Universität Koblenz-Landau
Campus Koblenz
Fachbereich 4
Institute for Web Science and Technologies

Masterarbeit

„Service Line Detection von Bussen mit Hidden Markov Modellen“

von

Sven Milker
M.Sc. Wirtschaftsinformatik
4. Semester
Matrikel-Nummer: 209110013

Betreuer:

Dr. Heinrich Hartmann
Prof. Dr. Steffen Staab

Erstprüfer:

Prof. Dr. Steffen Staab

Zweitprüfer:

Dr. Heinrich Hartmann

Erklärung nach §16 Abs. 6 Satz 5 PO:

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Arbeit von mir selbstständig und ohne fremde Hilfe angefertigt worden ist. Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere, außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenden Abbildungen, Skizzen und Tabellen. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

Koblenz, den 30.03.2015

Sven Milker

Zusammenfassung

Die Service Line Detection (SLD) behandelt die Erkennung von Verbindungslinien öffentlicher Verkehrsmittel wie Bussen, Zügen oder Straßenbahnen, mit denen sich eine Person fortbewegt. Bisherige Untersuchungen konzentrierten sich vor allem auf die Strecke, die z.B. ein Bus entlang fährt. Diese Arbeit versucht den Zeitpunkt des Ein- und Aussteigens an einer Haltestelle zu berücksichtigen und damit auf die Linie zu schlussfolgern. Dafür wird ein Hidden Markov Modell (HMM) verwendet, das auf statischen Fahrplandaten basiert. Die zurückgelegte Strecke einer Person wird über GPS Koordinaten gespeichert und kann mit Hilfe des Viterbi-Algorithmus und dem HMM ausgewertet werden.

Die Implementierung wird mit von Live+Gov bereitgestellten Daten getestet, die von Personen im Raum Helsinki aufgezeichnet worden sind. Da die meisten Testpersonen mit Bussen unterwegs waren, liegt der Fokus dieser Arbeit auf Buslinien.

Die Ergebnisse zeigen, dass sich HMM grundsätzlich für die Erkennung von Buslinien eignen und sowohl Stärken als auch Schwächen gegenüber anderen Implementierungen aufweisen. Insgesamt wurden 21 Trips untersucht, von denen der implementierte Algorithmus neun korrekt erkannt hat, sechs teilweise und weitere sechs gar nicht. Gründe dafür und der Vergleich mit der naiven Methode von Live+Gov werden diskutiert.

Abstract

The service line detection (SLD) tries to identify the routes of public transports like busses, trains or trams which somebody uses to get from one place to the other. Most of the recent works concentrated on the route a e.g. bus is driving. This work tries to respect the time and place someone is getting on a bus. To achieve this, this work proposes the use of a hidden markov model (HMM) with static timetable data. The distance someone travels is represented by GPS coordinates which can be used by the Viterbi algorithm to calculate the route which the user took.

The implementation will be tested on data recorded by people from Helsinki for Live+Gov. Since most of them used busses, the focus of this work will be on the detection of bus lines.

The results are showing that HMM can be used for the SLD. Altogether, 21 trips were tested with different results. Nine lines were correctly detected, six were partially correct and six resulted in wrong lines. Reasons for that and the comparison to the naive Live+Gov method will be discussed.

Inhaltsverzeichnis

1 Einleitung

Die Service Line Detection (SLD) behandelt die Erkennung von Verbindungslinien öffentlicher Verkehrsmittel wie z.B. Bussen, Zügen oder Straßenbahnen, mit denen sich eine Person im Alltag fortbewegt. Das Wissen darüber kann für kontextsensitive Applikationen nützlich sein, aber findet auch direkte Anwendung durch die Auswertung von Bewegungsprofilen, bei denen öffentliche Verkehrsmittel benutzt worden sind. Diese können Informationen darüber liefern, wie sich Personen mit mehreren Linien hintereinander fortbewegen.

Bisherige Ansätze verfolgten verschiedene Strategien. Teils wurden Klassifizierungsalgorithmen wie Decision Trees [1] oder Bayessche Netze [2] verwendet, die mit Informationen von GPS Position und statischen Fahrplandaten trainiert wurden. Bei einem anderen Ansatz wurde über das Smartphone nach in der Nähe liegenden Mobilfunkmasten gesucht und mit vorliegenden Daten abgeglichen, um die Linie zu erkennen [3]. Live+Gov verwendet Echtzeitdaten der Position von öffentlichen Verkehrsmitteln, die mit der aktuellen Position des Benutzers verglichen werden [4].

Diese Arbeit versucht den Zeitpunkt des Ein- und Aussteigens an einer Haltestelle zu berücksichtigen, um damit auf die Linie schlussfolgern zu können. Dafür wird ein Hidden Markov Modell (HMM) verwendet, das einen stochastischen Ansatz verfolgt und als dynamisches bayessches Netz angesehen werden kann. Dabei wird die SLD als HMM modelliert, was eine Abänderung des klassischen HMM voraussetzt. Die benötigten Daten bestehen einerseits aus Fahrplandaten, mit denen das HMM aufgebaut wird und andererseits aus GPS Positionsdaten, welche die zurückgelegte Strecke einer Person speichern. Für die Fahrplandaten wird die GTFS (General Transit Feed Specification) verwendet [5]. Um aus diesen Daten die korrekte Linie eines öffentlichen Verkehrsmittels zu ermitteln, werden die GPS Daten auf Basis des Modells und mit Hilfe des so genannten Viterbi-Algorithmus ausgewertet.

Diese Vorgehensweise hat den Vorteil, dass keine Trainingsphase benötigt wird, da sämtliche Daten im GTFS Format vorliegen. Dazu werden keine Live-Daten von den öffentlichen Verkehrsmitteln verwendet, die ohnehin nicht für jede Stadt verfügbar sind. Fahrplandaten im GTFS werden von jeder der größten 25 Städte der USA angeboten, aber weniger als die Hälfte stellen Live-Daten bereit [1]. Die Anwendung auf neue Städte setzt also lediglich voraus, dass Fahrplandaten im GTFS Format vorliegen.

Die Implementierung wird mit Daten getestet, die nicht unter Laborbedingungen entstanden sind, sondern von verschiedenen Personen stammen, die sich wie üblich im Alltag im Großraum Helsinkis fortbewegten. Diese Testdaten wurden von Live+Gov aufgezeichnet und mit den Fahrplandaten von Mattersoft [6] und HSL [7] zur Verfügung gestellt. Der Fokus der Tests liegt dabei auf Buslinien, da die Personen der Testdaten vor allem mit Bussen unterwegs waren. Theoretisch ist die Vorgehensweise aber auf alle öffentlichen Verkehrsmittel anwendbar.

Die Ergebnisse zeigen, dass sich HMM grundsätzlich für die Erkennung von Buslinien eignen und sowohl Stärken als auch Schwächen gegenüber anderen Implementierungen aufweisen. Insgesamt wurden 21 Trips untersucht, von denen der implementierte Algorithmus neun korrekt erkannt hat, sechs teilweise und weitere sechs gar nicht. Gründe dafür und der Vergleich mit der naiven Methode von Live+Gov werden diskutiert.

1.1 Motivation

Die SLD kann in verschiedenen Bereichen eingesetzt werden. Das Wissen über die genommene Linie kann dabei direkte Verwendung finden oder lediglich eine zusätzliche Information für ein anderes Problem darstellen.

Die direkte Anwendung der SLD wird z.B. für eine Auswertung des Fahrverhaltens von Passagieren benötigt. Da die Benutzung öffentlicher Verkehrsmittel in Städten steigt, ist eine genaue Planung notwendig, um deren Einsatz zu verbessern. Daher ist es wichtig, das Benutzungsverhalten der Passagiere zu analysieren. So können einfache Zählungen durchgeführt werden, bei denen die Nutzung einer einzelnen Linie ermittelt wird. Allerdings wird dabei nicht ersichtlich, wie sich eine Person innerhalb des gesamten Nahverkehrs mit öffentlichen Verkehrsmitteln fortbewegt. Durch die SLD kann ausgewertet werden, welche verschiedenen Linien eine Person benutzt hat und wie die Linien zusammenhängen. Bei der Verwendung von z.B. mehreren Buslinien kann die durchschnittliche Wartezeit berechnet und auf Basis dieser Daten der Fahrplan verbessert werden, indem Anschlussbusse zu angepassten Zeiten fahren. Zusätzlich kann ermittelt werden, ob es auf bestimmten Teilabschnitten zu einer Verspätung kommt oder ob sich eine Verspätung aufgrund vieler kleinerer Verzögerungen ergibt. Da die Erkennung nicht nur auf Buslinien beschränkt ist, kann auch das Verhalten analysiert werden, bei dem verschiedene Arten öffentlicher Verkehrsmittel benutzt werden. So könnte die Anbindung des Nahverkehrs an den Fernverkehr untersucht werden.

Die SLD ist des Weiteren für kontextsensitive Problemstellungen relevant. Sie wird z.B. verwendet, um eine genauere Klassifizierung des Verkehrsmittels, mit dem eine Person unterwegs ist, zu ermöglichen [1, 2]. Das Wissen über das verwendete Verkehrsmittel kann u.a. für Werbezwecke verwendet werden, um Autofahrern z.B. Coupons für das Tanken anzubieten [2]. Ein weiteres Szenario ist die Berechnung der Ankunftszeit von Bussen, um die Abweichungen vom Fahrplan zu ermitteln. Dafür wird die aktuelle Position eines Busses und dessen Linie benötigt [3].

1.2 Aufgabenstellung/Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines Programmes, das mit Hilfe eines Hidden Markov Modells und dem Viterbi-Algorithmus aus einer Reihe von GPS Koordinaten die zugehörige(n) Linie(n) erkennt, falls die Daten beim Fahren mit einem öffentlichen Verkehrsmittel aufgenommen worden sind. Das Programm soll eine möglichst hohe Genauigkeit aufweisen, was mit Testdaten aus Helsinki überprüft wird. Zum Schluss wird die Herangehensweise von Live+Gov mit der Methode dieser Arbeit verglichen und die Ergebnisse diskutiert.

1.3 Related Works

Verwandte Arbeiten konzentrierten sich teilweise nicht direkt auf die SLD, sondern vielmehr auf die Erkennung des benutzten Verkehrsmittels wie Auto oder Bus. Um diese besser unterscheiden zu können, wurde versucht, den GPS Positionen eine Linie zuzuordnen [1, 2].

Eine andere Arbeit versuchte die Ankunftszeit von Bussen an den Haltestellen mit Hilfe eines Netzwerks aus Personen mit ihren Smartphones zu berechnen. Dabei sollte die Buslinie, mit denen

eine Person unterwegs war, erkannt und die Ankunftszeit an kommenden Haltestellen ermittelt werden. Die tatsächliche Ankunftszeit konnte dann von anderen Personen eingesehen werden [3].

Für die Erkennung der Linie gab es verschiedene Herangehensweisen. Bei der Verwendung von Klassifizierungsalgorithmen wurden verschiedene Eigenschaften aus GPS Positionen und Fahrplandaten berechnet, wie z.B. die Durchschnittsgeschwindigkeit- und Beschleunigung, die Nähe zu Bushaltestellen [2] oder der Fréchet Abstand und Abweichungen vom Fahrplan [1].

[3] ermittelt die umliegenden Mobilfunkmasten und erstellt daraus eine Sequenz, die mit Hilfe eines speziellen Algorithmus die wahrscheinlichste Buslinie berechnet. Bei diesem Verfahren werden keine GPS Informationen benötigt.

Zusätzlich zu statischen Daten über Haltestellen, Linien und Haltezeiten, mit denen man nur die theoretische Position eines z.B. Busses bestimmen kann, werden auch Echtzeitdaten der Position verwendet [2]. Auch die SLD von Live+Gov verwendet diese Echtzeitdaten, um die Linie zu bestimmen. Die Überprüfung läuft dabei wie folgt ab:

Eine Reihe von GPS Koordinaten mit Zeitstempel werden dabei als Input verwendet. Falls es sich um aktuelle Daten handelt, werden die aktuellen Positionen umliegender Verkehrsmittel ermittelt und gewichtet. In einem weiteren Schritt wird jede GPS Koordinate der Reihe mit den statischen Daten der Linien, zu denen die ermittelten Verkehrsmittel gehören, verglichen. Je höher die Übereinstimmung, desto wahrscheinlicher ist, dass diese Linie verwendet wird.

1.4 Aufbau der Arbeit

Die Arbeit ist in acht Kapitel unterteilt, die sich aufeinander aufbauen. Kapitel 2 befasst sich mit den relevanten Grundlagen der Wahrscheinlichkeitstheorie und speziellen Berechnungen, die u.a. Distanzen zwischen zwei Koordinaten auf der Erdoberfläche ermitteln.

Kapitel 3 behandelt die verschiedenen Arten von Hidden Markov Modellen und den Viterbi-Algorithmus, der für die Berechnung der Linien relevant ist.

Darauf aufbauend erstellt Kapitel 4 ein HMM für SLD und beschreibt die relevanten Aspekte. Dazu gehören die Übergangs- und Beobachtungswahrscheinlichkeiten, die Zustände und der mögliche Beobachtungsraum.

Anschließend wird in Kapitel 5 die dazugehörige Umsetzung erläutert. Das für die Fahrpläne verwendete Format GTFS wird in Kapitel 5.1 beschrieben. Für das Importieren dieses Formats wird das Projekt *Onebusaway* verwendet, dessen Aufbau Kapitel 5.2 und dessen notwendigen Änderungen Kapitel 5.3 behandelt.

Kapitel 6 beinhaltet die Evaluation der Umsetzung. Dazu gehört die Beschreibung der Testdaten, die verwendet worden sind (Kapitel 6.1) und die Testphasen. Die erste Testphase (Kapitel 6.2) dient dazu, erste Ergebnisse zu erhalten und Probleme bei der Erkennung zu analysieren. Die zweite Testphase setzt die ermittelten Probleme um und die neuen Ergebnisse werden mit den Ergebnissen von Live+Gov verglichen (Kapitel 6.3).

Abschließend wird in Kapitel 7 das Fazit dieser Arbeit gezogen und in Kapitel 8 ein Ausblick auf mögliche Erweiterungen gegeben.

2 Grundlagen

2.1 Wahrscheinlichkeitstheorie

Diese Arbeit verwendet elementare Wahrscheinlichkeitstheorie, wie sie z.B. in [8] dargestellt wird. Dazu gehört die bedingte Wahrscheinlichkeit, für die Abbildung 1 ein Beispiel zeigt. Die Wahrscheinlichkeit, dass Ereignis B eintritt, liegt bei 20 %. Die Wahrscheinlichkeit für Ereignis A liegt bei 6 %, was sich aus der Multiplikation der 20 % für Ereignis B und der 30 % von Ereignis B nach Ereignis A ergibt. Die Wahrscheinlichkeiten werden wie folgt notiert:

$P[A] = 0{,}06$

$P[B] = 0{,}2$

Die im Beispiel dargestellten 30 % stehen für die Wahrscheinlichkeit, dass das Ereignis A eintritt, wenn das Ereignis B bereits eingetreten ist. Dies ist die bedingte Wahrscheinlichkeit, die wie folgt notiert wird:

$P[A|B] = 0{,}3$

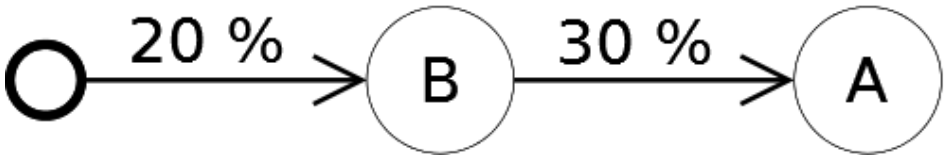

Abbildung 1 Beispiel einer bedingten Wahrscheinlichkeit

Die bedingte Wahrscheinlichkeit lässt sich dabei aus den einfachen Wahrscheinlichkeiten P[A] und P[B] berechnen. Für die zwei Ereignisse A, B ⊆ Ω mit P[B] ≠ 0 definieren wir:

$$\mathrm{P[A|B]} = \frac{\mathrm{P[A \cap B]}}{\mathrm{P[B]}}$$

P[_|B] ist ein Wahrscheinlichkeitsmaß auf Ω mit P[A|B] = 1 falls B ⊂ A.

