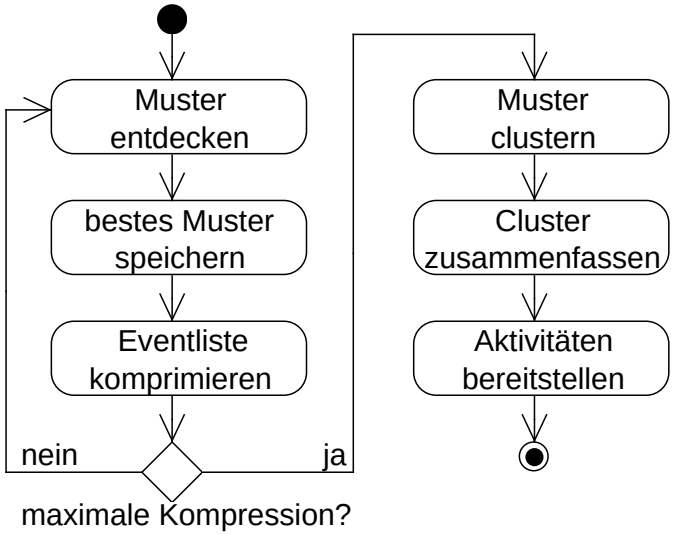


Abbildung 4.7: Ablaufplan der Aktivitätenentdeckung

4.6.1 Schritt 1: Muster entdecken

Im ersten Schritt werden Muster in der globalen Eventliste `eventList` entdeckt. Dazu werden zunächst initiale Muster entdeckt, die leicht erkannt werden können. Auf dieser Basis werden dann alle weiteren Muster entdeckt, die in der `eventList` stecken. Anschließend werden die entdeckten Muster evaluiert.

Das Verfahren zum Entdecken aller Muster basiert auf dem Apriori-Algorithmus [Beierle and Kern-Isberner, 2006, S. 147 f.].

Initiale Muster entdecken

Zunächst werden in der `eventList` initiale Muster gesucht, auf deren Basis weitere Muster entdeckt werden. Die initialen Muster haben alle eine Sequenzlänge von 1.
Die `eventList` wird Event für Event durchgegangen und für jedes Event wird berech-

net, wie oft es sich in der `eventList` wiederholt. Jedes Event, das sich mindestens zweimal wiederholt, stellt ein initiales Muster dar.

Alle initialen Muster werden in der Liste `patternList` gespeichert, in der sämtliche entdeckten Muster gespeichert werden.

Erweitere Muster entdecken

Die initialen Muster werden nun erweitert. Dazu wird die Sequenz des jeweiligen Musters um das Event erweitert, welches in der `eventList` direkt dahinter steht.

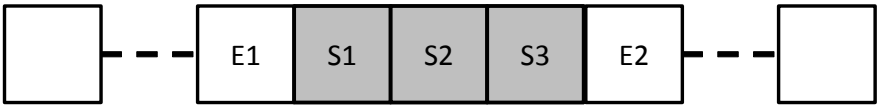


Abbildung 4.8: Erweitern einer Sequenz

In Abbildung 4.8 ist dieser Vorgang dargestellt. Die Sequenz besteht aus den Events $S1$, $S2$ und $S3$. Um die Sequenz zu erweitern, wird ihr nun das Event $E2$ angehängt.
Die erweiterten Muster werden in der Liste `extendedList` gespeichert.

Die Liste `extendedList` wird nun Muster für Muster durchgegangen, und für jedes Muster wird herausgefunden, wie oft es sich in der `eventList` wiederholt. Dazu wird ein Fenster mit der Größe der Sequenzlänge des jeweiligen Musters über die `eventList` geschoben. Bei jeder genauen Übereinstimmung mit der Sequenz aus diesem Fenster und der Sequenz des Musters wird der Zähler für die Musterwiederholungen um 1 erhöht.

Nun werden alle erweiterten Muster, die sich mindestens zweimal in der `eventList` wiederholen, ihrerseits erweitert. Das wiederholt sich solange, bis entweder keine Muster mehr erweitert werden können oder sich die erweiterten Muster nicht mehr in der `eventList` wiederholen.

Alle erweiterten Muster, die sich mindestens zweimal wiederholen, werden in der Liste `patternList` gespeichert. Am Ende dieses Unterschrittes enthält die Liste `patternList` alle möglichen Muster der `eventList`, die sich mindestens zweimal wiederholen.

Muster evaluieren

Zum Schluss des ersten Schrittes *Muster entdecken* werden alle entdeckten Muster in der Liste `patternList` evaluiert. Dazu werden der Wert, die Durchschnittszeit und die Standardabweichung der Zeit jedes einzelnen Musters berechnet.

Für die Berechnung werden folgende Ausgangsdaten benötigt:
E: Länge der eventList
N: Häufigkeit des Musters
T: Menge aller Zeitstempel der Rohdaten

Der Wert V eines Musters berechnet sich folgendermaßen:

$$V = \begin{cases} 0 & \text{wenn } L = 0 \\ \frac{N}{E} & \text{wenn } L = 1 \\ \frac{E}{E-(L*N)+L+N} & \text{sonst} \end{cases} \tag{4.1}$$

Ist die Länge eines Musters 0 oder 1, dann ist der Wert 0 bzw. der Quotient aus N und E. Ist die Länge größer als 1, dann wird die Anzahl der Rohdaten E dividiert durch E, abzüglich dem Produkt aus Länge des Musters L und dessen Häufigkeit N, zuzüglich den beiden Werten L und N.

Um die Standartabweichung des Zeitstempels zu berechnen, wird zunächst die Summe der Zeitstempel T_{sum} benötigt,

$$T_{\text{sum}} = \sum_{i=1}^{E} T_i \tag{4.2}$$

um daraus den Durchschnittswert T_{mean} zu berechnen.

$$T_{\text{mean}} = \frac{T_{\text{sum}}}{N} \tag{4.3}$$

Die Standardabweichung der Zeit des Musters t_{stddev} errechnet sich dann so:

$$t_{\text{stddev}} = \sqrt{\frac{\sum_{i=1}^{N}(T_i - T_{\text{mean}})^2}{L}} \tag{4.4}$$

Der Wert eines Musters wird im Schritt 2 verwendet, um das beste Muster zu bestimmen. Die Durchschnittszeit und deren Standardabweichung werden im letzten Schritt bei der Bereitstellung der Aktivitäten verwendet.

4.6.2 Schritt 2: Bestes Muster speichern

Nachdem im vorhergehenden Schritt sämtliche sich wiederholende Muster in der globalen Eventliste `eventList` gefunden wurden, wird nun das Muster mit dem höchsten Wert in eine separate Liste `bestPatternList` gespeichert.
Dieses beste Muster wird im nächsten Schritt für die Komprimierung der `eventList` verwendet. Die Muster der `bestPatternList` werden im Schritt 4 als Grundlage für die Clusterbildung verwendet.

4.6.3 Schritt 3: Eventliste komprimieren

In diesem Schritt werden sämtliche Vorkommen des besten Musters in der `eventList` markiert. Anschließend wird die `eventList` komprimiert.

Jedes Event enthält das Feld `copy`, welches angibt, was mit dem Event bei der Komprimierung geschehen soll. Das Feld `copy` kann bei der Komprimierung drei Werte annehmen, `TRUE`, `FALSE` und `NEW`.

Alle Events der `eventList`, die nicht Teil des besten Musters sind, erhalten den Wert `TRUE`.
Das erste Event der Sequenz eines Musters wird in der Liste `eventList` mit dem Wert `NEW` markiert. Alle weiteren Events der Sequenz werden mit `FALSE` markiert. Das passiert an allen Stellen in der `eventList`, an der die Mustersequenz vorkommt.

Sind alle Vorkommen des besten Musters in der Liste `eventList` markiert, wird die `eventList` komprimiert. Dazu wird die `eventList` Event für Event durchgegangen und der Wert von `copy` untersucht. Ist der Wert dieses Feldes `TRUE` oder `NEW`, dann wird dieses Event in eine neue Liste `eventListCompressed` kopiert. Alle Events, die mit `FALSE` markiert wurden, werden nicht in die neue Liste kopiert.