Falls die bedingte Wahrscheinlichkeit P[A|B] nicht gegeben ist, kann man diese mit Hilfe des Satz von Bayes berechnen. Dafür wird die umgekehrte bedingte Wahrscheinlichkeit P[B|A] der zwei Ereignisse A, B mit P[A] ≠ 0, P[B] ≠ 0 benötigt:

$$P[A|B] = \frac{P[B|A] * P[A]}{P[B]}$$

Beweis:

$$\frac{P[B|A] * P[A]}{P[B]}$$

$$= \frac{P[A \cap \mathrm{B}]}{P[A]} * \frac{P[A]}{P[B]}$$

$$\frac{P[\mathrm{A} \cap \mathrm{B}]}{P[B]}$$

2.2 Berechnungen mit Latitude und Longitude

Im späteren Verlauf der Arbeit werden u.a. Distanzen zwischen GPS Koordinaten berechnet, die durch spezielle Formeln ermittelt werden [9]. Diese Berechnungen basieren auf der Annahme, dass die Erde rein sphärisch ist und ignorieren somit, dass die Erde leicht ellipsenförmig ist. Dies führt zu einem maximalen Fehler von 0,3 %, was eine ausreichende Genauigkeit darstellt.

Für die Formeln werden folgenden Symbole verwendet:

φ: Latitude (in Bogenmaß)

λ: Longitude (in Bogenmaß)

$\Delta\varphi$: Latitude Delta beider Koordinaten $(\varphi_2 - \varphi_1)$

$\Delta\lambda$: Longitude Delta beider Koordinaten $(\lambda_2 - \lambda_1)$

R: Radius der Erde: 6.371 km

Dazu seien P_1, P_2 zwei Koordinaten auf der Erde, die definiert sind als $P_1 = (\varphi_1, \lambda_1)$, $P_2 = (\varphi_2, \lambda_2)$.

Die Formeln verwenden die Funktion atan2(y, x), die für die Umrechnung von kartesischen Koordinaten in Polarkoordinaten benötigt wird. Viele Programmiersprachen beinhalten eine vorgefertigte Funktion von atan2, daher gibt es verschiedene Implementierungen. Eine davon ist die folgende [10]:

$$atan2(y,x) \coloneqq \begin{cases} \arctan\frac{y}{x} & \text{für } x > 0 \\ \arctan\frac{y}{x} + \pi & \text{für } x < 0, y \geq 0 \\ \arctan\frac{y}{x} - \pi & \text{für } x < 0, y < 0 \\ +\frac{\pi}{2} & \text{für } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{für } x = 0, y < 0 \\ 0 & \text{für } x = 0, y = 0 \end{cases}$$

2.2.1 Großkreis Distanz

Die Großkreis Distanz beschreibt die kürzeste Distanz zwischen zwei GPS Koordinaten über die Erdoberfläche, wobei Erhebungen im Terrain ignoriert werden.

Diese Distanz kann mit drei verschiedenen Methoden berechnet werden: Mit der Haversine Formel, mit den sphärischen Gesetzen des Kosinus oder mit einer Approximation.

Die Haversine Formel eignet sich speziell für kurze Distanzen und wird wie folgt berechnet:

$$a = \sin^2\frac{\Delta\varphi}{2} + \cos\varphi_1 * \cos\varphi_2 * \sin\frac{\Delta\lambda}{2}$$

$$c = 2 * \operatorname{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R * c$$

Die sphärischen Gesetze des Kosinus eignen sich für Distanzen ab wenigen Metern:

$$\mathrm{d} = \operatorname{acos}(\sin\varphi 1 \cdot \sin\varphi 2 + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \cos\Delta\lambda) \cdot \mathrm{R}$$

Die Approximation ist geeignet in Situationen, in denen die Berechnungsgeschwindigkeit relevant ist:

$$x = \Delta\lambda \cdot \cos\varphi_{\mathrm{m}}$$

$$\mathrm{y} = \Delta\varphi$$

$$d = R * \sqrt{\mathrm{x}^2 + \mathrm{y}^2}$$

φ_{m} bezeichnet dabei den Mittelpunkt der Latituden beider Punkte: $\frac{(\varphi_1 + \varphi_2)}{2}$.

2.2.2 Peilung

Der aktuelle Kurs bzw. die Richtung ändert sich, wenn man dem Pfad auf einem Großkreis folgt. Dies wird durch die Peilung ausgedrückt, die für weitere Berechnungen benötigt wird. Die Formel für die initiale Peilung von P_1 nach P_2 ist definiert als:

$$\theta = \operatorname{atan2}(\sin\Delta\lambda \cdot \cos\varphi_2, \cos\varphi_1 \cdot \sin\varphi_2 - \sin\varphi_1 \cdot \cos\varphi_2 \cdot \cos\Delta\lambda)$$

2.2.3 Cross-Track Distanz

Die Cross-Track Distanz beschreibt die Distanz zwischen einer GPS Koordinate und einem Großkreis. Dafür wird die bereits beschriebene initiale Peilung benötigt. Der Großkreis wird durch zwei Koordinaten definiert: P_1 und P_2. P_3 ist die Koordinate, deren Abstand zu dem Großkreis durch die anderen beiden Koordinaten bestimmt werden soll.

$$\mathrm{d_{xt}} = \operatorname{asin}(\sin(\delta_{13}) \cdot \sin(\theta_{13} - \theta_{12})) \cdot \mathrm{R}$$

wobei δ_{13}: Distanz von P_1 zu P_3
θ_{13}: Initiale Peilung von P_1 zu P_3
θ_{12}: Initiale Peilung von P_1 zu P_2

2.2.4 Along-Track Distanz

Die Along-Track Distanz basiert auf der Cross-Track Distanz und beschreibt die Distanz von P_1 zu einer Koordinate auf dem Großkreis, der die kürzeste Distanz zu P_3 hat:

$$d_{at} = \text{acos}(\frac{\cos(\delta_{13})}{\cos(\delta_{xt})}) \cdot R$$

wobei δ_{12}: Distanz von P_1 zu P_2
θ_{xt}: Cross-Track Distanz

2.2.5 Berechnung des Endpunktes von Startpunkt per Distanz

$\phi_2 = \text{asin}(\sin \phi_1 \cdot \cos \delta + \cos \phi_1 \cdot \sin \delta \cdot \cos \theta)$

$$\lambda_2 = \lambda_1 + \text{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \varphi_1, \cos \delta - \sin \varphi_1 \cdot \sin \varphi_2)$$

Dabei ist θ die Peilung und δ der Winkelabstand:

$$\delta = \frac{\text{Distanz}}{Erdradius}$$

3 Hidden Markov Modelle

Prozesse der realen Welt folgen bestimmten Regeln, die jedoch oft unbekannt sind und lediglich die daraus resultierenden Ergebnisse beobachtbar. Aus diesen Beobachtungen kann jedoch wiederum auf die zugrunde liegenden Prozesse geschlossen werden. Um dies besser zu verstehen, soll es an einem Beispiel demonstriert werden:

Man stelle sich einen Raum vor, in dem zwei gezinkte Münzen liegen, bei denen mit einer bestimmten Wahrscheinlichkeit Kopf oder Zahl vorkommt. Für einen Außenstehenden sind diese Wahrscheinlichkeiten unbekannt. Man weiß lediglich, welche Sequenz an Kopf oder Zahl geworfen worden ist, zu welcher Wahrscheinlichkeit die zu werfende Münze wechselt und mit welcher Wahrscheinlichkeit eine Münze zu Beginn geworfen wird.

Mit Hilfe dieser Informationen können verschiedene Probleme gelöst werden. Zunächst kann zu einer gegebenen Sequenz von Kopf und Zahl die Wahrscheinlichkeit ermittelt werden, zu der sie auftritt. Dazu kann die Sequenz der geworfenen Münzen bestimmt werden, welche zu diesem Ergebnis geführt hat.

Hidden Markov Modelle ermöglichen die Modellierung solcher Prozesse. Sie eignen sich aufgrund ihrer mathematischen Struktur und Zuverlässigkeit in der Praxis für viele verschiedene Probleme. Das komplette Wahrscheinlichkeitsmodell würde alle vorherigen Münzwürfe in die Berechnung mit einbeziehen, aber HMM betrachten lediglich den direkten Vorgänger:

$$P\left[q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k, \dots\right] = P[q_t = S_j, q_{t-1} = S_i]$$

Im Folgenden wird die Notation der HMM beschrieben, die an Rabiner angelehnt ist [11].

3.1 Elemente und Notation

HMM können einen diskreten oder kontinuierlichen Zustands- und Beobachtungsraum haben und zeitabhängig oder –unabhängig sein. Da kontinuierliche Zustandsräume für den weiteren Verlauf nicht weiter relevant sind, werden lediglich die HMM mit kontinuierlichen Beobachtungsräumen und der Zeitabhängigkeit betrachtet. Dadurch ergeben sich vier verschiedene Möglichkeiten, wie Tabelle 1 zeigt.

Tabelle 1 Übersicht der HMM Arten

	Diskreter Beobachtungsraum	**Kontinuierlicher Beobachtungsraum**
Nicht zeitabhängig	Klassisches HMM	Kontinuierliches HMM
Zeitabhängig	Zeitabhängiges HMM	Kont. Und zeitabh. HMM

Das klassische HMM ist nicht zeitabhängig und besitzt einen diskreten Beobachtungsraum. Dieses wird im Folgenden erläutert. Änderungen zu den anderen HMM werden anschließend behandelt.

3.1.1 Klassisches HMM

Zustandsraum

Ein HMM besteht aus einer Menge an Zuständen, für die jeweils eine bestimmte Wahrscheinlichkeit definiert ist, mit denen in diesen Zustand gewechselt und mit denen sie beobachtet werden kann. Die Menge an möglichen Zuständen ist definiert als:

$$\mathrm{S} = \{S_1, \dots, S_N\}$$

N ist somit die Anzahl möglicher Zustände des Modells. Betrachtet man einen Zustand zum Zeitpunkt t wird dies als q_t bezeichnet. Der Zustand S_i zum Zeitpunkt t wäre somit $q_t = S_i$. Im Folgenden betrachten wir zufällige Folgen von Ereignissen, die wir mit $q_t \in S$ bezeichnen. Diese so genannte Zustandssequenz ist eine Reihe an Elementen aus S und ist definiert als:

$$\mathrm{Q} = \{q_1, \dots, q_T\}$$

Im Beispiel sind zwei Zustände vorhanden, nämlich ein Zustand für jede der beiden Münzen. Es gilt also $S = \{M_1, M_2\}$, mit N = 2.

Beobachtungsraum

Die möglichen Beobachtungen bilden die Menge aller möglichen Ergebnisse des Prozesses, die beobachtet werden können und sind definiert als:

$$\mathrm{V} = \{v_1, \dots, v_M\}$$

M ist somit die Anzahl an möglichen Beobachtungen, die in einem Zustand auftreten können. Im Beispiel ist dies Kopf oder Zahl. Es gilt also V = {Kopf, Zahl}, mit M = 2.

Im Folgenden betrachten wir zufällige Folgen von Beobachtungen, die wir mit $O_t \in V$ bezeichnen. Eine Beobachtungssequenz ist eine Reihe an Elementen aus V und ist definiert als:

$$\mathrm{O} = \{O, \dots, O_T\}$$

Dabei gibt es keine Beschränkungen, wie oft eine Beobachtung auftreten kann.

Übergangswahrscheinlichkeiten von Zuständen

Zustände können mit bestimmten Wahrscheinlichkeiten wechseln. a_{ij} ist die Wahrscheinlichkeit, zu welcher der Zustand von i nach j wechselt und ist definiert als:

$$a_{ij} = P\left[q_{t+1} = S_j \,\middle|\, q_t = S_i\right], \qquad 1 \leq i, j \leq N$$

Da es sich um Wahrscheinlichkeiten handelt, gilt zusätzlich:

$$0 \leq a_{ij} \leq 1$$

Außerdem muss die Summe der Übergangswahrscheinlichkeiten des aktuellen Zustands in alle anderen 100 % ergeben:

$$\sum_{j=1}^{N} \mathrm{a_{ij}} = 1$$

Bezogen auf das Beispiel sind korrekte Übergangswahrscheinlichkeiten:

a_{M1M1} = 70 %, a_{M1M2} = 30 %

a_{M2M1} = 40 %, a_{M1M2} = 60 %

Abbildung 2 zeigt die Übergangswahrscheinlichkeiten des HMM mit den beschriebenen Wahrscheinlichkeiten. Die Summe der Wahrscheinlichkeiten aller ausgehenden Pfeile eines Zustandes ist stets 1, so dass die Nebenbedingung erfüllt ist.

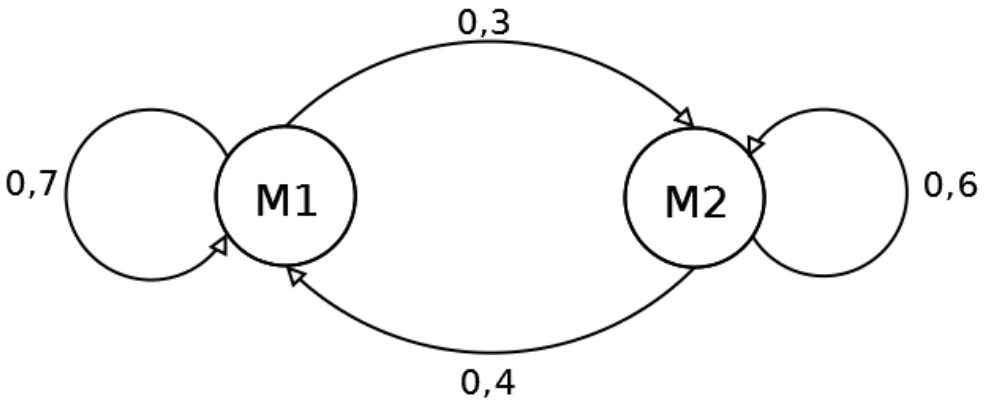

Abbildung 2 Übergangswahrscheinlichkeiten eines HMM

Beobachtungswahrscheinlichkeiten in Zuständen

Jede Beobachtung kann in jedem Zustand mit einer bestimmten Wahrscheinlichkeit eintreten. Sie ist als $b_j(k)$ definiert und beschreibt die Wahrscheinlichkeit, dass in Zustand j die Beobachtung k auftritt:

$$b_j(k) = P[O_t = v_k \mid q_t = S_j], \qquad 1 \leq j \leq N, 1 \leq k \leq M$$

Wie bei den Übergangswahrscheinlichkeiten gilt analog:

$$0 \leq b_j(k) \leq 1$$

Außerdem müssen die Wahrscheinlichkeiten einer Beobachtung in den verschiedenen Zuständen summiert 100 % ergeben:

$$\sum_{k=1}^{M} \mathrm{b_j}(k) = 1\,, \qquad 1 \leq \mathrm{j} \leq \mathrm{N}$$

Abbildung 3 zeigt die Beobachtungswahrscheinlichkeiten, die für das Beispiel festgelegt werden. Einfach umrandete Kreise sind dabei Zustände und dick umrandete Kreise mögliche Beobachtungen.