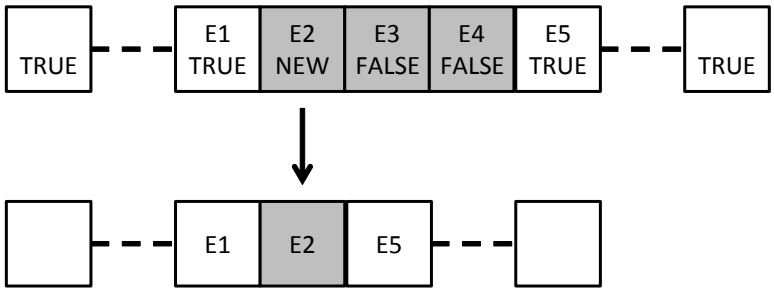


Abbildung 4.9: Eventliste komprimieren

Nun wird die Länge der Liste `eventList` mit der Liste `eventListCompressed` verglichen. Ist die Länge von `eventListCompressed` kürzer als die von `eventList`, war die Komprimierung erfolgreich. Sind beide Längen gleich groß, ist die maximale Kompression der Liste `eventList` erreicht.

Konnte die `eventList` komprimiert werden, wird die `eventList` durch die Liste `eventListCompressed` ersetzt und die Musterentdeckung fängt wieder bei Schritt 1 an.

Wenn die `eventList` nicht weiter komprimiert werden kann, wurden alle Muster entdeckt und die Musterentdeckung ist abgeschlossen.
Es stehen nun zwei Listen mit Mustern zur Verfügung. Eine Liste mit sämtlichen entdeckten Mustern aus allen Iterationen `patternList` und eine Liste mit allen besten Mustern `bestPattern` aus jeder Iteration.
Mit diesen beiden Listen wird nun mit Schritt 4 fortgefahren.

4.6.4 Schritt 4: Muster clustern

Wurden alle Muster in der `eventList` gefunden, werden die entdeckten Muster zu Clustern zusammengefasst. Für die Clusterbildung wird der k-means Algorithmus [Berry and Kogan, 2010, S. 81] verwendet.

Als erstes werden aus der Liste `bestPattern` die Cluster erzeugt. Dazu wird aus jedem der darin enthaltenen Muster ein neues Cluster generiert, dass nur das jeweilige Muster

als Clusterzentrum enthält, ansonsten aber leer ist. Die Cluster werden in der Liste `clusterList` gespeichert.

Sind die Cluster erzeugt, werden sie mit den Mustern aus `patternList` gefüllt. Die Muster werden nacheinander den Clustern hinzugefügt.
Dazu werden alle Distanzen von dem Muster zu allen Clustern berechnet und das Cluster mit der geringsten Distanz ausgewählt. Die geringste Distanz wird nun daraufhin untersucht, ob sie unterhalb der maximalen Distanz von 2 liegt. Ist das der Fall, wird das Muster zu diesem Cluster hinzugefügt und das Clusterzentrum dieses Cluster neu berechnet.
Liegt die geringste Distanz über dem Wert 2, wird das Muster keinem Cluster zugeordnet, sondern verworfen.

Sind sämtliche Muster einem Cluster zugeordnet, werden die entstandenen Cluster im nächsten Schritt komprimiert.

4.6.5 Schritt 5: Cluster zusammenfassen

Nachdem die Clusterbildung abgeschlossen ist, wird versucht, die Cluster zusammenzufassen.

Dazu wird die Liste `clusterList` Cluster für Cluster durchgegangen und für jedes Cluster werden die Distanzen zu allen anderen Clustern berechnet. Die Distanz zweier Cluster entspricht der Distanz der Clusterzentren.
Von den berechneten Distanzen wird die geringste Distanz ausgewählt. Ist deren Wert kleiner als 2, werden die Cluster zu einem Cluster zusammengefasst. Dazu werden die Muster des einen Clusters dem anderen Cluster hinzugefügt.

Sind die Cluster zusammengefasst worden, ist die Clusterbildung abgeschlossen und die Aktivitätenentdeckung kann im nächsten Schritt mit der Erzeugung und Bereitstellung abgeschlossen werden.

4.6.6 Schritt 6: Cluster bereitstellen

Im letzten Schritt der Aktivitätenentdeckung werden die entdeckten Cluster in Aktivitäten umgewandelt. Dabei werden die Aktivitäten mit zusätzlichen Informationen angereichert, die die Interpretation der Aktivitäten erleichtern.

```
{
    "containedTypes": [
        "AlarmEvent",
        "PositionEvent"
    ],
    "meantime": 1.38182507E12,
    "stddevtime": 2730000.0,
    "functionalSpaces": [
        "DININGROOM_TABLE"
    ],
    "patternList": [
        [
            {
                "time": 1381822340000,
                "name": "Telefon",
                "description": "Kontakt1"
            },
            {
                "time": 1381827800000,
                "x": 2.0054425647346696,
                "y": 1.51,
                "fspace": "DININGROOM_TABLE"
            }
        ]
    ]
}
```

Listing 4.2: Aktivität

Die zusätzlichen Informationen werden aus den Clustern und den darin enthaltenen Mustern einer Aktivität erstellt. Bei diesen Informationen handelt es sich um die enthaltenen Eventtypen (`containedTypes`), die Durchschnittszeit (`meantime`) und Standardabweichung (`stddevtime`) des Clusterzentrums, die enthaltenen Functional Spaces (`functionalSpaces`) sowie eine Liste der enthaltenen Muster (`patternList`).

Die Aktivitäten werden im JSON Format über das ActiveMQ auf dem Topic *activitiesTopic* bekanntgegeben.

In Listing 4.2 ist ein Beispiel angegeben, wie eine Nachricht mit einer Aktivität aussieht.

Graphische Darstellung

Die entdeckten Aktivitäten können auch grafisch dargestellt werden. Dazu habe ich eine einfache Oberfläche implementiert, die den Grundriss des Livingplace zeigt. Auf diesem Grundriss sind die Functional Spaces farblich hervorgehoben.

Auf dieser Oberfläche können nun die Positionsdaten der Aktivitäten angezeigt werden. Jeder Punkt entspricht einem Positionswert. Jede Aktivität ist durch eine andere Farbe dargestellt.
Diese Darstellung eignet sich dazu herauszufinden, ob die Muster sinnvoll zu Aktivitäten geclustert worden sind.