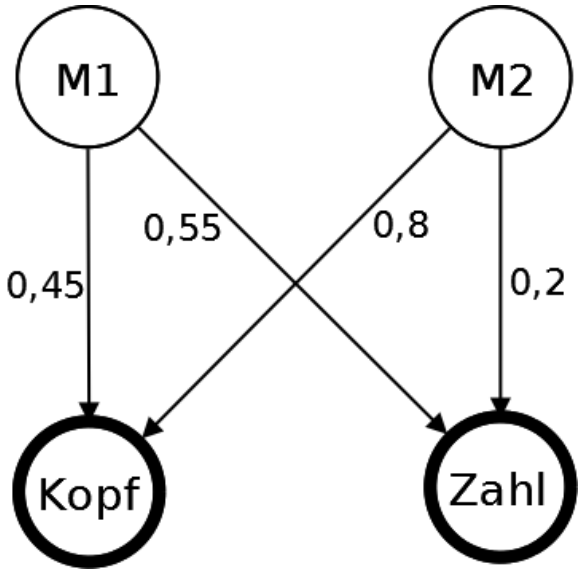

Abbildung 3 Beobachtungswahrscheinlichkeiten eines HMM

Anfangswahrscheinlichkeit von Zuständen

Jeder Zustand kann zu einer bestimmten Wahrscheinlichkeit den Anfangszustand darstellen. Die Anfangswahrscheinlichkeit für den Zustand i wird mit π_i notiert und ist definiert als:

$$\pi_i = P[q_1 = S_i], \qquad 1 < i < N$$

Die Summe aller Anfangswahrscheinlichkeiten muss 100 % ergeben:

$$\sum_{i=1}^{N} \pi_i = 1$$

Im Beispiel sind die Anfangswahrscheinlichkeiten wie folgt:

Münze 1: 40 %
Münze 2: 60 %

Zusammenfassung

Ein HMM besteht demnach aus den Mengen der Zustände S und der Beobachtungen V, den Anfangswahrscheinlichkeiten π_i, den Übergangswahrscheinlichkeiten a_{ij} und den Beobachtungswahrscheinlichkeiten $b_j(k)$.

Dazu besitzt es die folgenden Eigenschaften:

$$\sum_{i=1}^{N} \pi_i = 1 \qquad \sum_{j=1}^{N} a_{ij} = 1 \qquad \sum_{k=1}^{M} b_j(k) = 1$$

3.1.2 Kontinuierliches HMM

Im kontinuierlichen HMM sind die möglichen Beobachtungen unbegrenzt, so dass $V = \mathbb{R}^N$ gilt. Daher wird eine Wahrscheinlichkeitsdichtefunktion (WDF) verwendet, um die Wahrscheinlichkeiten zu beschreiben. Die Beobachtungswahrscheinlichkeit $b_j(k)$ kann dabei einfach mit der WDF ersetzt

werden [12], für die das Symbol β verwendet wird. Auch die dazugehörigen Bedingungen müssen abgeändert werden. Bei der WDF muss das Integral über das Intervall 1 ergeben. Für eine Funktion f gilt dabei allgemein:

$$\int_{\vec{x}\in\mathbb{R}^N}^{\infty} f(\vec{x})\ d\vec{x} = 1$$

Im HMM wird die Beobachtungswahrscheinlichkeit $b_j(k)$ also zu $\beta_j(k)$ mit $k \in V$, was die WDF darstellt. Die Bedingung dafür ist:

$$\int \beta_j(k)\mathrm{d}k = 1$$

Der einfachste Fall der WDF beschreibt dabei eine gleichmäßige Verteilung der Wahrscheinlichkeiten. Die Indikatorfunktion f(x, y) wird verwendet, um nur einen bestimmten Bereich zu betrachten. Die WDF ist dabei

$$\beta(k) = h * f(x, y)$$

wobei h die Wahrscheinlichkeit eines Punktes darstellt. Um dies zu verdeutlichen, soll ein Kreis als Beispiel dienen. Die dazugehörige Indikatorfunktion ist:

$$f(x,y) = \begin{cases} 0, & x^2 + y^2 > R^2 \\ 1, & x^2 + y^2 \leq R^2 \end{cases}$$

Zwischen den beiden Punkten x bzw. y, dem Radius r und dem Winkel φ besteht dabei ein Zusammenhang, der in Abbildung 4 dargestellt wird.

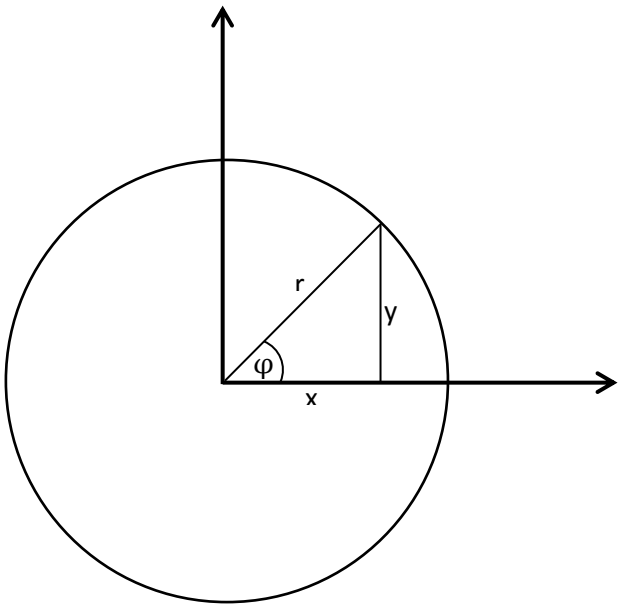

Abbildung 4 Darstellung des Verhältnis zwischen Koordinaten, Radius und Winkel

Über die Gesetzmäßigkeiten rechtwinkliger Dreiecke können x und y in Abhängigkeit von r und φ berechnet werden. Es gilt:

$x = r * cos\varphi$

$y = r * sin\varphi$

Die beiden Koordinaten können wie folgt in der Funktion ersetzt werden:

$$f(r * cos\varphi, r * sin\varphi) = \begin{Bmatrix} 0, (r * cos\varphi)^2 + (r * sin\varphi)^2 > R^2 \\ 1, (r * cos\varphi)^2 + (r * sin\varphi)^2 \leq R^2 \end{Bmatrix}$$

$$= \begin{Bmatrix} 0, r > R \\ 1, r \leq R \end{Bmatrix}$$

Um die Koordinaten x und y mit r und φ zu substituieren, wird die Jacobi Determinante benötigt.

$$\det \begin{pmatrix} \frac{dx}{dr} & \frac{dx}{d\varphi} \\ \frac{dy}{dr} & \frac{dy}{\mathrm{d}\varphi} \end{pmatrix} = \det \begin{pmatrix} cos\varphi & -r * sin\varphi \\ sin\varphi & r * cos\varphi \end{pmatrix}$$

= $cos\varphi * \mathrm{r} * \cos\varphi - \sin\varphi * (-\mathrm{r} * \sin\varphi) = r * (\cos^2\varphi + \sin^2\varphi)$ = r

Die Substitution mit der Jacobi Konstanten ergibt folgendes:

$$\int_{r=0}^{\infty} \left(\int_{\varphi=0}^{2\pi} \begin{Bmatrix} 0, & r > R \\ 1, & r \leq R \end{Bmatrix} * r \ \mathrm{d}r \right) d\varphi$$

Da die Funktion lediglich vom äußeren Integral abhängt, das über r integriert, kann das innere Integral nach außen gezogen werden:

$$\int_{r=0}^{\infty} \begin{Bmatrix} 0, & r > R \\ 1, & r \leq R \end{Bmatrix} * r \ dr * \int_{\varphi=0}^{2\pi} 1 \ d\varphi$$

Da für $r \geq R$ das Ergebnis stets null ist, kann das Intervall auch auf den Bereich zwischen r = 0 und R beschränkt werden.

$$= \ 2\pi * \int_{r=0}^{R} \begin{Bmatrix} 0, & r > R \\ 1, & r \leq R \end{Bmatrix} * r \ dr$$

Das Integral kann damit über die Stammfunktion von r

$$\mathrm{F(r)} \ = \frac{1}{2} r^2$$

gelöst werden:

$$= 2\pi * (\frac{1}{2} R^2 - 0)$$

$$= \pi * R^2$$

Dies entspricht der Formel des Flächeninhaltes eines Kreises. Über die gegebene Bedingung der Wahrscheinlichkeitsdichte lässt sich somit h bestimmen:

$$\int \beta_j(k)dk = 1$$

$$\int h * f(x,y)dxdy = 1$$

$$h * \int f(x,y)dxdy = 1$$

Dieses Integral wurde soeben berechnet und wird nun eingesetzt:

$$h * \pi * R^2 = 1$$

$$h = \frac{1}{\pi * R^2}$$

Das Integral ist also das Volumen unter dem Kreis, der 1 ergeben muss. Dies kann man sich als Zylinder vorstellen, wie Abbildung 5 zeigt. Die Höhe h ist dabei die Wahrscheinlichkeit.

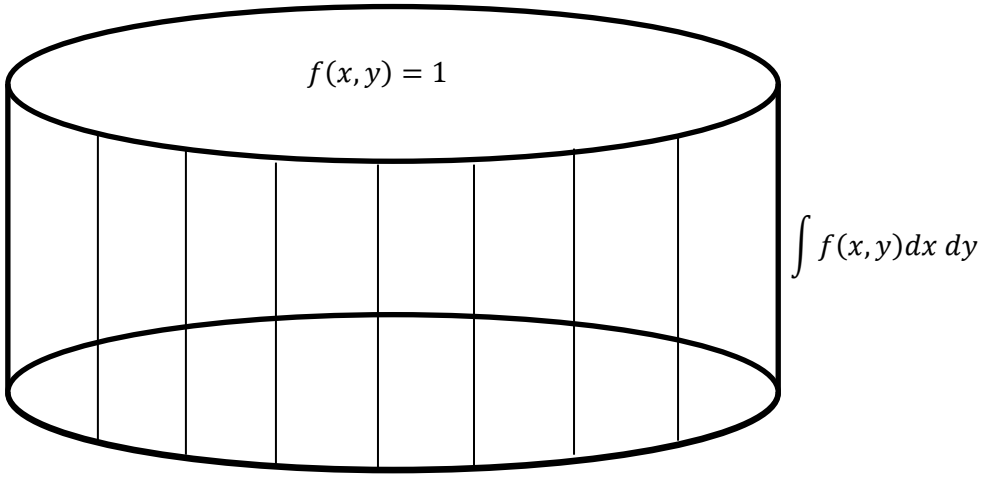

Abbildung 5 Graphische Darstellung der Wahrscheinlichkeitsdichte eines Kreises

Für Flächen allgemein gilt:

$$V = A * h$$

Und somit

$$h = \frac{1}{A}$$

Die Wahrscheinlichkeit ist somit lediglich vom Flächeninhalt abhängig.

3.1.3 Zeitabhängiges HMM

Bei einem zeitabhängigen HMM können sich die Wahrscheinlichkeiten je nach Zeitpunkt der Beobachtung ändern. Die Zeit ist dabei nicht kontinuierlich und besitzt somit eine endliche Anzahl an Zeitpunkten. Die im vorherigen Kapitel definierten Bedingungen müssen zu jedem Zeitpunkt gelten. Für Übergangswahrscheinlichkeiten gilt damit:

$$a_{ij}(t) = P\left[q_{t+1} = S_j \,\middle|\, q_t = S_i\right], \qquad 1 \le i, j \le N$$

$$0 \le a_{ij}(t) \le 1$$

$$\sum_{j=1}^{N} \mathrm{a_{ij}}(t) = 1$$

Das gleiche gilt für Beobachtungswahrscheinlichkeiten:

$$b_{jk}(t) = P[O_t = v_k \mid q_t = S_j], \qquad 1 \le j \le N, 1 \le k \le M$$

$$0 \le b_{jk}(t) \le 1$$

$$\sum_{k=1}^{M} \mathrm{b_{jk}}(t) = 1\,, \qquad 1 \le \mathrm{j} \le \mathrm{N}$$

Als letztes ist auch die Anfangswahrscheinlichkeit zeitabhängig:

$$\pi_i(t) = P[q_t = S_i], \qquad 1 < i < N$$

$$\sum_{i=1}^{N} \pi_{\mathrm{i}}(t) = 1$$

3.1.4 Kontinuierliches und zeitabhängiges HMM

Bei diesem HMM werden die Aspekte aus den beiden vorherigen Kapiteln miteinander kombiniert. Da lediglich die Beobachtungswahrscheinlichkeit kontinuierlich ist, wird nur diese beschrieben. Übergangs- und Anfangswahrscheinlichkeiten bleiben so wie in Kapitel 3.1.3 beschrieben.

Für kontinuierliche und zeitabhängige Beobachtungswahrscheinlichkeiten gilt:

$$\beta_{\mathrm{jk}}(t) = h * f(x, y)$$

$$\int \beta_{\mathrm{jk}}(t)\mathrm{dk} = 1$$

3.2 Viterbi-Algorithmus

Der Viterbi-Algorithmus ist eine optimierte Herangehensweise für die Berechnung der Zustandssequenz, welche die höchste Wahrscheinlichkeit zu der gegebenen Beobachtungssequenz erreicht. Dabei werden Zwischenergebnisse gespeichert, um redundante Berechnungen zu vermeiden.

3.2.1 Funktionsweise

Der Algorithmus wird in zwei Berechnungsabschnitte eingeteilt. Zunächst wird die gegebene Beobachtungssequenz Schritt für Schritt abgearbeitet und in jedem Schritt für jeden Zustand die Wahrscheinlichkeit gespeichert, die für das Erreichen dieses Zustands vorliegt. Die Wahrscheinlichkeit, dass Zustand A in Schritt t vorkommt, wird dabei als $\delta_t(A)$ notiert. Zusätzlich wird auch der Vorzustand gespeichert, der zu dieser maximalen Wahrscheinlichkeit für den aktuellen Zustand geführt hat. Dieser wird als $\psi_t(A)$ notiert.

Der zweite Berechnungsschritt verwendet die gespeicherten Informationen über den Vorzustand und ermittelt die wahrscheinlichste Zustandssequenz über den gesamten Zeitraum der Beobachtungen.

Die gesamte Berechnung unterteilt sich in vier Berechnungsschritte, wie Tabelle 2 zeigt:

Tabelle 2 Berechnungsschritte des Viterbi-Algorithmus

Schritt	Berechnung	Bemerkung
Initialisierung	$\delta_1(\mathrm{i}) = \pi_\mathrm{i} * \mathrm{b_i}(\mathrm{O_i})$ $1 \leq i \leq N$ $\psi_1(\mathrm{i}) = 0$	Die initiale Wahrscheinlichkeit wird mit der Beobachtungswahrscheinlichkeit multipliziert. Da es keinen Vorzustand gibt, wird dieser Werte in $\delta_1(\mathrm{i})$ gespeichert. $\psi_1(\mathrm{i})$ wird in diesem Schritt nicht benötigt und gleich null gesetzt.
Rekursion	$\delta_\mathrm{t}(j) = \max_{1 \leq i \leq N}[\delta_{\mathrm{t-1}}(\mathrm{i}) * \mathrm{a_{ij}}] * b_j(O_t)$ $2 \leq t \leq T$ $1 \leq j \leq N$ $\psi_\mathrm{t}(j) = \operatorname{argmax}_{1 \leq i \leq N}[\delta_{\mathrm{t-1}}(\mathrm{i}) * \mathrm{a_{ij}}]$ $2 \leq t \leq T$ $1 \leq j \leq N$	$\delta_\mathrm{t}(j)$ speichert für den aktuellen Zustand j zum Zeitpunkt t die größte Wahrscheinlichkeit, die sich aus der Multiplikation der Wahrscheinlichkeit des Vorzustandes $\delta_{\mathrm{t-1}}(\mathrm{i})$, der Übergangswahrscheinlichkeit des Vorzustandes in den aktuellen Zustand $\mathrm{a_{ij}}$ und der Beobachtungswahrscheinlichkeit $\mathrm{b_j}(\mathrm{O_t})$ ergibt. Es wird dabei über alle Zustände als Vorzustand iteriert und die höchste Wahrscheinlichkeit übernommen. $\psi_\mathrm{t}(j)$ speichert den zu der höchsten Wahrscheinlichkeit gehörigen Zustand, wobei die Beobachtungswahrscheinlichkeit nicht beachtet werden muss, da sie für den aktuellen Zustand immer gleich ist und nicht vom Vorzustand abhängt.
Termination	$p^* = \max_{1 \leq i \leq N} \delta_\mathrm{T}(\mathrm{i})$ $\mathrm{q_T}^* = \max_{1 \leq i \leq N} \delta_\mathrm{T}(\mathrm{i})$	Der Algorithmus terminiert, sobald die letzte Beobachtung untersucht und berechnet wurde. Danach wird über alle $\delta_\mathrm{T}(\mathrm{i})$ iteriert und die höchste Wahrscheinlichkeit in p^* und der der dazugehörige Zustand in q_T^* gespeichert.
Backtracking	$\mathrm{q_t^*} = \psi_{\mathrm{t+1}}(\mathrm{q_{t+1}^*})$ t = T-1, T-2, …, 1	Beim Backtracking wird über die in den vorherigen Schritten in ψ gespeicherten Zuständen rückwärts iteriert. Dabei wird beim vorletzten Wert in ψ angefangen und zu jedem folgenden Zeitpunkt über den Nachfolger der neue Zustand ermittelt.

3.2.2 Herleitung

Gesucht wird zunächst die Wahrscheinlichkeit einer Zustandssequenz $Q = q_1, q_2 \dots q_T$, welche die Beobachtungssequenz $O = O_1,\ O_2 \dots O_T$ am besten beschreibt, also argmax P[Q|O].

Nach den Regeln von Bayes gilt $\mathrm{P[O|Q]} = \frac{P[Q,O]}{P[O]}$. Da die Zustandssequenz maximiert werden soll, gilt argmax P[Q|O] = argmax P[Q, O], da P[O] konstant ist und auf das Maximieren von P[Q, O] keinen Einfluss hat. Der Viterbi-Algorithmus arbeitet daher auch bei der normalen Wahrscheinlichkeit mit dieser Annahme.

Für den Viterbi-Algorithmus gilt:

$$\delta_t(S) = \max_{s_1,\dots,s_{t-1}} P[q_1 = s_1, \dots, q_{t-1} = s_{t-1}, q_t = S, O_1 = v_1, \dots, O_t = v_t]$$

Durch $q_t = S$ wird der letzte Zustand genau festgelegt und die maximale Wahrscheinlichkeit wird von allen möglichen Pfaden der Vorzustände ermittelt. Für t = 1 gilt:

$$\delta_1(S) = \max_S P[q_1 = S, O_1 = v_1]$$

$$= \max_S P[O_1 = v_1 \mid q_1 = S] * P[q_1 = S]$$

$$= \max_S b_S(v_1) * \pi(S)$$

Für t = 2 gilt:

$$\delta_2(S) = \max_{s_1,S} P[q_1 = s_1, q_2 = S, O_1 = v_1, O_2 = v_2]$$

$$= \max_{s_1,S} P[O_2 = v_2 \mid q_1 = s_1, q_2 = S, O_1 = v_1] * P[q_1 = s_1, q_2 = S, O_1 = v_1]$$

Da in diesem Fall O_2 weder von der Beobachtung O_1 noch vom Zustand q_1 abhängt, können diese gestrichen werden:

$$= \max_{s_1,S} P[O_2 = v_2 | \cancel{q_1 = s_1}, q_2 = S, \cancel{O_1 = v_1}] * P[q_1 = s_1, q_2 = S, O_1 = v_1]$$

$$= \max_{s_1,S} P[O_2 = v_2 | \, q_2 = S] * P[q_2 = S \mid q_1 = s_1, \cancel{O_1 = v_1}] * \; P[q_1 = s_1, O_1 = v_1]$$

$$= \max_{s_1,S} P[O_2 = v_2 | \, q_2 = S] * P[q_2 = S \mid q_1 = s_1] * \; P[O_1 = v_1 \mid q_1 = s_1] * \; P[\, q_1 = s_1]$$

$$= b_S(v_2) * a_{s1S} * b_{s1}(v_1) * \; \pi(s_1)$$

Um zu zeigen, dass sich die Multiplikationskette für jedes weitere t erweitert, wird die Induktion verwendet. Der Induktionsanfang mit t = 1 wurde bereits gezeigt. Im Folgenden wird der Induktionsschritt mit t+1 behandelt:

$$\delta_{t+1}(S) = \max_{s_1,\dots,s_t} P[q_1 = s_1, \dots, q_t = s_t, q_{t+1} = S, O_1 = v_1, \dots, O_{t+1} = v_{t+1}]$$

$$= \max_{s_1,\dots,s_t} P[O_{t+1} = v_{t+1} \mid q_{t+1} = S, \cancel{q_1 = s_1, \dots, q_t = s_t, O_1 = v_1, \dots, O_t = v_t}] * P[q_1 = s_1, \dots, q_{t+1} = s_{t+1}, \; O_1 = v_1, \dots, O_t = v_t]$$

$$= \max_{s_1,\dots,s_t} P[O_{t+1} = v_{t+1} \mid q_{t+1} = S] * P[\, q_{t+1} = S \mid q_t = s_t, \cancel{q_1 = s_1, \dots, q_{t-1} = s_{t-1}, O_1 = v_1, \dots, O_t = v_t}] * \; P[q_1 = s_1, \dots, q_t = s_t, \; O_1 = v_1, \dots, O_t = v_t]$$

$$= \max_{s_1,\dots,s_t} b_S(v_{t+1}) * a_{s_t S} * P[q_1 = s_1, \dots, q_t = s_t,\ O_1 = v_1, \dots, O_t = v_t]$$

$$= b_S(v_{t+1}) * \max_{s_1,\dots,s_t} P[q_1 = s_1, \dots, q_t = s_t,\ O_1 = v_1, \dots, O_t = v_t] * a_{s_t S}$$

$$= b_S(v_{t+1}) * \max_{T} \left(\max_{s_1,\dots,s_{t-1}} P[q_1 = s_1, \dots, q_t = T,\ O_1 = v_1, \dots, O_t = v_t] *\ a_{s_t S} \right)$$

$$= b_S(v_{t+1}) * \max_{T} (\delta_t(T) * a_{TS})$$

3.2.3 Beispiel-Rechnung

Um den Viterbi-Algorithmus besser zu verstehen, wird mit dem bereits beschriebenen Modell die wahrscheinlichste Zustandssequenz einer beispielhaften Beobachtungssequenz berechnet. Für die Übersichtlichkeit zeigt Abbildung 6 das gesamte Modell, das für die folgende Beispielrechnung verwendet wird.

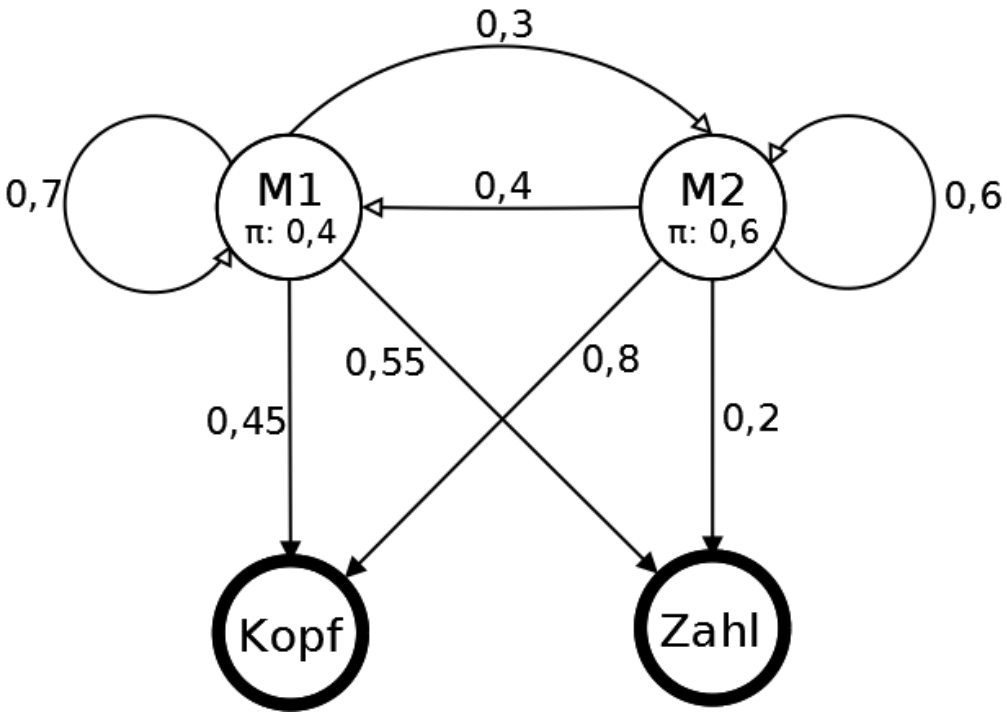

Abbildung 6 Beispiel eines Hidden Markov Modells

Die folgende Beobachtungssequenz soll für das Beispiel verwendet werden:

O = { Zahl, Kopf, Kopf }

Tabelle 3 zeigt den Forward-Backwards Algorithmus für das Beispiel und Tabelle 4 das dazugehörige Backtracking.

Tabelle 3 Forward-Backwards Algorithmus von Viterbi

Schritt #	Berechnung	Münze 1 (M1)	Münze 2 (M2)
1: Initialisierung, O = Zahl	$\delta_1(i) = \pi_i * b_i(\text{Zahl})$ $\psi_1(i) = 0$	$\delta_1(M1) = 0{,}22$ $\psi_1(M1) = 0$	$\delta_1(M2) = 0{,}12$ $\psi_1(M2) = 0$
2	$\delta_2(j) = \max_{1 \le i \le N} [\delta_1(i) * a_{ij}] * b_j(Kopf)$ $\psi_2(j) = \operatorname{argmax}_{1 \le i \le N} [\delta_1(i) * a_{ij}]$	i = M1: 0,0693 i = M2: 0,0216 $\delta_2(M1) = 0{,}0693$ $\psi_2(M1) = M1$	i = M1: 0,0528 i = M2: 0,0576 $\delta_2(M2) = 0{,}0576$ $\psi_2(M2) = M2$
3	$\delta_3(j) = \max_{1 \le i \le N} [\delta_2(i) * a_{ij}] * b_j(Kopf)$ $\psi_3(j) = \operatorname{argmax}_{1 \le i \le N} [\delta_2(i) * a_{ij}]$	i = M1: 0,0218295 i = M2: 0,010368 $\delta_3(M_1) = 0{,}0218295$ $\psi_3(M1) = M1$	i = M1: 0,016632 i = M2: 0,027648 $\delta_3(M2) = 0{,}027648$ $\psi_3(M2) = M2$
4: Termination	$p* = \max_{1 \le i \le N} \delta_3(i)$ $q_3* = \max_{1 \le i \le N} \delta_3(i)$	P* = 0,027648 $q_3* = M2$	

Tabelle 4 Backtracking von Viterbi

Schritt #	Berechnung	Zustand
1	$q2* = \psi_3(M2)$	M2
2	$q1* = \psi_2(M2)$	M2

Daraus ergibt sich die Zustandssequenz M2 → M2 → M2 mit einer Wahrscheinlichkeit von 2,7648 %.

4 Modellierung

Um die Hidden Markov Modelle auf die Service Line Detection zu übertragen, müssen diverse Anpassungen durchgeführt werden. Das HMM ist kontinuierlich und zeitabhängig. Der kontinuierliche Beobachtungsraum ist dazu kein einfaches Koordinatensystem, sondern besteht aus GPS-Koordinaten mit Longitude, Latitude und Altitude. Die Berechnung der Distanzen ändert sich, was im Folgenden beschrieben wird.

Eine weitere wichtige Änderung ist die Modellierung der Zustände des HMM. In den beschriebenen HMM wurde nicht zwischen Zustandsarten unterschieden, sondern ein Zustand wurde behandelt wie jeder andere und lediglich Beobachtungs- und Übergangswahrscheinlichkeiten waren verschieden. Für die SLD werden Zustandsarten benötigt, da die Berechnung der Wahrscheinlichkeiten je nach Art verschieden ist. Das folgende Kapitel geht auf diese Zustände ein.

4.1 Zustände

Die Zustände unterteilen sich in drei Zustandsarten. Die erste Zustandsart ist *Neutral*, von der es immer nur einen einzigen Zustand gibt, der im Folgenden als *neutraler Zustand* bezeichnet wird. Dieser Zustand bezeichnet die Momente, in denen eine Person sich weder an einer Haltestelle noch in einer Linie befindet.

Die beiden anderen Zustandsarten sind Haltestellen und Linien. Es wird die folgende Notation verwendet:

H Bezeichnet die Menge aller Haltestellen und $\#H$ deren Anzahl. Nach dem gleichen Prinzip werden Linien (L), der neutrale Zustand (N) und die Menge aller Zustände (S) notiert.

Die drei verschiedenen Zustandsarten ergeben die folgende Zustandsmenge:

$$S = H \cup L \cup \{ N \}$$

Die Anzahl der Zustände ist je nach Busliniennetz demnach unterschiedlich, bestehend aus der Summe aller Haltestellen und Linien, und dem neutralen Zustand:

$$\#S = \#H + \#L + 1$$

Jede Zustandsart unterscheidet sich bezüglich der grundlegenden Berechnung von Übergangswahrscheinlichkeit und Beobachtungswahrscheinlichkeit. Zwischen unterschiedlichen Zustandsarten sind die Berechnungen gleich.

4.2 Mögliche Beobachtungen

Die Menge möglicher Beobachtungen ist entweder die Menge aller GPS-Koordinaten oder eine begrenzte Menge in dem Gebiet, in dem die relevanten Haltestellen und Linien liegen. Für die Berechnung der Distanz werden lediglich die Latitude und die Longitude verwendet. Der Beobachtungsraum ist damit:

$$V = \mathbb{R}^2$$

Allerdings ist der Zahlenraum für Latitude und Longitude eingeschränkt, da die Werte im Dezimalgrad gespeichert sind und nur in einem bestimmten Bereich valide GPS Koordinaten darstellen.

Bei der Latitude ist dies ein Bereich zwischen -90 und 90 und bei der Longitude zwischen -180 und 180, wobei das negative Vorzeichen Süden bzw. Westen darstellt:

$$V = [-180, 180] \times [-90, 90] \subseteq \mathbb{R}^2$$

4.3 Übergangswahrscheinlichkeiten

Tabelle 5 zeigt eine Übersicht, welche generellen Übergänge zwischen Zustandsarten möglich sind.

Tabelle 5 Generelle Übergangsmöglichkeiten zwischen Zustandsarten

Von \ Nach	Haltestelle	Linie	Neutral
Haltestelle	Nur möglich bei gleicher Haltestelle	Möglich, wenn Linie an Haltestelle hält	Möglich
Linie	Möglich, wenn Linie an Haltestelle hält	Nur möglich bei gleicher Linie	Nicht möglich
Neutral	Möglich	Nicht möglich	Möglich

Die Übergangswahrscheinlichkeiten sind zeitabhängig und werden als $a_{ST}(t)$ definiert, wobei der Übergang von Zustand S in Zustand T beschrieben wird. Auch bei der Zeitabhängigkeit muss die Menge aller Übergangswahrscheinlichkeiten 100 % ergeben:

$$\sum_{T} a_{ST}(t) = 1$$

Im Folgenden werden die genauen Übergangswahrscheinlichkeiten modelliert, wobei jeweils von einer Zustandsart ausgegangen wird und der Übergang in die folgende Zustandsart betrachtet wird.