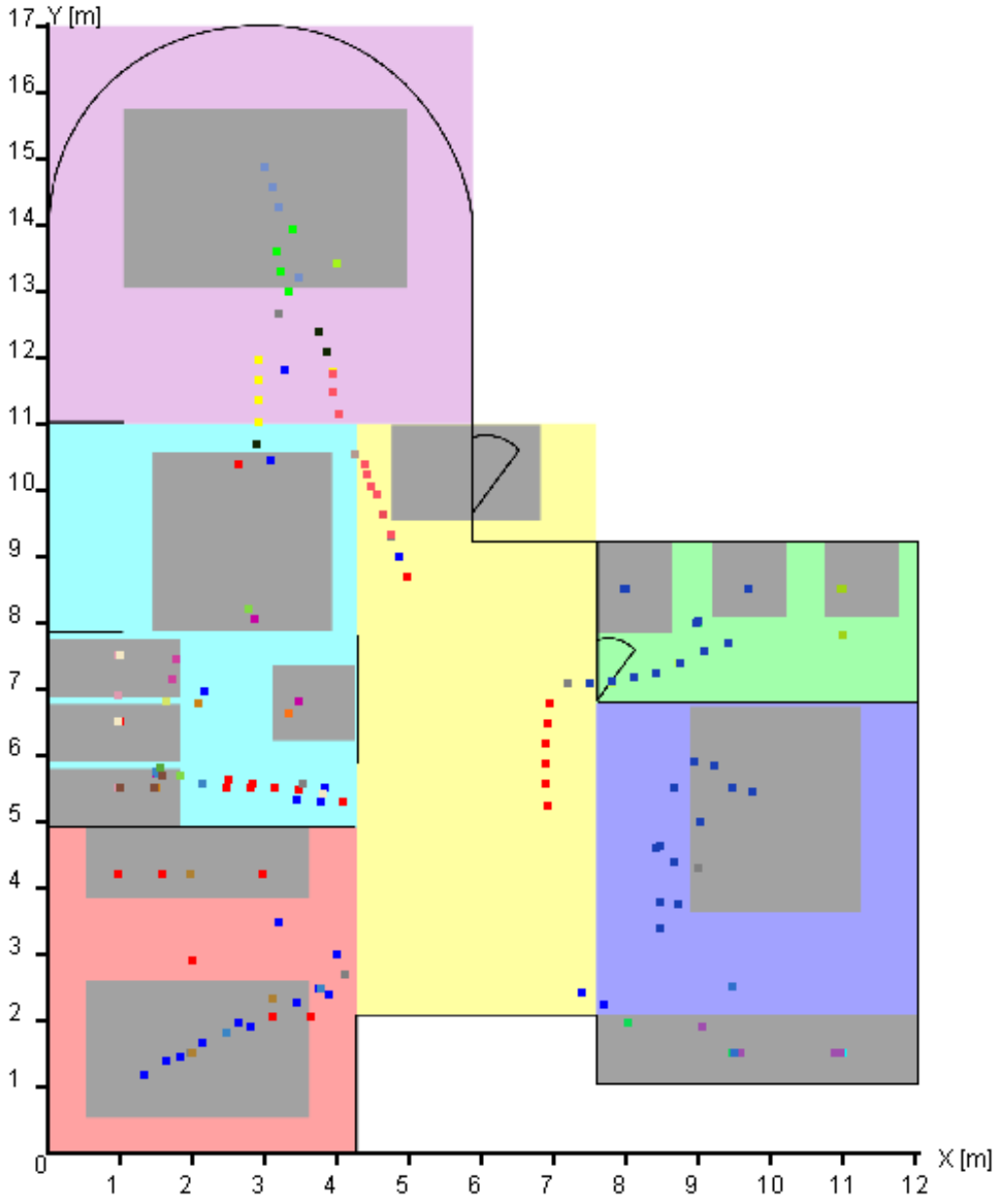


Abbildung 4.10: Ergebnis der Aktivitätenentdeckung

4.7 Fazit

In diesem Kapitel habe ich eine Implementierung des Designs aus Kapitel 3 vorgestellt. Mit dieser Implementierung können aus einer Datei Sensordaten ausgelesen werden, die zunächst auf Fehler untersucht werden. Die überprüften Sensordaten werden als Events in eine Liste gespeichert, die die Basis für die Aktivitätenentdeckung bildet. In der anschließenden Aktivitätenentdeckung werden alle enthaltenen Muster in der Eventliste entdeckt. Aus diesen Mustern werden dann Cluster erzeugt, welche anschließend zu Aktivitäten erweitert werden. Zum Schluss werden die Aktivitäten über das ActiveMQ versendet.

Bei der Implementierung wurde darauf geachtet, dass eine Erweiterung ohne großen Aufwand möglich ist.
Um einen weiteren Sensortyp hinzuzufügen, muss nur eine neue Klasse erstellt werden, die von der Eventklasse erbt. Die neue Klasse muss einen Konstruktor haben, der alle Sensorwerte aufnimmt, oder einen String mit den enthaltenen Werten parsen kann. Weiterhin muss die Klasse die Methoden `equals(Object)` und `distanceTo(IEvent)` implementieren. Letztere Methode berechnet die Distanz zwischen einem Objekt dieser Klasse und dem übergebenen Event.

Für die Implementierung verwende ich die Programmiersprache Java und das JDK in der Version 1.7. Die Entwicklung des Programmcodes fand in der Entwicklungsumgebung Eclipse 4.2 (Juno) statt.
Als zusätzliche Bibliotheken verwende ich *Gson* von Google für die Verarbeitung der JSON-Nachrichten, die LivingPlaceMassaging (Version 1.0.3) Bibliothek und den Quellcode des Scriptsimultors (Version 0.2.2).
Das Projekt verwendet Maven für den Buildprozess und Git für die Versionsverwaltung.

5 Evaluierung

In diesem Kapitel werde ich die praktische Anwendung meiner Implementierung beschreiben. Dazu werden Sensordaten erzeugt, die anschließend der Aktivitätenentdeckung zugeführt werden. Es werden drei Versuche durchgeführt, in denen jeweils eine andere Zusammensetzung der Sensortypen untersucht wird. Dabei soll ermittelt werden, wie sich die Qualität der Aktivitätenentdeckung verändert, wenn bestimmte Sensortypen nicht zur Verfügung stehen. Weiterhin soll herausgefunden werden, wie sich unterschiedliche Fehlerraten der Sensordaten auf die Ergebnisse der Aktivitätenentdeckung auswirken.
Anschließend werden die entstandenen Ergebnisse erläutert.

5.1 Ausgangsdaten

Da es im Livingplace nicht möglich ist, Sensordaten in ausreichender Qualität zu erzeugen, werden sämtliche Sensordaten für diese Masterarbeit mit Hilfe des Scriptsimulators erzeugt.
Für die Erzeugung der Sensordaten werden zunächst mehrere kleinere Szenarien erstellt, die typische Aktivitäten eines Bewohners darstellen. Diese Szenarien werden wiederholt ausgeführt. Dazu werden aus den Szenarien sieben verschiedene Tage modelliert, die jeweils ein Tagesszenario bilden. Die sieben Tagesszenarien werden wiederum zu einem Szenario zusammengefügt, das eine komplette Woche eines Bewohners im Livingplace beschreibt.

Folgende Szenarien werden im Laufe der Woche wiederholt ausgeführt:

	Sequenzlänge		
Szenario	Versuch 1	Versuch 2	Versuch 3
Abwaschen	11	7	3
Ankleiden	13	3	10
Arbeiten	97	86	9
Duschen	10	6	4
Essen	78	70	8
Fernsehen	10	4	6
Hinsetzen	2	0	2
Kaffeekochen	23	13	10
Schlafen abends	17	0	17
Schlafen morgens	19	0	19
Toilette	8	4	4
Touchtable	13	13	0

Tabelle 5.1: Szenarien und Sequenzlängen

Die Szenarien sollen nach der Aktivitätenentdeckung in den Aktivitäten enthalten sein. Dabei wird berechnet, wie genau die Sequenzen der Szenarien in den Aktivitäten vorkommen.

Zusätzlich zu diesen festen, regelmäßigen Aktivitäten werden in den Tagesszenarien weitere Sensordaten erzeugt. Diese Sensordaten werden zwischen den Szenarien erzeugt und dienen dazu, den Tagesszenarien einen individuellen Charakter zu geben und die Entdeckung der Aktivitäten zu erschweren.
Alle Szenarien werden nacheinander abgespielt. Es gibt keine Überlappung der Szenarien.

Als Sensortypen werden alle Typen verwendet, die in 4.1 aufgeführt sind. Mit dem Wochenszenario werden verschiedene Datensets erzeugt. Diese Sets unterscheiden sich in der Fehlerrate, mit der die Sensordaten erzeugt werden.
Die Fehlerrate gibt an, wie genau die Sensordaten sind. Bei numerischen Werten der Sensordaten ist die Fehlerrate die Abweichung der Sensorwerte. Bei Sensordaten, die Zustände repräsentieren, gibt die Fehlerrate an, wie wahrscheinlich es ist, dass ein

falscher Zustand erzeugt wird. Zudem besteht die Möglichkeit, dass ein Sensorwert erst gar nicht erzeugt wird.
Für die Tests werden Fehlerraten von 0%, 5%, 10% und 20% verwendet. Durch die Fehlerrate der Sensordaten soll herausgefunden werden, wie zuverlässig die Aktivitätenentdeckung funktioniert, wenn die Sensordaten fehlerbehaftet sind.

Die Ausgangsdaten liegen alle als Dateien vor, wie in Kapitel 4.1 beschrieben.

5.2 Aktivitätenentdeckung

Für die Aktivitätenentdeckung werden verschiedene Datensets erstellt. Dazu wird das im vorherigen Kapitel beschriebene Wochenszenario mit unterschiedlichen Fehlerraten wiederholt ausgeführt und die erzeugten Sensordaten werden in unterschiedlichen Dateien abgespeichert.
Die Aktivitätenentdeckung läuft deterministisch ab, dadurch wird für ein bestimmtes Set an Ausgangsdaten immer dasselbe Ergebnis erzeugt. Bei der Erzeugung der Sensordaten durch den Scriptsimulator hingegen werden die Sensordaten zufällig verändert, wenn eine Fehlerrate größer als 0% vorgegeben worden ist. Wird das gleiche Skript mit einer Fehlerrate größer als 0% wiederholt abgespielt, wird somit jedes Mal ein leicht unterschiedliches Set an Sensordaten erzeugt.

Für die Ausgangsdaten mit 0% Fehlerrate bedeutet das, dass nur ein Datenset erzeugt werden muss. Für die Ausgangsdaten mit einer Fehlerrate von 5%, 10% und 20% hingegen werden jeweils 10 verschiedene Datensets erzeugt. Dadurch soll vermieden werden, dass Ausreißer, die zufällig erzeugt wurden, das Endergebnis verfälschen. Für das Endergebnis wird der Mittelwert aller 10 Datensets der jeweiligen Fehlerrate berechnet.

Um herauszufinden, wie die Aktivitätenentdeckung funktioniert, wenn bestimmte Sensortypen nicht berücksichtigt werden, werden sie beim Einlesen der Datensets ignoriert. Dadurch ist es nicht nötig, für jede Kombination an Sensortypen eigene Datensets zu erzeugen.

Anschließend an die Aktivitätenentdeckung wird für jedes Szenario aus dem vorherigen Kapitel berechnet, wie hoch die Übereinstimmung der Szenariensequenzen in den Aktivitäten ist. Dazu werden sämtliche Cluster nacheinander durchsucht, ob die Sequenz der Sensordaten des jeweiligen Szenarios in den Sequenzen der Muster des Clusters vollständig enthalten ist. Ist das der Fall, beträgt die Übereinstimmung 100%. Ist die Sequenz in keinem der Cluster vollständig enthalten, wird die geringste Distanz der Szenariosequenzen zu den Mustern der Cluster berechnet. Anschließend wird die Distanz im Verhältnis zu der Sequenzlänge gestellt. Das Ergebnis spiegelt wieder, wie hoch die Übereinstimmung der Szenariosequenzen in den Aktivitäten ist.

5.3 Ergebnis

In den folgenden Kapiteln werden drei Versuche durchgeführt, die jeweils eine andere Zusammensetzung der verwendeten Sensortypen beinhalten. In jedem Versuch wird untersucht, wie zuverlässig die Aktivitätenentdeckung arbeitet, wenn die Sensordaten unterschiedliche Fehlerraten aufweisen.
Anschließend werden die Ergebnisse verglichen und erläutert.

Bei den Abbildungen 5.1, 5.2 und 5.3 ist zu beachten, dass die Ergebnisse innerhalb der einzelnen Fehlerraten nach den Sequenzlängen der Szenarien aufsteigend sortiert sind.

5.3.1 Versuch 1: alle Sensortypen

Im ersten Versuch werden die verschiedenen Datensets mit allen Sensortypen verwendet. In Tabelle 5.2 und Abbildung 5.1 sind die Ergebnisse zu sehen.

	Übereinstimmung bei einer Fehlerrate von:			
Szenario	**0%**	**5%**	**10%**	**20%**
Abwaschen	48,7	58,1	53,9	45,6
Ankleiden	84,6	39,8	38,8	31,9
Arbeiten	30,0	26,5	26,0	21,7
Duschen	100,0	49,0	43,9	39,7
Essen	39,5	65,5	52,2	51,0
Fernsehen	60,0	60,0	48,2	48,7
Hinsetzen	100,0	100,0	92,0	80,5
Kaffeekochen	56,5	30,9	26,6	17,7
Schlafen abends	100,0	11,4	13,3	9,8
Schlafen morgens	94,7	64,0	58,3	45,5
Toilette	100,0	8,9	10,4	7,9
Touchtable	68,6	51,1	53,0	48,5
Durchschnitt	73,6	47,1	43,1	37,4

Tabelle 5.2: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, unter Berücksichtigung aller Sensortypen

Zunächst fällt auf, dass die Ergebnisse innerhalb derselben Fehlerrate stark schwanken. Bei einer Fehlerrate von 0% weisen die Szenarien *Duschen*, *Hinsetzen*, *Schlafen abends* und *Toilette* eine Übereinstimmung ihrer Sequenzen mit den entdeckten Aktivitäten von 100% auf. Das Szenario *Schlafen morgens* kommt immerhin noch auf 94,7%.
Dem gegenüber stehen die Szenarien *Abwaschen*, *Arbeiten* und *Essen*, deren Übereinstimmungen alle unterhalb von 50% liegen.
Im Durchschnitt liegt die Übereinstimmung aller Szenarien bei 73,6%

Bei den Datensets mit Fehlerraten ist zu sehen, dass die Übereinstimmung der Szenariensequenzen mit den entdeckten Aktivitäten stark abnimmt. Bereits bei einer Fehlerrate von 5% liegt die durchschnittliche Übereinstimmung bei 47,1% und nimmt bei den höheren Fehlerraten weiter ab.

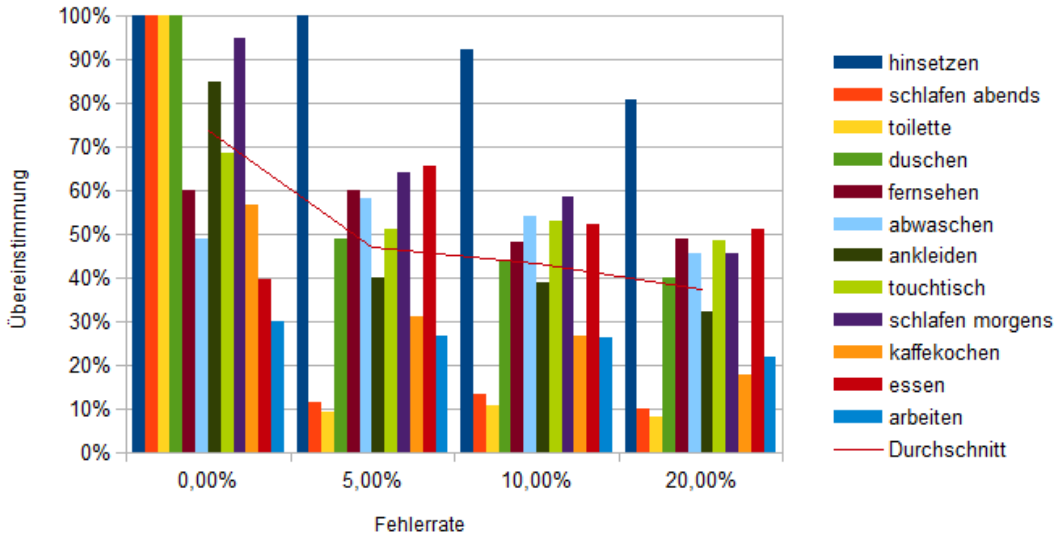


Abbildung 5.1: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, unter Berücksichtigung aller Sensortypen

Auffällig ist das Szenario *Hinsetzen*. Dieses Szenario erzielt bei allen Fehlerraten eine hohe Übereinstimmung zwischen 100% und 80,5%. Das wird daran liegen, dass das Szenario lediglich aus zwei Couchevents besteht. Dadurch ist es recht einfach, das Szenario in den Sensordaten als Aktivität zu erkennen.

Die Vermutung, dass kürzere Szenarien eine höhere Übereinstimmung bedeuten, bestätigt sich aber nicht. Die Ergebnisse lassen keine Tendenz erkennen, dass die Übereinstimmung abnimmt, je länger die Sequenzen werden.

5.3.2 Versuch 2: nur Positionsdaten

Im zweiten Versuch wird die Aktivitätenentdeckung nur mit Positionsdaten durchgeführt. In Tabelle 5.3 und Abbildung 5.2 sind die Ergebnisse zu sehen.