4.3.1 Neutraler Zustand

Der Übergang vom neutralen Zustand zu einer Haltestelle ist nicht abhängig von der aktuellen Zeit und somit immer möglich. Die Übergangswahrscheinlichkeit ist für jede Haltestelle gleich. Da auch der Verbleib im neutralen Zustand möglich ist und die Wahrscheinlichkeit aller ausgehenden Übergange 100 % ergeben muss, ist die Übergangswahrscheinlichkeit:

$$a_{NN} = a_{NH} = \frac{1}{\#H + 1}$$

Wie bereits erwähnt ist ein Übergang vom neutralen Zustand zu einer Linie nicht möglich, da davon ausgegangen wird, dass man nur an einer Haltestelle in ein öffentliches Verkehrsmittel einsteigen kann.

4.3.2 Haltestelle

Zeitlich gesehen ist ein Übergang von einer Haltestelle in den neutralen Zustand immer möglich. Die dazugehörige Übergangswahrscheinlichkeit ist indirekt jedoch trotzdem zeitabhängig, da ein Übergang in Linien möglich ist. Je nach Anzahl der Linien, in denen ein Übergang stattfinden kann, ändert sich auch die Übergangswahrscheinlichkeit in den neutralen Zustand.

In dem Fall, dass keine Linie an der Haltestelle hält, ist die Übergangswahrscheinlichkeit:

$$a_{HN} = a_{H_x H_x} = \frac{1}{2}$$

Anders ausgedrückt ist nur ein Übergang in den neutralen Zustand und der Verbleib im aktuellen Zustand bzw. der Haltestelle möglich. Die Wahrscheinlichkeiten werden gleichmäßig aufgeteilt.

Da an jeder Haltestelle auch Linien halten, müssen diese mit in Betracht gezogen werden. Dazu muss überprüft werden, welche Linien im Zeitraum der Beobachtung an der Haltestelle halten. Dafür wird ein Zeitfenster verwendet, das z.B. auf fünf Minuten vor und 10 Minuten nach dem planmäßigem Halten an der Haltestelle festgesetzt wird. Die Funktion l(H, t) steht im Folgenden für die Anzahl der Linien, die wie beschrieben um den Zeitpunkt t an der Haltestelle H halten. Dazu kommt, dass lediglich in eine Linie gewechselt werden kann, wenn die Linie selbst im Zeitraum an der Haltestelle hält. Dafür wird die Funktion l(H, L, t) verwendet, die wahr zurückgibt, falls an Haltestelle H die Linie L zum Zeitpunkt t hält.

Das Zeitfenster wird zu Testzwecken auf verschiedene Werte festgelegt. Da zwischen Ankunfts- und Abfahrtzeit an Haltestellen unterschieden wird, ist der Referenzwert für das Zeitfenster die Abfahrtszeit.

Die Übergangswahrscheinlichkeit ist für die Übergänge in den neutralen Zustand und beim Verbleib in der Haltestelle demnach:

$$a_{HN}(t) = a_{HH}(t) = \frac{1}{l(H,t)+2}$$

Für die Übergangswahrscheinlichkeit in eine Linie gilt:

$$a_{HL}(t) = \begin{Bmatrix} \frac{1}{l(H,t)+2} & falls\ l(H,L,t) \\ 0 & sonst \end{Bmatrix}$$

4.3.3 Linie

Für den Übergang von einer Linie in einen anderen Zustand gilt das gleiche, wie bei den Haltestellen. Der einzige Unterschied ist, dass ein Übergang in den neutralen Zustand nicht möglich ist. Analog ist ein Übergang von einer Linie in eine andere nicht möglich und die Referenzzeit für das Zeitfenster ist die Ankunftszeit des Haltens. Die Funktion h(L, t) steht im Folgenden für die Anzahl der Haltestellen, die um den Zeitpunkt t von der Linie L angefahren werden und die Funktion h(L, H, t) gibt an, ob die Linie L zum Zeitpunkt t an der Haltestelle H hält. Es gilt:

$$a_{LL}(t) = \frac{1}{h(L,t)+1}$$

$$a_{LH}(t) = \begin{Bmatrix} \frac{1}{h(L,t)+1} & falls\ h(L,H,t) \\ 0 & sonst \end{Bmatrix}$$

4.3.4 Übersicht

Tabelle 6 zeigt eine Übersicht der Übergangswahrscheinlichkeiten.

Tabelle 6 Übergangswahrscheinlichkeiten zwischen Zustandsarten

Von \ Nach	Linie	Haltestelle	Neutral
Linie	$\frac{1}{h(L,t)+1}$	$\begin{Bmatrix} \frac{1}{h(L,t)+1} & falls\ h(L,H,t) \\ 0 & sonst \end{Bmatrix}$	0
Haltestelle	$\begin{Bmatrix} \frac{1}{l(H,t)+2} & falls\ l(H,L,t) \\ 0 & sonst \end{Bmatrix}$	$\frac{1}{l(H,t)+2}$	
Neutral	0	$\frac{1}{\#H+1}$	

4.4 Beobachtungswahrscheinlichkeiten

Bei allen Zuständen wird die bereits beschriebene Wahrscheinlichkeitsdichte verwendet, um die Beobachtungswahrscheinlichkeit zu bestimmen, die für die SLD nicht zeitabhängig modelliert wird. Da allerdings z.B. eine Haltestelle nur durch eine GPS Koordinate definiert ist, wird ein größerer Bereich um die Haltestelle herum betrachtet, um mögliche Ungenauigkeiten von GPS Messungen auszugleichen. Der Flächeninhalt ist bei den Zustandsarten verschieden.

Bei Haltestellen wird eine kreisförmige Fläche um die Haltestelle als Wahrscheinlichkeitsdichte verwendet. Abbildung 7 zeigt die Haltestellen auf einer Buslinie, an denen der Bus laut Fahrplan hält. Die Haltestellen liegen demnach auf der Linie, so dass sie bei einer Fahrt beobachtet werden können. Berechnet wird die Beobachtungswahrscheinlichkeit der Haltestelle mit:

$$\beta_H(k) = \begin{Bmatrix} \frac{1}{\pi * r_k^2} & falls\ r(k,H) \\ 0 & sonst \end{Bmatrix}$$

wobei r(k,H) angibt, ob die Beobachtung k innerhalb des Kreises von Haltestelle H liegt.

Linien bestehen nicht aus einer einzigen GPS Koordinate, sondern aus einer längeren Reihe von diesen. Abbildung 8 zeigt die GPS Koordinaten, die eine mögliche Strecke einer Linie darstellen. In diesem Fall ist der Flächeninhalt nicht mit einem einfachen Kreis darstellbar, vielmehr ist es eine Fläche, die mit einem bestimmten Abstand entlang der Linie verläuft. Der Abstand zwischen den Koordinaten einer Linie ist beliebig und besitzt keinen Maximalabstand, der eingehalten werden muss. Dies ist für die Berechnung des Flächeninhaltes relevant, der in Abbildung 9 dargestellt ist und dessen genaue Berechnung in Kapitel 5.4 behandelt wird. Es gilt:

$$\beta_L(k) = \begin{Bmatrix} \frac{1}{A_L} & falls\ r(k,L) \\ 0 & sonst \end{Bmatrix}$$

wobei r(k,L) angibt, ob die Beobachtung k innerhalb der Fläche um die Linie L liegt.

Der neutrale Zustand umfasst die gesamte Fläche, die überhaupt beobachtbar ist. Dies wäre entweder die gesamte Oberfläche der Erde oder eine reduzierter Bereich in dem die Haltestellen und Linien liegen. Da der neutrale Zustand immer beobachtet werden kann, gilt:

$$\beta_N(k) = \frac{1}{A_N}$$

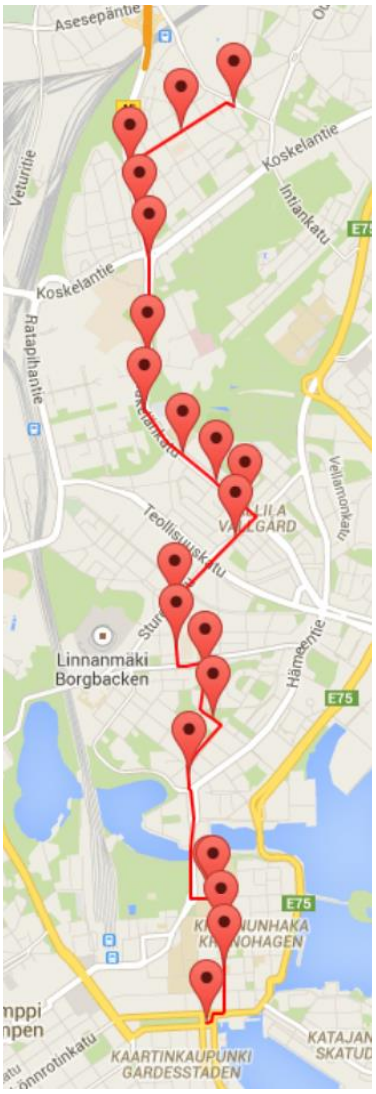

Abbildung 7 Haltestellen auf einer Linie

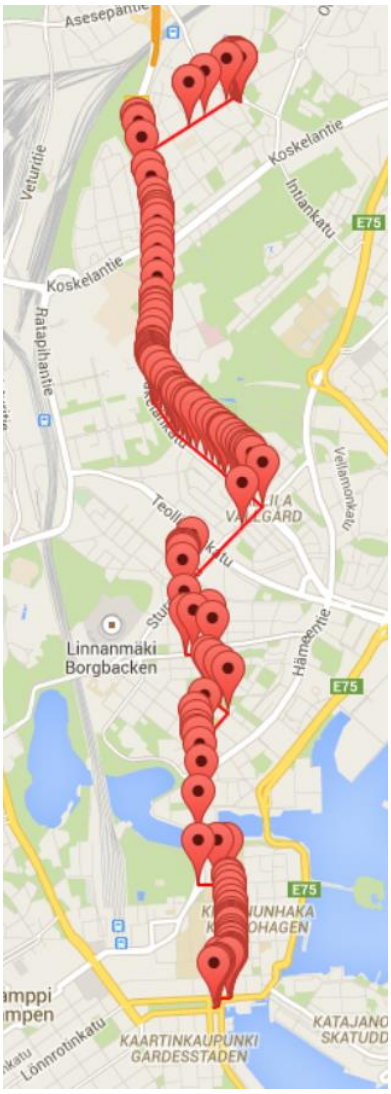

Abbildung 8 GPS Koordinaten eines Trips einer Linie

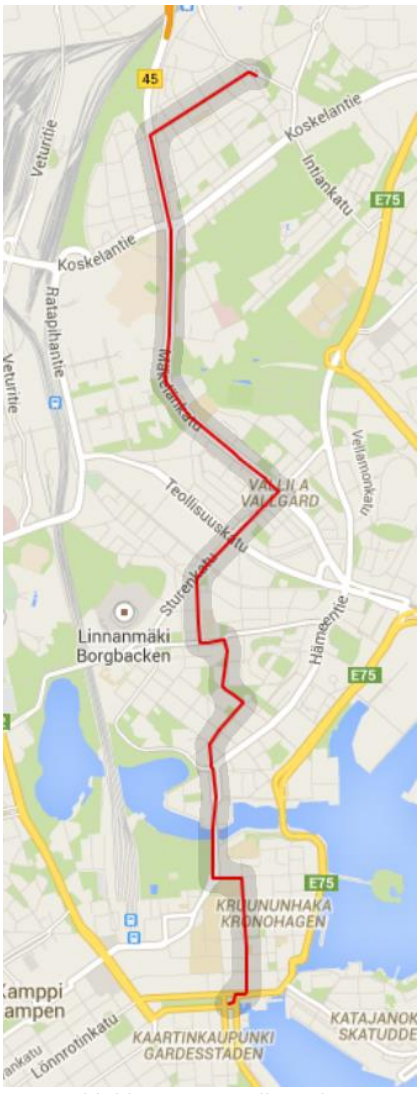

Abbildung 9 Darstellung des Flächeninhalts eines Trips

5 Umsetzung

Dieses Kapitel beschreibt die Umsetzung der SLD mit den in den vorherigen Kapiteln beschriebenen Aspekten. Bevor eine Beobachtungssequenz mit dem Viterbi-Algorithmus analysiert werden kann, müssen die für das HMM benötigten Daten importiert werden. Die Daten über Buslinienverbindungen usw. liegen im GTFS-Format vor, das in Kapitel 5.1 erläutert wird. Für das Einlesen der Daten in diesem Format wird ein Parser benötigt, der in Kapitel 5.2 beschrieben wird.

Da das klassische HMM den Anforderungen der SLD nicht erfüllt, beschreibt Kapitel 5.3 die notwendigen Anpassungen.

Nachdem alle Daten vorliegen müssen die Beobachtungswahrscheinlichkeiten berechnet werden, was in Kapitel 5.4 beschrieben wird.

5.1 GTFS

Die General Transit Feed Specification (GTFS) ist ein von Google entwickeltes einheitliches Format für die Speicherung von Daten über das öffentliche Transportsystem [5]. Dazu gehören Fahrplan- und geographische Daten über Linien und Haltestellen. Da dieses Format sämtliche Aspekte öffentlicher Verkehrsmittel abbildet, können alle benötigten Daten ausgelesen und verwendet werden. Die GTFS beinhaltet 13 verschiedene Inhaltstypen, von denen sieben für die SLD relevant sind und im Folgenden beschrieben werden.

5.1.1 Anbieter

Ein Anbieter ist ein Unternehmen, das Verbindungen des öffentlichen Personennahverkehrs anbietet. Die GTFS kann Daten von verschiedenen Unternehmen beinhalten, die daher mit einer eindeutigen Bezeichnung versehen werden, falls mehrere Anbieter vorhanden sind.

Da bei der SLD nicht ausgeschlossen werden kann, dass eine Person öffentliche Verkehrsmittel verschiedener Anbieter verwendet, werden in dieser Arbeit die Daten aller Anbieter betrachtet.

Beispiel:

agency_id	*agency_name*	*agency_url*	*agency_timezone*	*agency_lang*	*agency_phone*
HSL	Helsingin seudun liikenne	http://www.hsl.fi/	Europe/Helsinki	fi	(09)4766 4444

5.1.2 Haltestellen

Eine Haltestelle beinhaltet eine eindeutige Bezeichnung, einen Namen und die geographischen Daten in Form von Latitude und Longitude. Dazu kann eine Haltestelle als Station definiert sein, die verschiedene Haltestellen enthält.

Beispiel:

stop_id	*stop_code*	*stop_name*	*stop_desc*	*stop_lat*	*stop_lon*	zone_id
1010109	2016	Liisanpuistikko	KRH	60.174278	24.960387	1

stop_url	*location_type*	*parent_station*	*wheelchair_boarding*
http://aikataulut.hsl.fi/pysakit/fi/1010109.html	0		2

5.1.3 Linien

Eine Linie besteht aus einer Gruppe von Fahrten, die in Kapitel 5.1.4 beschrieben werden. Außerdem ist ein Linientyp enthalten, der zwischen den folgenden acht verschiedenen öffentlichen Verkehrsmitteln unterscheidet:

- Straßenbahn
- U-Bahn
- Eisenbahn
- Bus
- Fähre
- Kabelbahn
- Gondel
- Seilbahn

Für die SLD kann theoretisch jedes dieser Verkehrsmittel verwendet werden. Lediglich bei U-Bahnen ist der Empfang des Satelliten eingeschränkt und eine Aufnahme von Bewegungsdaten einer Person nicht möglich.