Szenario	Übereinstimmung bei einer Fehlerrate von: 0%	5%	10%	20%
Abwaschen	100,0	93,7	87,3	86,6
Ankleiden	98,3	87,1	84,5	71,0
Arbeiten	21,8	9,5	11,8	8,6
Duschen	93,7	94,5	92,6	94,7
Essen	28,5	11,7	14,3	10,6
Fernsehen	98,8	98,8	98,7	98,8
Hinsetzen	-	-	-	-
Kaffeekochen	100,0	57,2	49,9	45,9
Schlafen abends	-	-	-	-
Schlafen morgens	-	-	-	-
Toilette	100,0	100,0	99,0	97,4
Touchtable	87,9	56,1	53,4	46,8
Durchschnitt	81,0	75,1	73,0	62,3

Tabelle 5.3: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, nur unter Berücksichtigung von Positionsdaten

Die Szenarien *Schlafen abends*, *Schlafen morgens* und *Hinsetzen* enthalten keine Positionsdaten und können somit nicht entdeckt werden.

Im Vergleich zu den Ergebnissen von Versuch 1 ist eine deutliche Verbesserung zu erkennen. Bei dem Datenset mit einer Fehlerrate von 0% beträgt die durchschnittliche Übereinstimmung 81,00%. Bei den höheren Fehlerraten nimmt die durchschnittliche Übereinstimmung leicht ab und beträgt bei einer Fehlerrate von 20% noch 62.3%.

In diesem Versuch ist in Abbildung 5.2 die Tendenz zu erkennen, dass die Sequenzlängen der Szenarien Einfluss auf die Übereinstimmungen haben. Ab einer Länge von 13 nimmt die Übereinstimmung kontinuierlich ab. Die Szenarien *Essen* und *Arbeiten* haben eine Sequenzlänge von 70 bzw. 86 und erlangen bei allen Fehlerraten Übereinstimmungen von unter 30%.

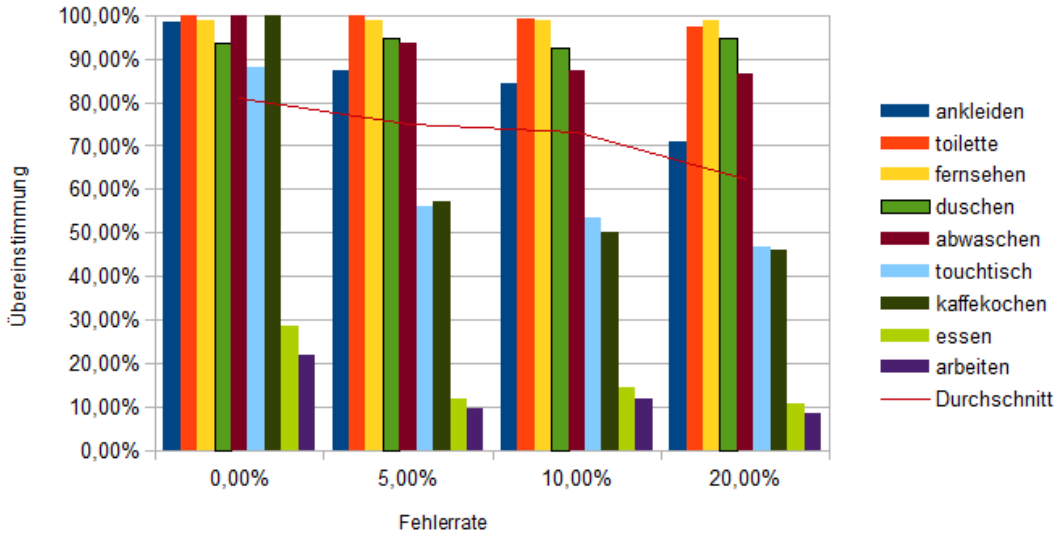


Abbildung 5.2: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, nur unter Berücksichtigung von Positionsdaten

5.3.3 Versuch 3: keine Positionsdaten

Der dritte Versuch bildet das Gegenstück zu Versuch 2. Hier werden alle Sensortypen verwendet, aber die Positionsdaten weggelassen. In Tabelle 5.4 und Abbildung 5.3 sind die Ergebnisse zu sehen.

Da das Szenario *Touchtable* ausschließlich Positionsdaten enthält, kann es in diesem Versuch nicht entdeckt werden.

Im Vergleich zu Versuch 1 wurden in diesem Versuch die durchschnittlichen Werte der Übereinstimmungen verbessert. Bei einer Fehlerrate von 0% liegt das Ergebnis auch über dem von Versuch 2. Bei den Fehlerraten von 5%, 10% und 20% fallen die durchschnittlichen Übereinstimmungen stärker ab als bei Versuch 2 und liegen zwischen den Ergebnissen von Versuch 1 und Versuch 2.

Wie in Versuch 2 ist auch hier die Tendenz zu erkennen, dass die Übereinstimmung abnimmt, je länger die Sequenzen der Szenarien werden.

Szenario	Übereinstimmung bei einer Fehlerrate von: 0%	5%	10%	20%
Abwaschen	50,0	48,0	47,0	41,0
Ankleiden	100,0	91,7	100,0	78,4
Arbeiten	90,9	64,5	43,7	35,0
Duschen	75,0	72,5	68,8	65,0
Essen	100,0	80,0	65,0	42,5
Fernsehen	100,0	83,3	66,7	49,0
Hinsetzen	100,0	100,0	89,5	85,5
Kaffeekochen	100,0	71,0	48,0	38,0
Schlafen abends	85,7	66,3	58,3	44,9
Schlafen morgens	94,7	32,5	26,6	17,7
Toilette	75,0	75,0	78,7	70,0
Touchtable	-	-	-	-
Durchschnitt	88,3	71,3	62,9	51,6

Tabelle 5.4: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, ohne Berücksichtigung von Positionsdaten

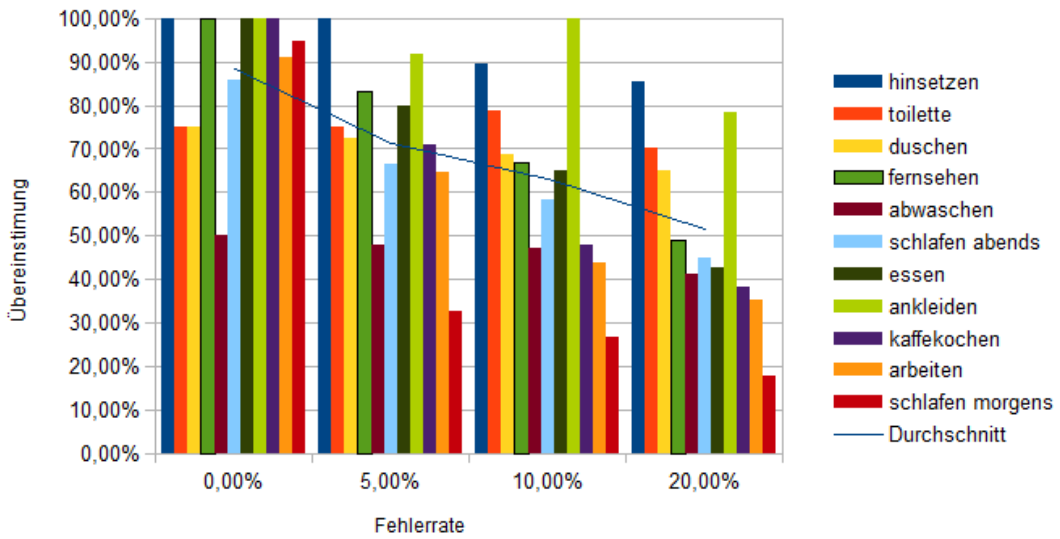


Abbildung 5.3: Übereinstimmungen [%] der entdeckten Aktivitäten mit den vorgegebenen Szenarien, ohne Berücksichtigung von Positionsdaten

5.3.4 Auswertung der Versuche

Die drei Versuche zeigen, dass die Sequenzlänge der Szenarien Einfluss auf die Höhe der Übereinstimmungen hat. Je länger die Sequenz, desto geringer fällt die Übereinstimmung aus. Das lässt darauf schließen, dass die Aktivitätenentdeckung bessere Ergebnisse produziert, wenn die zu entdeckenden Aktivitäten kurz sind.

	durchschnittliche Übereinstimmung bei einer Fehlerrate von:			
Versuch	**0%**	**5%**	**10%**	**20%**
1	73,6	47,1	43,1	37,4
2	81,0	75,1	73,0	62,3
3	88,3	71,3	62,9	51,6

Tabelle 5.5: Vergleich der durchschnittlichen Übereinstimmungen [%] aus den Versuchen 1 bis 3

Weiterhin ist zu beobachten, dass die Aktivitätenenteckung im Versuch 2 die besten Ergebnisse erzielen konnte.