Beispiel:

route_id	*agency_id*	*route_short_name*	*route_long_name*
1001	HSL	1	Kauppatori - Käpylä

route_desc	*route_type*	*route_url*
		http://aikataulut.hsl.fi/linjat/fi/h1_1a.html

5.1.4 Fahrten

Eine Fahrt ist einer Linie zugeordnet und enthält einen Verweis auf einen Service (siehe Kapitel 5.1.5) und die Form der Fahrt (siehe Kapitel 5.1.6). Einer Fahrt sind zwei oder mehr Haltestellen zugeordnet.

Beispiel:

route_id	*service_id*	*trip_id*	*trip_headsign*
1001	1001_20140825_20141030_Ke	1001_20140825_Ke_1_0953	Käpylä

direction_id	*shape_id*	*wheelchair_accessible*	*bikes_allowed*
0	1001_20140811_1	1	2

5.1.5 Services

Ein Service ist eine Ansammlung an Kalenderdaten in einem bestimmten Zeitraum, der über ein Start- und Enddatum definiert wird. Die Daten bestimmen, an welchen Tagen der Service verfügbar ist, wobei dies nach einem wöchentlichen Zeitplan geschieht, bei dem die Verfügbarkeit des Service von Montag bis Sonntag festgelegt ist.

Dazu können Ausnahmen vorliegen, die für einen bestimmten Tag den Service verfügbar oder nicht verfügbar macht. Auf diese Weise können z.B. außerplanmäßige Busverbindungen angeboten oder vom planmäßigen Fahrplan gestrichen werden.

Beispiel:

service_id	*monday*	*tuesday*	*wednesday*	*thursday*
1001_20140825_20141030_Ke	0	0	1	0

friday	*saturday*	*sunday*	*start_date*	*end_date*
0	0	0	20140825	20141030

In diesem Beispiel wird der Service zwischen dem 25.08.2014 und 30.10.2014 nur mittwochs angeboten.

5.1.6 Formen

Eine Form definiert die Fahrt mit einem Segment von GPS Koordinaten, die mit Latitude und Longitude vorliegen. Zu jeder Koordinate wird eine Sequenznummer gespeichert, welche die Reihenfolge festlegt. Somit kann eine Fahrt mit GPS Koordinaten nachvollzogen werden.

Beispiel:

shape_id	*shape_pt_lat*	*shape_pt_lon*	*shape_pt_sequence*
1001_20140811_1	60.167430	24.951684	1

5.1.7 Haltezeit

Für jede Fahrt ist für jede Haltestelle die Zeit hinterlegt, zu welcher z.B. der Bus planmäßig dort hält.

Beispiel:

trip_id	*arrival_time*	*departure_time*	*stop_id*
1001_20140825_Ke_1_0953	09:53:00	09:53:00	1030423

stop_sequence	*stop_headsign*	*pickup_type*	*drop_off_type*	*shape_dist_traveled*
1		0	0	0.0000

5.2 Onebusaway

Onebusaway ist ein Projekt, das aktuell an der Universität von Südflorida und der Georgia Tech entwickelt wird. Das Projekt wird für das Importieren der GTFS Daten verwendet. Diese Daten liegen in verschiedenen Textdateien, die jeweils in eine eigene JAVA-Klasse importiert werden.

Das Projekt wird in die eigene Struktur integriert und beinhaltet bereits Funktionen, die für den Datenzugriff relevant sind. Allerdings wird das Projekt an verschiedenen Stellen angepasst, um die Daten für den Viterbi-Algorithmus bereitzustellen.

Um die relevanten Verbindungen zwischen den Klassen zu zeigen, beinhaltet das in Abbildung 10 gezeigte UML Diagramm eine sehr rudimentäre Darstellung vom unveränderten Onebusaway. In der Klasse *StopTime* wird die Referenz auf den dazugehörigen Trip und die Haltestelle gespeichert. Weder die Haltestelle noch der Trip enthält Informationen über Haltezeiten.

Um die GPS Koordinaten zu einem Trip zu erhalten, wird die Klasse *GtfsRelationalDao* benötigt, da die Daten nicht in den Trips direkt gespeichert werden. Ähnlich verhält es sich mit Daten bezüglich der Haltezeiten von einer Haltestelle, die man über diese Klasse ermitteln kann.

Um zu ermitteln, ob ein eine Haltezeit am Tag der Beobachtung gültig war, wird die Klasse *CalendarService* verwendet, welche die erläuterten Kalenderdaten mit vorhanden Ausnahmen berücksichtigt.

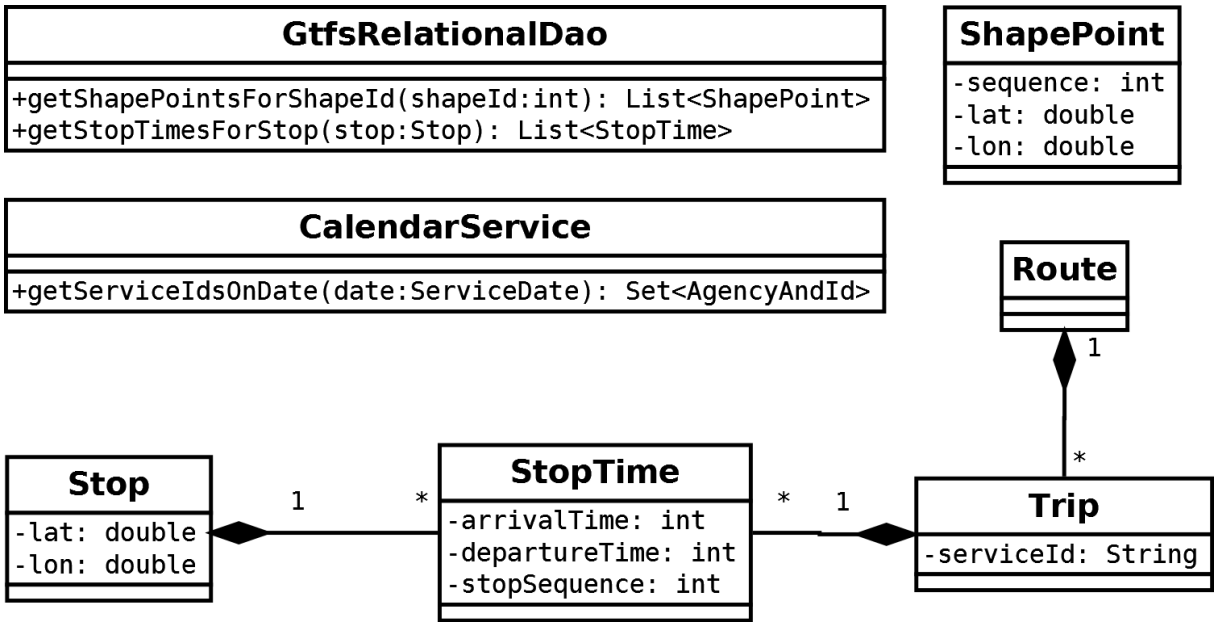

Abbildung 10 UML Diagramm Ausschnitt von Onebusaway

5.3 Anpassung an SLD

Im Modellierungskapitel wurden die drei Zustandsarten Neutral, Haltestelle und Linie beschrieben. Daher muss die beschriebene Struktur von Onebusaway verändert werden. Die abstrakte Klasse *State* wird als Oberklasse für die Zustandsarten verwendet. Der neutrale Zustand ist eine komplett neue Klasse, die hinzugefügt wird. Der Zustand Haltestelle wird durch die Klasse *Stop* und der

Zustand Linie durch die Klasse *Route* repräsentiert. Die Modellierung sah vor, dass eine Haltezeit einer Linie zugeordnet wird. Durch den Aufbau der GTFS wird eine Linie allerdings in mehrere Trips unterteilt. Da ein Trip verschiedenen Strecken folgen kann, müssen auch mehrere Strecken bei der Berechnung der Beobachtungswahrscheinlichkeit berücksichtigt werden und eine Linie deckt womöglich eine größere Fläche von mehreren Trips ab. Bei Übergangswahrscheinlichkeiten wird ein Übergang für die Linie mit dem Übergang in einen Trip gleichgesetzt.

Abbildung 11 zeigt das UML Diagramm dieser abgeänderten Variante. Für den direkten Zugriff auf die relevanten Daten werden in den Haltestellen die Haltezeiten nach Linien gruppiert gespeichert. Das ist notwendig für die Übergangswahrscheinlichkeiten zwischen Haltestellen und Linien, da überprüft wird, ob eine bestimmte Linie zu einer bestimmten Zeit an der Haltestelle hält.

Analog dazu werden in Linien die Haltezeiten nach Haltestellen gruppiert sortiert, um in diesem Fall die Übergangswahrscheinlichkeiten zwischen Linie zurück zur Haltestelle zu ermitteln.

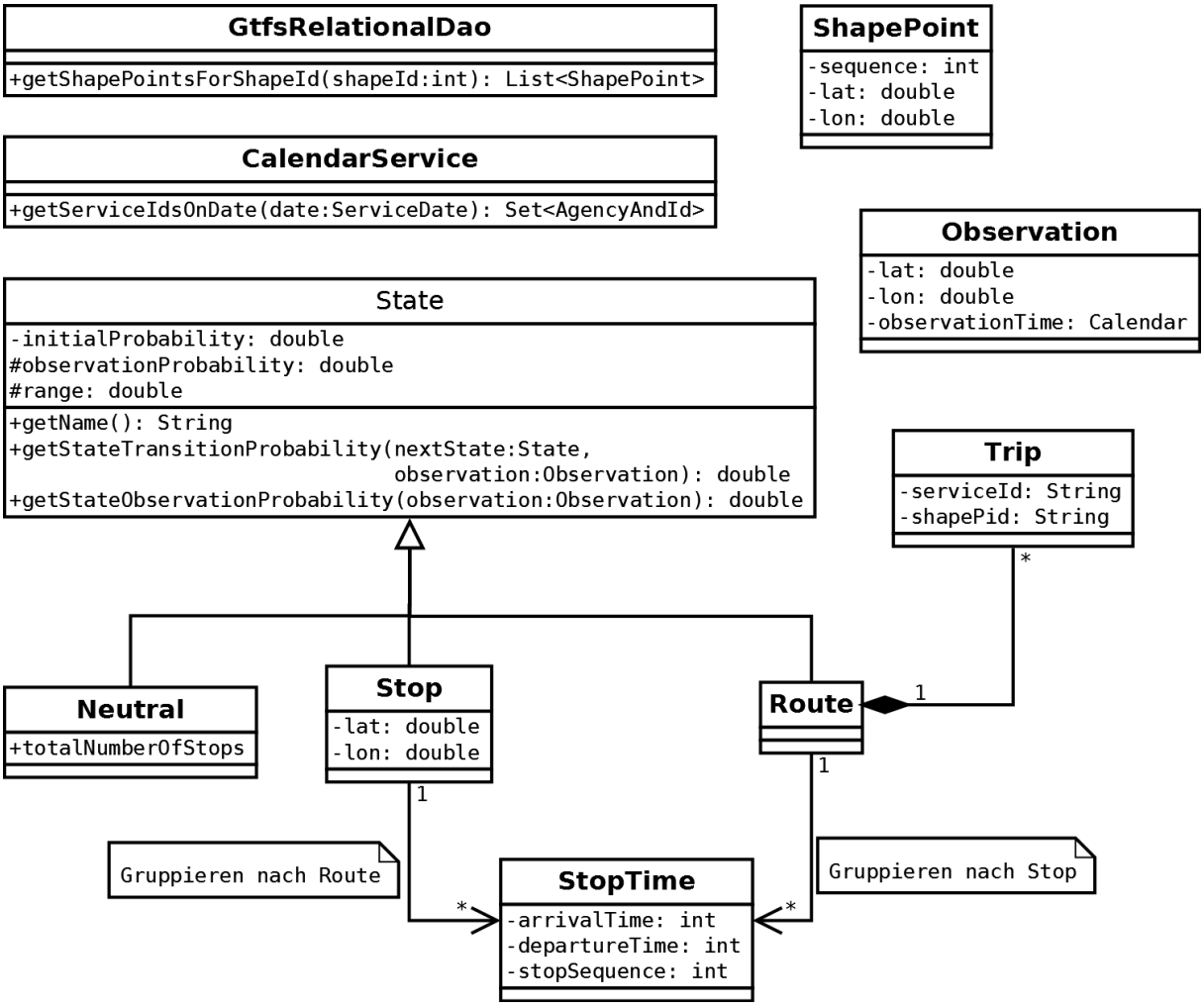

Abbildung 11 Onebusaway für HMM

5.4 Berechnung der Beobachtungswahrscheinlichkeit

Um den Flächeninhalt einer Linie zu berechnen, werden die Formeln aus Kapitel 2.2 benötigt, welche die Krümmung der Erdoberfläche mit einbeziehen. Da die Berechnung des Flächeninhalts um einen

Polygonzug herum aufgrund von möglichen Überschneidungen kompliziert ist, wird der Flächeninhalt approximiert. Dafür wird die Entfernung von einzelnen GPS Koordinaten zu der Linie berechnet. Falls der berechnete Wert einen bestimmten Abstand zur Linie hat, gehört dieser Punkt zum Flächeninhalt. Je dichter das Koordinatennetz ist, desto höher ist die Wahrscheinlichkeit eines genauen Flächeninhaltes, wie Abbildung 12 verdeutlichen soll.

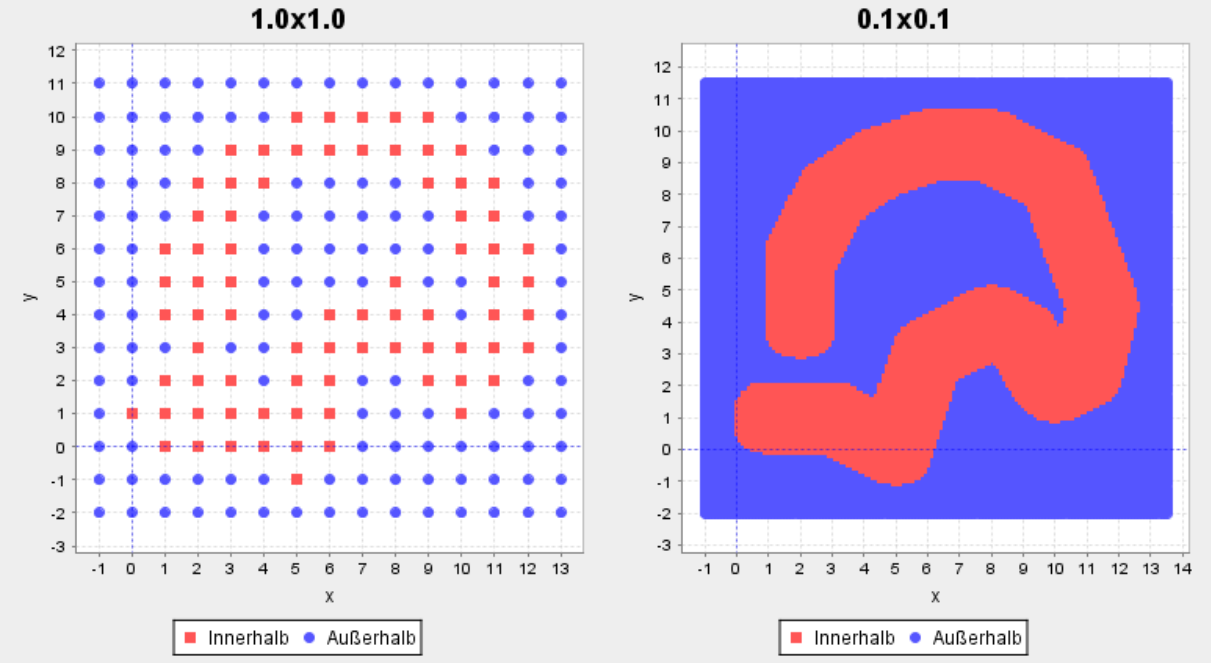

Abbildung 12 Approximation des Flächeninhaltes eines Polygonzuges

Der Bereich, der für eine Linie untersucht werden muss, wird über die Punkte des Polygonzuges ermittelt. Die am weitesten außen liegenden Koordinaten bilden am Ende ein Rechteck, das die Grenzen darstellt. Lediglich in diesem Bereich muss über die Koordinaten iteriert werden, der in Abbildung 12 blau markiert ist.

Wichtig dabei ist, dass die Distanz zum Polygonzug bestimmt wird und nicht nur die Distanz zu einzelnen Koordinaten des Polygonzuges. Ob eine Koordinate zum Flächeninhalt gehört, bestimmt sich durch den Abstand zur Linie, der vorher festgelegt wird. Dafür wird eine Kombination aus der Cross-Track Distanz und Along-Track Distanz verwendet, wie sie in Kapitel 2.2.3 bzw. Kapitel 2.2.4 erläutert worden sind. Dabei wird zunächst die Distanz zum Großkreis berechnet, der durch zwei aufeinanderfolgende Koordinaten des Polygonzuges führt. Um dies zu verdeutlichen, zeigt Abbildung 13 den Zusammenhang einiger Beispiel- Koordinaten.

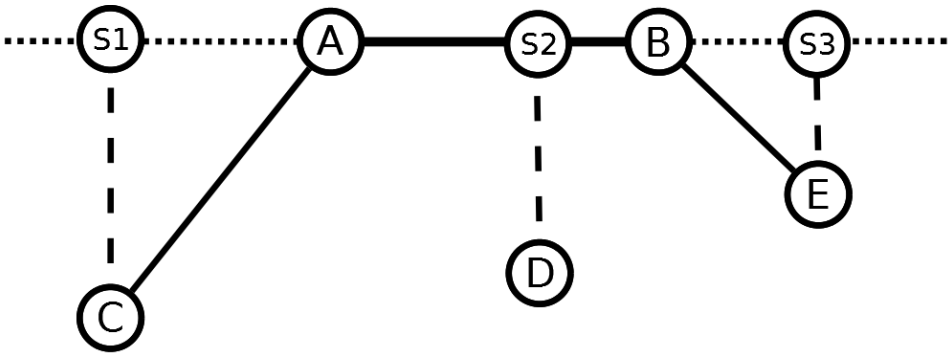

Abbildung 13 Darstellung des Abstandes dreier Koordinaten zum Großkreis

Die Koordinaten C, D und E stammen von einer Beobachtung deren Distanz zur Linie von A nach B ermittelt werden soll. S1, S2 und S3 stellen die Koordinaten dar, zu denen die Cross-Track Distanz berechnet wird. Das Problem bei der Verwendung dieser Formel wird deutlich, wenn man die Koordinate C betrachtet. Die gesuchte Distanz ist die von C nach A, da lediglich die Distanz zu der Linie zwischen A und B gesucht wird. Da die Cross-Track Distanz die Distanz zu dem Großkreis, der durch diese beiden Koordinaten führt, ermittelt, wird die Distanz zu S1 errechnet. Um zu überprüfen, ob die Distanz zu einer Koordinate zwischen A und B berechnet worden ist, wird die Along-Track Distanz verwendet. Wie bereits beschrieben steht sie für die Distanz zwischen der Koordinate A und der Koordinate, welche die kürzeste Distanz zwischen der Koordinate C und dem Großkreis, der durch A und B führt, aufweist. Dies ist S1 in Abbildung 13. Wurde der Großkreis von A nach B berechnet, so ist die Along-Track Distanz zu S1 negativ. Falls dies der Fall ist, wird die direkte Distanz zwischen C und A ermittelt, da dies die kürzeste Distanz zwischen C und der Linie durch A und B ist.

Für die Cross-Track Distanz der Koordinaten D und E liegt eine positive Along-Track Distanz vor. Um zu ermitteln, ob die korrekte Distanz ermittelt worden ist, wird die Along-Track Distanz des jeweiligen Punktes mit der Distanz von A nach B verglichen. Für die Koordinate D ist die Along-Track Distanz kleiner als dieser Abstand, so dass die Distanz zur Linie von A nach B die Cross-Track Distanz ist. Die Along-Track Distanz für die Koordinate E ist höher, also wird die einfache Distanz zwischen E und B ermittelt.

Dieses Verfahren muss für alle Koordinatenpaare des Polygonzuges als A und B durchgeführt werden.

6 Evaluation

Dieses Kapitel beschreibt die Evaluation der vorgestellten Implementierung und geht dabei auf die verwendeten Testdaten ein und beschreibt die zwei durchgeführten Testphasen.

6.1 Datenlage

6.1.1 Testdaten

Für das Testen des beschriebenen Algorithmus werden aufgezeichnete Daten aus Helsinki verwendet. Insgesamt liegen 398 verschiedene Trips von 61 verschiedenen anonymen Personen vor. Demographische Daten sind nicht vorhanden.

Um die Ergebnisse des in dieser Arbeit entwickelten Algorithmus zu überprüfen, werden Informationen über die tatsächlich benutzten Fortbewegungsmittel benötigt. Da dies nicht für jeden Trip der Fall ist, grenzt sich der Testdatensatz mit der höchsten Priorität weiter ein. Zunächst werden elf Trips mit Bussen untersucht, die in Tabelle 7 aufgelistet sind.

Neben den von den Testpersonen gemachten Annotationen zur tatsächlichen Linie werden auch die Ergebnisse von Live+Gov verwendet. Da nicht von jedem Trip das Ergebnis vorliegt, wird lediglich in der zweiten Testphase der Vergleich mit Live+Gov durchgeführt. Dies sind insgesamt zehn Trips.

Die Testdaten wurden zwischen dem 30. September und 15. Oktober 2014 aufgezeichnet, so dass ein GTFS Datensatz verwendet, der zu diesem Zeitpunkt gültig war. Dieser wird im folgenden Kapitel beschrieben.

Tabelle 7 Auflistung der ersten Test-Trips

Trip Id	Linie	Anzahl Beobachtungen
8	1070T	137
11	1073	250
16	1004	138
17	1024	472
21	2315	333
23	4741	472
26	5530	281
36	1070T	310
50	2150A	263
51	2270	89
52	2550	247

6.1.2 GTFS Daten

Es ist nur ein einziger Anbieter vorhanden, die „Helsingin seudun liikenne" (HSL), die 2010 gegründet worden ist und im Großraum Helsinkis Verkehrsverbindungen anbietet.

Insgesamt sind 580 Linien vorhanden und vier der acht vorhandenen Verkehrsmittelarten werden angeboten:

- Straßenbahnlinien: 13 (2,2 %)
- U-Bahnlinien: 3 (0,5 %)
- Buslinien: 548 (94,5 %)
- Fähren: 2 (0,3 %)

14 weitere Linien sind mit einer in der in der Dokumentation nicht hinterlegten Verkehrsmittelart hinterlegt. Diese machen die restlichen 2,5 % aus.

Insgesamt sind 177.817 Fahrten zu den 580 Linien hinterlegt, was im Schnitt ca. 360 Fahrten pro Linie entspricht. Dazu kommen 7570 Haltestellen und 86 Stationen.

6.2 Testphase I

Die erste Testphase verwendet das HMM, wie es beschrieben worden ist. Der Abstand zu Haltestellen bezüglich der Beobachtungswahrscheinlichkeit von Koordinaten beträgt in dieser Testphase 50 Meter, bei Linien 100 Meter.

Ein Übergang zwischen Haltestelle und Linie und andersrum ist dann möglich, wenn der Zeitpunkt der Beobachtung fünf Minuten vor oder zehn Minuten nach der Ankunfts- bzw. Abfahrtzeit liegt.

Schon bei den ersten Tests zeigte sich, dass die Wahrscheinlichkeiten bei über 200 Beobachtungen in einem zu niedrigen Bereich lagen, so dass der Algorithmus abbrach. Daher wurde die Beobachtungssequenz in kleinere Sequenzen aufgeteilt.

Die Ergebnisse der ersten Testphase zeigen, dass eine Erkennung von Linien nach dem in dieser Arbeit vorgestellten Verfahren möglich ist, aber noch verschiedene Schwachstellen und dadurch Fehlerkennungen aufweist.

Zunächst ein Beispiel einer Zustandssequenz, um die Struktur zu beschreiben. Tabelle 8 zeigt einen Auszug der Ergebnis-Zustandssequenz von Trip 11. Gleiche Zustände, die direkt aufeinander folgten, wurden für eine bessere Übersicht entfernt. So wurde z.B. Linie 2554_554 von Beobachtung 30 bis 35 sechs Mal erkannt. Eintrag Nummer 50 ist dabei das Ende eines untersuchten Abschnittes. An diesem Beispiel ist erkennbar, dass durch das Zerschneiden der Beobachtungssequenz die Linie abgebrochen worden ist. Diese Fälle werden im Folgenden nicht beachtet und die Erkennung dieser Linien außen vor gelassen, da eine nächste Haltestelle nicht erreicht wurde.

Tabelle 8 Auszug der Ergebnis-Zustandssequenz von Trip 11

#	Uhrzeit	Zustand
1	07:51:43	Neutral
2	07:51:49	Stop 1381121_Malmin tori
5	07:52:06	Route 1077A_77A
7	07:52:17	Stop 1382126_Malmin asema tulo
14	07:52:57	Neutral
15	07:53:03	Stop 1381101_Malmin asema(term.)
18	07:53:19	Route 4519_519
21	07:53:35	Stop 1382123_Ala-Malmi
22	07:53:41	Route 2554_554
23	07:53:46	Stop 1382124_Ala-Malmi
30	07:54:23	Route 2554_554
36	07:54:56	Stop 1382116_Riihenkulma
43	07:55:35	Route 2554_554
50	07:56:15	Route 2554_554
1	07:56:20	Neutral
2	07:56:25	Stop 1382128_Latokartanontie
3	07:56:31	Stop 1382128_Latokartanontie

Die gezeigte Zustandssequenz ist so zu interpretieren, dass der Benutzer zu Fuß zur Haltestelle Malmin tori gegangen ist, sich 17 Sekunden nach dem Aufenthalt dort bereits in Linie 1077A befunden und wiederum elf Sekunden später die nächste Haltestelle Malmin asema tulo erreicht hat. Daraufhin ist er zu Fuß zu der Haltestelle Malmin asema(term.) gegangen, die sich im gleichen Terminal wie die Haltestelle davor befindet. Von dort aus erreichte er innerhalb von 32 Sekunden mit Linie 4519 die Haltestelle Ala-Malmi.

Dieser Ablauf ist in der Praxis nahezu ausgeschlossen und stimmt auch nicht mit der vom Benutzer gemachten Anmerkung überein, laut der er die gesamte Zeit mit Linie 1073 unterwegs war. Somit ist keine Linie in dieser Zustandssequenz korrekt. Dennoch lassen sich verschiedene Eigenschaften herauslesen.

Das auffälligste an der Zustandssequenz ist, dass der Algorithmus immer eine Linie verlässt, wenn es an einer Haltestelle möglich ist. Somit wird immer wieder aufs Neue lediglich die Strecke zwischen zwei Haltestellen untersucht. Dies ist auch bei allen anderen untersuchten Linien der Fall. Die Begründung dafür ist, dass die Beobachtungswahrscheinlichkeiten von Haltestellen wesentlich höher sind als die von Linien. Da die dazugehörige Übergangswahrscheinlichkeit von Linie zu Linie bzw. Linie zu Haltestelle stets gleich ist, hängt die Wahrscheinlichkeit für den Zustand insgesamt nur von der Beobachtungswahrscheinlichkeit und dem Vorgänger ab.

Im dem Fall, dass trotz des Wechsels zur Haltestelle immer wieder der Übergang in die korrekte Linie erfolgen würde, könnte man die Zustandssequenz einfach so interpretieren, dass die Person im Bus sitzen geblieben ist. Allerdings wird nur in wenigen Fällen öfters als zwei Mal die korrekte Linie erkannt. Das Problem dabei ist wieder die Beobachtungswahrscheinlichkeit, aber in diesem Fall die der Linien, die sich immer unterscheidet. Dadurch werden Linien priorisiert, deren Trip-Strecken die

geringste Fläche einnehmen, da eine kleinere Fläche zu einer höheren Beobachtungswahrscheinlichkeit führt.

Ein weiterer kleinerer Fehler zeigt sich in den Zuständen 21-23, bei denen ein Wechsel von Haltestelle Ala-Malmi in Linie 2554 und wieder zurück in Haltestelle Ala-Malmi stattfindet. Auch diese Zustandsabfolge tritt häufiger auf und liegt daran, dass sich die Person innerhalb während dieser drei Zustände zunächst innerhalb des Radius der Haltestelle befand, dann in einer Beobachtung außerhalb war und schließlich wieder innerhalb.

Da eine quantitative Präsentation der Ergebnisse aufgrund der sehr fehlerhaften Erkennungsrate wenig sinnvoll ist, werden direkt die zweite Testphase und die damit durchgeführten Änderungen beschrieben.

6.3 Testphase II

Der dauerhafte Zustandswechsel zurück in eine Haltestelle führt dazu, dass immer wieder nur einzelne Abschnitte zwischen zwei Haltestellen untersucht werden. Die resultierende Zustandssequenz sollte daher eine längere Reihe von Zuständen aufweisen, die zu einer einzigen Linie gehören. Das Problem dabei ist, dass eine Untersuchung längerer Beobachtungssequenzen nicht möglich war, weil die Wahrscheinlichkeiten zu niedrig waren. Daher werden die Wahrscheinlichkeiten manuell gesetzt und damit die beschriebenen Bedingungen der Wahrscheinlichkeiten verletzt. Da die absoluten Wahrscheinlichkeiten allerdings keine wesentliche Rolle für die Funktionalität des Algorithmus spielen, ist dies nicht weiter problematisch.

Um den ständigen Wechsel in eine Haltestelle zu verhindern und möglichst lange auf einer Linie zu bleiben, wird die Übergangswahrscheinlichkeit von einer Linie zu einer Haltestelle und umgekehrt niedrig angesetzt. Das soll zur Folge haben, dass Zustandsfolgen möglichst wenige Zustandswechsel haben und somit die längste mögliche Folge einer Linie am wahrscheinlichsten wird. Der Zeitraum, an dem Übergänge möglich sind, bleibt bei fünf Minuten vor und zehn Minuten nach der Ankunfts- bzw. Abfahrtzeit. Tabelle 9 zeigt die neuen Übergangswahrscheinlichkeiten.

Tabelle 9 Übergangswahrscheinlichkeiten der zweiten Testphase

Von \ Nach	Linie	Haltestelle	Neutral
Linie	1 wenn gleiche Linie 0 sonst	0,1 wenn hält 0 sonst	0
Haltestelle	0,1 wenn hält 0 sonst	1 wenn gleich Haltestelle 0 sonst	1
Neutral	0	1	0,9

Um zu gewährleisten, dass der Algorithmus auch längere Beobachtungssequenzen verwenden kann, werden die Beobachtungswahrscheinlichkeiten manuell gesetzt. Tabelle 10 zeigt die manuell gesetzten Beobachtungswahrscheinlichkeiten.