Ein Kriterium für die Aktivitätenentdeckung sind die Distanzen zwischen den Events und Mustern. Im Versuch 2 werden die Distanzen ausschließlich zwischen Positionsdaten berechnet, während in den Versuchen 1 und 2 die Distanzen zwischen verschiedenen Typen berechnet werden. Wie in Kapitel 3.4 ist die Berechnung der Distanzen zwischen verschiedenen Typen schwierig. Das spiegelt sich auch in den Ergebnissen wieder.

5.4 Fazit

Die Durchführung der drei Versuche zeigt, dass meine Implementierung einer Aktivitätenentdeckung funktioniert. Die in Kapitel 2.4 gestellten Anforderungen können erfüllt werden.

Die höchste Übereinstimmung der Aktivitätenentdeckung liegt bei einer Fehlerrate von 0% bei 88,3%. Diese Fehlerrate setzt aber voraus, dass die Sensoren und die Übertragung fehlerfrei funktionieren. In der Praxis ist das aber leider selten der Fall. Daher sind die Ergebnisse mit einer Fehlerrate von 5% realistischer.
Hier liegen die Ergebnisse in den Versuchen 2 und 3 bei 75,1% und 71,6%. Diese Werte lassen sich mit denen der Projekte aus Kapitel 2.3 vergleichen und erfüllen die gesteckten Ziele aus Kapitel 2.4. Lediglich im Versuch 1 liegt die durchschnittliche Übereinstimmung bei 47,1%.

Die drei Versuche machen deutlich, dass die Auswahl der verwendeten Sensortypen Einfluss auf die Zuverlässigkeit der Aktivitätenentdeckung hat. In Versuch 1 wurden sämtliche zur Verfügung stehende Sensortypen verwendet. Das Ergebnis zeigt, dass dadurch keine zuverlässige Entdeckung erreicht wird.
In den Versuchen 2 und 3 hingegen konnten Ergebnisse von über 70% erreicht werden. Hier zeigt sich, dass das Weglassen von bestimmten Sensoren zu besseren Ergebnissen führt. Dabei ist aber zu beachten, dass die ausgewählten Sensoren auch tatsächlich bei den Aktivitäten ausgelöst werden. Ansonsten können die Aktivitäten gar nicht entdeckt werden.
Der Anteil der Positionsdaten in den verwendeten Datensets beträgt jeweils ca. 88% an den gesamten Sensordaten. Dadurch stehen für den Versuch 2 deutlich mehr Sensordaten zur Verfügung als für den Versuch 3.
Es reicht also nicht aus, nur den Fokus auf bestimmte Sensortypen zu legen, diese Sensoren müssen auch ausreichend Daten liefern, damit die Aktivitätenentdeckung zuverlässig arbeiten kann.

Um die Ergebnisse der Aktivitätenentdeckung zu verbessern sind zunächst bessere Ausgangsdaten nötig. Im Rahmen dieser Masterarbeit wurden simulierte Daten verwendet, die ausreichend sind, um die Funktionsweise des Verfahrens zu testen. Für eine

Weiterentwicklung und Verbesserung des Verfahrens sind aber reale Daten notwendig. Ansonsten besteht die Gefahr, dass das Verfahren nur für simulierte Daten funktioniert, aber nicht in der Praxis. Stehen diese Daten zur Verfügung, bieten die vergleichenden Arbeiten in Kapitel 2.3 Ansätze, wie man die Aktivitätenentdeckung weiter verbessern kann.

Ein Aspekt, der in dieser Masterarbeit nicht berücksichtigt wurde, ist der Datenschutz. Ein Hauptbestandteil dieser Masterarbeit ist, jede Bewegung und Aktion des Bewohners aufzuzeichnen und aus den entstandenen Daten Profile des Bewohners zu erstellen.
Im wissenschaftlichen experimentellen Umfeld mag es noch vertretbar sein, wenn der Datenschutz nicht beachtet wird, bzw. keine hohe Priorität hat. Wenn die Aktivitätenentdeckung und -interpretation aber im täglichen Leben in der Praxis angewandt wird, ist ein funktionierender Datenschutz unverzichtbar.
Hierfür muss ein Verfahren entwickelt und umgesetzt werden, dass den Datenschutz sicher stellt. Dazu gehört eine sichere Übertragung und Speicherung der Sensordaten sowie der entdeckten Aktivitäten.

6 Schluss

In diesem Kapitel wird die Masterarbeit zusammengefasst und erläutert, welche Ziele erreicht wurden. Im Einzelnen wird dargestellt, wo Schwierigkeiten aufgetaucht sind und wie ich sie gelöst habe. Zudem wird erläutert, was mir nach meiner Ansicht gut gelungen ist, bzw. was weniger gut gelungen ist.
Zum Schluss folgt einen Ausblick, wie die Ergebnisse dieser Masterarbeit in Zukunft weiterverwendet werden können.

6.1 Zusammenfassung

In dieser Masterarbeit habe ich mich mit der Clusterbildung von Sensordaten in einem Smarthome zum Zweck der Aktivitäteninterpretation beschäftigt. Ich habe dazu in dem Kapitel 2 die Grundlagen der Wissensentdeckung aus Datenbanken erläutert und dargelegt, welche Verfahren sich für diese Masterarbeit eignen und welche nicht. In Kapitel 2.3 habe ich beschrieben, wie andere Projektgruppen aus anderen Universitäten und Hochschulen dieses Thema bereits angegangen sind, welche Vor- und Nachteile die jeweiligen Verfahren haben und ob sie für diese Masterarbeit geeignet sind.
Auf dieser Grundlage habe ich das Verfahren des CASAS-Projektes als am geeignetsten für meine Masterarbeit befunden und darauf meinen Lösungsansatz begründet.

Das Design meines Programms für die Clusterbildung und Aktivitäteninterpretation der Sensordaten habe ich in Kapitel 3 beschrieben. Dazu habe ich die einzelnen Bestandteile des Programms, wie Klassen und Felder, erklärt und anschließend einen Programmablauf beschrieben.

Die Implementierung dieses Designs wurde anschließend im Kapitel 4 im Detail erläutert.

Die Ergebnisse und Daten aus der praktischen Anwendung meines Verfahrens ist in Kapitel 5 dokumentiert. Die Ergebnisse zeigen, dass die Qualität der Aktivitätenentdeckung stark von der Qualität der Sensordaten abhängt und davon, welche Sensortypen verwendet werden.

In Kapitel 5.4 habe ich die Ergebnisse der Versuche interpretiert und bin zu dem Schluss gekommen, dass das Verfahren grundsätzlich in der Lage ist, Aktivitäten in Sensordaten zu erkennen und die gesteckten Ziele aus Kapitel 2.4 erfüllt werden können.

6.2 Bewertung

In dieser Masterarbeit gab es diverse Herausforderungen zu bestehen, manche waren einfacher zu lösen als andere. Zu den größten Herausforderungen zählte die Beschaffung von Sensordaten, ohne die die Aktivitätenentdeckung nicht möglich wäre. Da die Sensordaten nicht durch das Livingplace selbst erzeugt werden können, wie in Kapitel 2.2, mussten alle Daten künstlich erzeugt werden.
Um das zu erreichen, habe ich im Vorwege dieser Masterarbeit den Scriptsimulator erstellt, mit dessen Hilfe die benötigten Sensordaten erzeugt werden können. Allerdings kann eine Simulation die Realität nie vollständig wiedergeben. Daher eignen sich simulierte Daten zwar um herauszufinden, ob das Verfahren generell funktioniert, für eine weitere Entwicklung sind aber reale Daten nötig.

Eine weitere Schwierigkeit dieser Masterarbeit war, die Berechnung der Distanzen zwischen Events und Sequenzen. Hierfür war es notwendig, in umfangreichen Tests die besten Werte für die Distanzen zu ermitteln. Die Evaluation meines Verfahrens hat gezeigt, dass eine genaue Auswahl der verwendeten Sensortypen Einfluss auf das Ergebnis hat. Wenn der Anteil der verschiedenen Sensortypen starke Unterschiede aufweist, ist es schwierig, die Distanzen gegeneinander zu gewichten.
An dieser Stelle kann das Verfahren weiter verbessert werden.

Der Quellcode für die Aktivitätenentdeckung wurde so entwickelt, dass jederzeit weitere Sensoren hinzugefügt werden können. Der Aufbau der Klassen und Pakete ist übersichtlich und kompakt gehalten, sodass Verbesserungen ohne großen Aufwand möglich sind.

6.3 Ausblick

Die Ergebnisse dieser Masterarbeit eignen sich sehr gut, um in der Zukunft erweitert zu werden bzw. von anderen Projekten aufgegriffen zu werden. Dabei sind verschiedene Ansätze möglich, von denen die interessantesten nachfolgend beschrieben werden.

Zur Zeit ist die Datenbasis der Aktivitätenentdeckung dateibasiert. Für diese Dateien werden alle Sensordaten im Voraus erzeugt. Hier sind zwei Verbesserungen sinnvoll. Zunächst sollte die persistente Speicherung der Sensordaten durch ein Datenbanksystem geschehen. Mit der MongoDB [BOE, 2010] steht dafür im Livingplace bereits eine Datenbank zur Verfügung. Dadurch kann erreicht werden, dass die Aktivitätenentdeckung besser von der Datenbasis getrennt ist und dass die Erzeugung und Zusammenstellung der Ausgangsdaten unabhängig von der Aktivitätenentdeckung geschehen kann.
Die zweite Verbesserung basiert auf der Speicherung der Sensordaten in einem Datenbanksystem. Momentan werden erst alle Daten erzeugt, um sie anschließend komplett zu verarbeiten. Vorteilhafter wäre es, wenn die Aktivitätenentdeckung bereits mit wenigen Sensordaten als Grundlage beginnt und die Aktivitäten kontinuierlich verbessert werden, sobald weitere Sensordaten erzeugt werden.

Das Datenbanksystem lässt sich auch dafür verwenden, die entdeckten Aktivitäten zu verwalten. Aktuell werden die Aktivitäten nach der Erzeugung über das ActiveMQ in einem eigene Topic bekanntgegeben. Damit die Aktivitäten aber auch von Projekten genutzt werden können, die bei der Erzeugung der Aktivitäten das Topic nicht abonniert haben, sollten sie in einer Datenbank persistent gespeichert werden.

Eine weitere Möglichkeit, die entdeckten Aktivitäten zu erweitern, ist die Verknüpfung mit der Ontologie [Staab and Studer, 2009] des Livingplace. Die Ontologie wurde

von Jens Ellenberg in seiner Masterarbeit [Ellenberg, 2012] entwickelt und für die Erkennung von Aktivitäten eingesetzt.

Die Auswertung der Aktivitäten kann auf mehrere Arten erfolgen. Eine Möglichkeit ist dabei das Event Stream Processing (ESP) [Luckham, 2001], mit dessen Hilfe aus einem Datenstrom Muster erkannt werden können.
Dabei werden Abfragen definiert, die dem SQL Syntax [Unterstein and Matthiessen, 2012] ähneln, die dann mit dem aktuellen Datenstrom abgeglichen werden. Diese Abfragen können dabei aus den Informationen der Aktivitäten erzeugt werden.
Dadurch lässt sich eine Steuerung der Wohnung erzeugen, die sich individuell auf die Gewohnheiten des Bewohners einstellt.

Wie gut sich die entdeckten Aktivitäten für eine Haussteuerung eignen, lässt sich momentan schlecht abschätzen. Dazu muss zunächst einmal eine entsprechende Steuerung für das Livingplace implementiert werden. Die Steuerung muss in der Lage sein, die entdeckten Aktivitäten zu interpretieren, um daraus Regeln für eine Steuerung zu generieren.

Zur Zeit ist die Aktivitätenentdeckung darauf ausgelegt, die Aktivitäten von einer einzelnen Person zu entdecken. Welchen Einfluss mehrere Personen auf die Aktivitätenentdeckung haben, kann nicht abgeschätzt werden. Interessant ist es hier, herauszufinden, ob Unterschiede in der Ausführung der Aktivitäten der einzelnen Personen zu einer verbesserten Aktivitätenentdeckung führen können.

Literaturverzeichnis

MongoDB - Sag Ja zu NoSQL. entwickler.press, Frankfurt am Main, 2010. ISBN 3-86802-057-8.

Von Geodaten bis NoSQL: Leistungsstarke PHP-Anwendungen. Hanser, München, 2012. ISBN 3-446-42995-6.

Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weiber. *Multivariate Analysemethoden : eine anwendungsorientierte Einführung*. Springer, Berlin, 11 edition, 2006. ISBN 978-354-02787-0-2.

Athanasios Bamis, Jia Fang, and Andreas Savvides. Discovering routine events in sensor streams for macroscopic sensing composition. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 408–409, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-988-6. doi: http://doi.acm.org/10.1145/1791212.1791278. URL http://doi.acm.org/10.1145/1791212.1791278.

Andreas Basener. Ausarbeitung aw1 drahtlose sensornetzwerke im kontext ambient assisted living. Technical report, HAW Hamburg, Februar 2011a. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-aw1/basener/bericht.pdf.

Andreas Basener. Ausarbeitung aw2 erlernen und erkennen von verhaltensmustern in sensordaten. Technical report, HAW Hamburg, August 2011b. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2011-aw2/basener/bericht.pdf.

Andreas Basener. Ausarbeitung pj1 entwicklung eines skriptsimulators für das living place. Technical report, HAW Hamburg, August 2011c. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2011-proj1/basener.pdf.

Andreas Basener. Ausarbeitung pj2 weiterentwicklung des skriptsimulators. Technical report, HAW Hamburg, Februar 2012. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master11-12-proj2/basener.pdf.

Ehrhard Behrends. *Elementare Stochastik : ein Lernbuch - von Studierenden mitentwickelt*. Vieweg+Teubner Verlag, Wiesbaden, 2013. ISBN 978-3-8348-2331-1.

Christoph Beierle and Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme*. Vieweg+Teubner, 2006. ISBN 978-3-8348-0504-1.

Michael W. Berry and Jacob Kogan. *Text mining : applications and theory*. John Wiley & Sons, Hoboken, NJ, USA, 2010. ISBN 978-0-470-74982-1.

Diane J. Cook and Lawrence B. Holder. Sensor selection to support practical use of health-monitoring smart environments. *ata Mining and Knowledge Discovery*, 2011. URL http://eecs.wsu.edu/~cook/pubs/dmkd11.pdf.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Wiley, New York, NY, USA, 2 edition, 2001. ISBN 978-0-471-05669-0.

Jens Ellenberg. Event stream processing mit esper unter einsatz von datamining verfahren. Master's thesis, Hochschule für Angewandte Wissenschaften, Hamburg, 2009. URL http://opus.haw-hamburg.de/volltexte/2009/819/.

Jens Ellenberg. Seminarausarbeitung klassifizierung von kontext in einer intelligenten wohnung. Technical report, HAW Hamburg, Februar 2011. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/ellenberg/bericht.pdf.

Jens Ellenberg. Ontologiebasierte aktivitätserkennung im smart home kontext. Masterarbeit, HAW Hamburg, Hamburg, Januar 2012. URL http:

//users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/ellenberg.pdf.

Jens Ellenberg, Bastian Karstaedt, Sören Voskuhl, Kai von Luck, and Birgit Wendholt. An environment for context-aware applications in smart homes. In *2011 INTERNATIONAL CONFERENCE ON INDOOR POSITIONING AND INDOOR NAVIGATION*, 2011. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/IPIN2011.pdf.

Martin Ester and Jörg Sander. *Knowledge Discovery in Databases*. Springer, 2000. ISBN 9783642583315.

Martin Ester, Hans peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

Jia Fang, Athanasios Bamis, and Andreas Savvides. Discovering routine events and their periods in sensor time series data. Technical report, Yale University, 2009. URL http://www.eng.yale.edu/enalab/publications/RTSS_Periodicity.pdf.

Usama Fayyad, Gregory Piatetsky-shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996.

Gernot A. Fink. *Mustererkennung mit Markov-Modellen - Theorie - Praxis - Anwendungsgebiete*. Vieweg+Teubner, 2003. ISBN 9783519004530.

Erich Gamma. *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 5. edition, 2001. ISBN 3-8273-1862-9.

Gesellschaft für Informatik E.V. Complex event processing (cep). online, Juni 2011. URL http://www.gi.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/complex-event-processing-cep-221.html.

Valerie Guralnik and George Karypis. A scalable algorithm for clustering sequential data. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM

'01, pages 179–186, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1119-8. URL http://dl.acm.org/citation.cfm?id=645496.658051.

Muhammad Raffay Hamid. *A Computational Framework For Unsupervised Analysis of Everyday Human Activities*. PhD thesis, Georgia Institute of Technologie, 2008. URL http://etd.gatech.edu/theses/available/etd-06232008-101404/.

Frank Hardenack. Das intelligente bett - sensorbasierte detektion von schlafphasen. Technical report, HAW Hamburg, Hamburg, 2011. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/hardenack.pdf.

Stephan Hartmann. The world as a process: Simulations in the natural and social sciences, August 2005. URL http://philsci-archive.pitt.edu/2412/.

Edwin O. Heierman, III and Diane J. Cook. Improving home automation by discovering regularly occurring device usage patterns. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 537–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1978-4. URL http://portal.acm.org/citation.cfm?id=951949.952078.

Ekbert Hering and Gert Schönfelder. *Sensoren in Wissenschaft und Technik : Funktionsweise und Einsatzgebiete*. Vieweg + Teubner, Wiesbaden, 2012. ISBN 3-8348-0169-0.

Knut Hildebrand, Marcus Gebauer, Holger Hinrichs, and Michael Mielke. *Daten- und Informationsqualität : Auf dem Weg zur Information Excellence*. Vieweg+Teubner Verlag, Wiesbaden, 2011. ISBN 978-3-8348-9953-8. URL http://dx.doi.org/10.1007/978-3-8348-9953-8.

Peter J. Huber. From large to huge: A statistican's reaction to kdd & dm. In *Third International Conference on Knowledge Discovery and Data Mining*.

John P. A. Ioannidis. Why most published research findings are false. *PLoS Med*, 2 (8):e124, 08 2005. doi: 10.1371/journal.pmed.0020124. URL http://dx.doi.org/10.1371%2Fjournal.pmed.0020124.

Brian D. Jones. Aware home research initiative research projects, 2011. URL http://awarehome.imtc.gatech.edu/about-us/GT_ahri-overview_1-13-2011.pdf.

Bastian Karstaedt. Entwicklung eines indoor spatial information services für smart homes auf basis der industry foundation classes. Technical report, HAW Hamburg, Februar 2011. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/karstaedt/bericht.pdf.

A. Ketterlin. Clustering sequences of complex objects. In *Third International Conference on Knowledge Discovery and Data Mining*.

Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.6429.

Donald Ervin Knuth. *The Art of Computer Programming 1. Fundamental Algorithms*. Addison-Wesley, 3. edition, 2002. ISBN 0-201-89683-4.

Daniel T. Larose. *Data mining methods and models*. Wiley, Hoboken, NJ, USA, 2006. ISBN 0-471-75648-2.

VI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966. URL http://www.bibsonomy.org/bibtex/21a29b294552edb63828d57f3495e2eb2/brightbyte.

David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0201727897.

Dimitrios Lymberopoulos, Athanasios Bamis, and Andreas Savvides. Extracting spatio-temporal human activity patterns in assisted living using a home sensor network. In

Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments, PETRA '08, pages 29:1–29:8, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-067-8. doi: http://doi.acm.org/10.1145/1389586.1389621. URL http://doi.acm.org/10.1145/1389586.1389621.

Ketil Malde, Eivind Coward, and Inge Jonassen. Fast sequenze clustering using a suffix array algorithm. Technical report, University of Bergen, 2003.

Leon Matthews. Using posix epoch timestamps. URL http://lost.co.nz/main/programming/epoch.html.

Rainer Merkl and Stephan Waack. *Bioinformatik interaktiv - Algorithmen und Praxis*. Vch Verlagsgesellschaft Mbh, Weinheim, 2005. ISBN 978-3527306626.

D. Minnen, T. Starner, I. Essa, and C. Isbell. Discovering characteristic actions from on-body sensor data. In *Wearable Computers, 2006 10th IEEE International Symposium on*, pages 11–18, 2006. doi: 10.1109/ISWC.2006.286337. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4067720.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997. ISBN 978-0070428072.

Gholamreza Nakhaeizadeh and Alexander Schnabl. Development of multi-criteria metrics for evaluation of data mining algorithms. In *Third International Conference on Knowledge Discovery and Data Mining*.

Mosawer Nurzai. Couch 2.0 - eine kontextsensitive installation für eine smart-home-umgebung. Technical report, HAW Hamburg, Hamburg, 2013. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/nurzai.pdf.

Otto Oberhauser. *Automatisches Klassifizieren : Entwicklungsstand - Methodik - Anwendungsbereiche*. Lang Verlag, Frankfurt am Main, 2005. ISBN 3-631-53684-4.

Kjell Otto and Sören Voskuhl. Weiterentwicklung der architektur des living place hamburg. Technical report, HAW Hamburg.

Alexander Pautz. Kabelloses sensornetzwerk im living place hamburg. Technical report, HAW Hamburg, Hamburg, 2010. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/pautz/bericht.pdf.

Sandeep Rana, Sanjay Jasola, and Rajesh Kumar. A hybrid sequential approach for data clustering using k-means and particle swarm optimization algorithm. In *International Journal of Engineering, Science and Technology*, volume 2, pages 167–176, 2010. URL http://www.ajol.info/index.php/ijest/article/view/63708/51535.

P. Rashidi and D.J. Cook. Mining sensor streams for discovering human activity patterns over time. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 431–440, 2010. doi: 10.1109/ICDM.2010.40. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5693997.

Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 2011. URL http://eecs.wsu.edu/~cook/pubs/tkde10.pdf.

P. Rieber. *Dynamische Webseiten in der Praxis: PHP, MySQL, CSS, Javascript, XHTML und Ajax*. mitp Professional. mitp-Verlag, 2009. ISBN 9783826617829.

Thomas A. Runkler. *Information Mining - Methoden, Algorithmen und Anwendungen intelligenter Datenanalyse*. Vieweg, Braunchweig/Wiesbaden, 2000. ISBN 3-528-05741-6.

Edmund Schiessle. *Sensortechnik und Meßwertaufnahme*. Vogel Verlag Und Druck, Würzburg, 1992. ISBN 9783802304705.

George Davey Smith. Data dredging, bias, or confounding. *BMJ*, 325:1437–1438, Dezember 2002. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1124898/.

B. Snyder, D. Bosanac, and R. Davies. *ActiveMQ in Action*. Manning Pubs Co Series. Manning Publications, 2011. ISBN 9781933988948.

Steffen Staab and Rudi Studer. *Handbook on Ontologies*. Springer Publishing Company, Incorporated, 2nd edition, 2009. ISBN 3540709991, 9783540709992.

Michael Unterstein and Günter Matthiessen. *Relationale Datenbanken und SQL in Theorie und Praxis*. Springer, Berlin, 5 edition, 2012. ISBN 978-3-642-28986-6.

Prof. Dr. Kai von Luck, Prof. Dr. Gunter Klemke, Sebastian Gregor, Mohammad Ali Rahimi, and Matthias Vogt. Living place hamburg – a place for concepts of it based modern living. Technical report, Hamburg University of Applied Sciences, Mai 2010. URL http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf.

Sören Voskuhl. Seminarausarbeitung entwicklung einer architektur für context aware systeme. Technical report, HAW Hamburg, Februar 2011. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/voskuhl/bericht.pdf.

Sören Voskuhl. Entwicklung einer architektur für context-aware systeme. Masterarbeit, HAW Hamburg, Hamburg, Januar 2012. URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/voskuhl.pdf.

Washington State University. Homepage, Juni 2011. URL http://www.wsu.edu/.

Ian H. Witten and Eibe Frank. *Datamining - Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005. ISBN 0-12-088407-0.

Yale University. Behaviourscope projekt homepage. URL http://www.eng.yale.edu/enalab/behaviorscope.htm.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 9. Dezember 2013 Andreas Basener