Tabelle 10 Beobachtungswahrscheinlichkeiten der zweiten Testphase

Neutral	Haltestelle	Linie
0,1	1.01	1

Der neutrale Zustand soll nicht wahrscheinlicher sein als eine Sequenz aus Linien, die erst mit einer geringen Übergangswahrscheinlichkeit auftreten. Daher ist die Beobachtungswahrscheinlichkeit einer Linie deutlich höher als die des neutralen Zustandes. Eine Haltestelle soll minimal wahrscheinlicher sein, damit der Übergang in eine Linie so spät wie möglich erfolgt, da dies erst die tatsächliche Abfahrtszeit ist.

Eine Zustandssequenz endete oft in einer Linie. Da dadurch der Ausstieg an einer Haltestelle nicht berücksichtigt wird, muss für die letzte Beobachtung definitiv der neutrale Zustand eintreten. Um das zu gewährleisten, wird an das Ende jeder Beobachtungssequenz eine Beobachtung hinzugefügt, die diesen Effekt herbeiführt.

Die Umstellungen gegenüber der ersten Testphase erbrachten die beschriebenen Effekte. Tabelle 11 auf Seite 42 zeigt die Ergebnisse der zweiten Testphase. Auch wenn eine genauere Erkennung der Linien erfolgt, bleiben diverse Probleme bestehen. Trip 11 zeigt dabei gleich zwei dieser Probleme auf. Zunächst werden zwei verschiedene Linien erkannt, wobei Linie 1070T die Hauptlinie über mehrere Haltestellen ist und Linie 1055 lediglich über eine weitere Haltestelle verläuft. Die Fehlerkennung der Linie 1073 liegt in der Strecke, die über den Bereich der Beobachtung zu 100 % identisch ist. Abbildung 14 zeigt die Problematik auf der Karte.

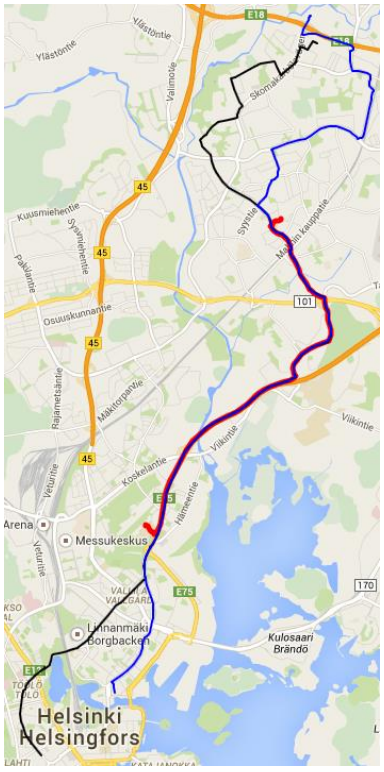

Abbildung 14 Darstellung von Trip 11, Linie 1070T und Linie 1073

Tabelle 11 Ergebnisse der zweiten Testphase

Trip Id	Erwartete Zustandssequenz	Ergebnis	Bemerkung
8	Sumatrantie 1070T Malmin tori	Wie erwartete Zutandssequenz	
11	Malmin sairaala 1073 Kumpulan kampus	Malmin sairaala 1070T Kumpulan kampus 1055 A.I. Virtasen aukio	Geht nach Verlassen der ersten Linie zu Fuß zu einer weiteren Haltestelle
16	Lasipalatsi 1004 Aleksanterinkatu	Wie erwartete Zutandssequenz	
17	Lasipalatsi 1024 Rajasaarentie	Wie erwartete Zutandssequenz	
21	Rajaportti 2315 Ruosilanpolku	Wie erwartete Zutandssequenz	
23	Rautatientori 4741 Kanneltie	Rautatientori 4741 Sumatrantie 1070T Viikki Neutral Louhikkotie 4741 Kanneltie	Auf halber Strecke wird keine Linie mehr erkannt, obwohl die Beobachtungen der Strecke genau folgen
26	Lastutie 5530 Lehtimäki	Keine Linie erkannt	Zu starke Schwankungen um die Linie herum
36	Kumpulan kampus 1070T Malmin tori	Wie erwartete Zutandssequenz	
50	Länsiväylä 2150A Alakaupunki	Kampintori 1065A Länsiväylä 2158 Karhusaarensolmu 2150 Alakaupunki	Der Anfang der GPS Koordinaten ist etwas ungenau. Bis zur Haltestelle Länsiväylä sollte daher Neutral erkannt werden. Danach ist die korrekte Linie unwahrscheinlicher, da die Einstiegsbushaltestelle nach einer Sequenz aus neutralen Zuständen eine sehr geringe Wahrscheinlichkeit hat. Linie 2158 fährt aber von der Ausstiegshaltestelle von Linie 1065A ab und ist damit wahrscheinlicher. Linie 2150 wird erst nach Ende von Linie 2158 erkannt und besitzt in diesem Bereich die gleiche Strecke wie Linie 2510A.
51	Pitäjänmäen asema 2270 Leppävaaran keskus	Wie erwartete Zutandssequenz	
52	Leppävaaran asema 2550 Takomotie	Wie erwartete Zutandssequenz	

Eine genauere Betrachtung der Zwischenschritte des Viterbi-Algorithmus zeigt, dass die korrekte Linie ebenfalls in den betrachteten Zuständen war. Tabelle 12 zeigt die Wahrscheinlichkeiten, die einen Schritt vor dem Übergang in den Zustand der Haltestelle vorliegen. Die höchsten Wahrscheinlichkeiten weisen die Linien 1070T und 1073 auf. Der Grund für die Erkennung von Linie 1070T liegt lediglich an der Reihenfolge, in der die Zustände in die Liste geladen worden sind.

Tabelle 12 Zwischenschritt 208 des Viterbi-Algorithmus für Trip 11

Route 1055_55	1.305994514823038E-7
Route 1070T_70T	1.305994514823038E-5
Route 4740_740	1.305994514823038E-7
Route 4730_730	1.305994514823038E-7
Route 4731_731	1.305994514823038E-7
Route 4732_732	1.305994514823038E-7
Route 4734_734	1.305994514823038E-7
Route 4741_741	1.305994514823038E-7
Route 4742_742	1.305994514823038E-7
Route 1077_77	1.305994514823038E-7
Route 1075_75	1.305994514823038E-7
Route 1073_73	1.305994514823038E-5
Route 1056_56	1.305994514823038E-7
Route 9788_788	1.305994514823038E-7
Route 4741K_741K	1.305994514823038E-7
Route 9738_738	1.305994514823038E-7
Neutral	6.52997257411519E-9

Eine Unterscheidung dieser Linien ist nur über die Haltezeiten der Busse möglich. Die ungefähre Uhrzeit, an dem die Person an der Haltestelle stand, war 7:48:35 Uhr. Die Abfahrtszeit für Linie 1070T an dem Tag war 7:39:00 Uhr, also nur knapp in dem Zeitraum der Abfahrt. Die Abfahrtszeit von Linie 1073 hingegen war um 7:48:00 Uhr, also kam der Bus wahrscheinlich sehr pünktlich. Die Zeitdifferenz zur eigentlichen Abfahrtzeit könnte man also bei der Wahrscheinlichkeit berücksichtigen.

Die Abfahrtzeiten der Busse sind dann problematisch, wenn der Bus Verspätung hatte. Denkbar ist eine Änderung, bei der sämtliche Zeiten der Beobachtung gültig sind, aber die Wahrscheinlichkeit rapide sinkt, wenn die Zeiten sehr weit auseinander liegen.

Um diese Methode mit der naiven Methode von Live+Gov zu vergleichen, werden weitere zehn Trips getestet, bei denen sowohl deren Ergebnisse vorliegen als auch in der Benutzerannotation die korrekte Linie hinterlegt ist. Die Ergebnisse zeigen, dass beide Methoden ähnlich gute Ergebnisse erzielen. Tabelle 13 zeigt die Ergebnisse in der Übersicht. Beide Verfahren haben korrekte und fehlerhafte Erkennungen. Die Gründe liegen dabei vor allem an zu ungenauen Messungen der Position, aber auch an der Implementierung mit dem Zeitpunkt des Einsteigens.

Tabelle 13 Vergleich der Ergebnisse dieser Arbeit mit Live+Gov

Trip Id	Erwartete Zustandssequenz	Ergebnis dieser Arbeit	Ergebnis Live+Gov	Bemerkung
66	Hattulantie 1065A Haapaniemi	Rautalammintie 1001A Hattulantie Neutral Sörnäinen(M) 4650 Haapaniemi	1066A	Sprunghafte GPS Koordinaten bei der Beobachtung. Übergangswahrscheinlichkeit für die korrekte Linie bei 0, weil außerhalb des Zeitraums.
98	Von Daehnin katu 1068 Latokartano 2550 Oulunkylän asema Neutral Oulunkylän asema 3001I Tikkurilan asema	Von Daehnin katu 1068 Viikin tiedepuisto Neutral Pukinmäki 3001I_I Puistola	1068	Linie 1068 wird zu lange erkannt, da die darauffolgende Linie 2550 gar nicht erkannt wird. In diesem Bereich sind die GPS Koordinaten zu ungenau. Aus demselben Grund wird auch Linie 3001I lediglich auf einer kürzeren Strecke erkannt. Die Einstiegshaltestelle ist eine spätere und die Ausstiegshaltestelle eine frühere.
107	Haagan paloasema 1041 Kauppakorkeakoulut	Wie erwartete Zutandssequenz	1041	
114	Niemenmäki 1039 Munkkiniemen aukio	Munkkivuori 2552 Otaniemenristi	1014 2552	Die Beobachtungssequenz läuft entlang der gesamten Linie 2552. Die Person nahm wahrscheinlich zwei verschiedene Linien für diese Strecke, wobei Linie 1039 die erste davon war.
132	Hakaniemen metroasema 1300M Kampin metroasema	Keine Linie erkannt	1300M	Die Linie 1300M ist die einer U-Bahn. Es konnten während der Fahrt keinerlei GPS Signale empfangen werden, so dass Einstieg und Ausstieg jeweils eine Koordinate darstellen, die in der Sequenz zu einer gerade Strecke durch Helsinki führten. Daher konnte keine Strecke erkannt werden.
140	2550 1057	Itäkeskus(M) 2550 Viikin tiedepuisto 1057 Valtimontie Neutral 1057 Korppaantie	2550 1057	
159	1024	Kolmikulma 1010	1006	Die falsche Linie wird erkannt, weil lediglich eine Beobachtung

		Ylioppilastalo		im Bereich der ersten Haltestelle liegt. Der erste Zustand ist aber immer Neutral. Daher kann kein Übergang von Haltestelle zu Linie erfolgen.
177	Pasilansilta 1023 Kulttuuritalo	Wie erwartete Zutandssequenz	1070T	
207	Lapinrinne 2150A	Kamppi 2102T Lapinrinne 2112 Länsiväylä 2143A Matinsolmu 2150A Alakaupunki	2150A	Die ersten aufgezeichneten GPS Koordinaten liegen etwas weiter entfernt vom eigentlichen Einstieg in die Haltestelle, so dass von Messungsfehlern auszugehen ist. Dass im Anschluss nicht direkt Linie 2150A erkannt wird, liegt daran, dass die Einstiegshaltestelle von Linie 2112 die Ausstiegshaltestelle von Linie 2102T ist.
228	Eiran sairaala 1018	Eiran sairaala 1018 Viiskulma 1014 Haartmaninkatu	1014	Linie 1018 wird anfangs korrekt erkannt, weil es die einzige Linie ist, die an der Haltestelle abfährt. Da zwischen zwei Haltestellen die Wahrscheinlichkeit für Linie 1018 null wird, ist Linie 1014 wahrscheinlicher, da diese Linie weiter beobachtet wird.

7 Fazit

Hidden Markov Modelle eigenen sich für die Service Line Detection, wenn auch in abgewandelter Form. Das Verwenden von Wahrscheinlichkeiten, die den Bedingungen des Modells entsprechen, führt zu einer Priorisierung von kürzeren Linien und von Haltestellen, da diese eine höhere Beobachtungswahrscheinlichkeit aufweisen. Das daraus resultierende Wechseln zu jeder Haltestelle, an der die Person vorbeikam, führt zu einer Zerstückelung der Beobachtungen, da es im Zustand einer Haltestelle keine Rolle mehr spielt, woher die Person kam.

Durch das Festlegen von Übergangs- und Beobachtungswahrscheinlichkeiten können die zu erreichenden Zustandssequenzen gesteuert werden. Das Ergebnis ist die längste zusammenhängende Linie, bei der die Abfahrtszeit an der Einstiegshaltestelle und Ankunftszeit an der Ausstiegshaltestelle in dem vorgegebenen Zeitrahmen liegt.

Der Vorteil gegenüber der naiven Methode von Live+Gov liegt darin, dass durch die Notwendigkeit des modellierten Einsteigens an einer Haltestelle besser zwischen Linien, welche dieselbe Strecke fahren, unterschieden werden kann. Wie die Beispiele gezeigt haben, sind auch Fälle eingetreten, in

denen trotzdem die falsche Linie berechnet worden ist, da zwei verschiedene Linien in dem Zeitraum die gleiche Strecke fuhren. Daraus ergeben sich diverse Limitationen. Zunächst ist die Erkennung der Linie stark abhängig vom Zeitpunkt, zu dem sie gefahren worden ist. Problematisch ist dies, wenn z.B. der Bus starke Verspätung hat. Die Methode von Live+Gov ist davon unabhängig und kann die korrekte Linie dennoch erkennen, während die hier vorgestellte Methode kein Ergebnis erzielen kann. Daher ist keines der beiden Verfahren deutlich besser als das andere. Dennoch ist die Verwendung der Haltezeiten elementar für die Unterscheidung von Linien, welche dieselbe Strecke befahren. Mit weiteren Verbesserungen kann dieses Verfahren somit genauere Ergebnisse erzielen. Das folgende Kapitel diskutiert dabei diverse Möglichkeiten.

8 Ausblick

Diese Arbeit bietet eine Grundlage von HMM für die SLD und kann an vielen Stellen erweitert und optimiert werden. Zunächst ist eine Anpassung der gleichverteilten Übergangswahrscheinlichkeiten zwischen Haltestelle und Linie denkbar, um zwei Linien, die während der Beobachtung die gleiche Strecke fahren, zu unterscheiden. Dabei kann eine zur eigentlichen Abfahrtzeit relative Wahrscheinlichkeitsverteilung verwendet werden. Je näher ein Übergang zur Linie an der Abfahrtszeit ist, desto höher ist die Wahrscheinlichkeit dieser Linie gegenüber einer anderen. Auch denkbar ist es, Statistiken der Verspätungen von Linien mit einfließen zu lassen, um eine individuelle Verteilung von Übergangswahrscheinlichkeiten für jede Haltestelle zu ermitteln.

Die aktuelle Implementierung des Algorithmus behandelt sämtliche Trips einer Linie als einen einzigen Trip. Theoretisch ist es somit möglich, dass der Zustand zunächst von einer Haltestelle zu einer Linie wechselt und später von dieser Linie in eine andere Haltestelle, obwohl die Haltezeiten zu zwei verschiedenen Trips gehören. Dieser Fall kann dann eintreten, wenn z.B. ein Fußgänger entlang einer Linie geht. Um dies zu verhindern, könnten Trips als Zustand definiert werden. Dadurch sind die zukünftigen Haltezeiten und damit Übergangswahrscheinlichkeiten mit dem Eintritt in die Linie eindeutig festgelegt. Für folgende Haltestellen würde es nur noch eine Haltezeit geben. Eine andere Möglichkeit besteht darin, die Beobachtungswahrscheinlichkeiten von Linien zeitabhängig zu modellieren, wodurch diese null werden, falls die Beobachtung nicht in einem bestimmten Zeitfenster auftritt.

Die Tests der Implementierung dieser Arbeit wurden zwar mit echten Daten getestet, die von Personen während ihres normalen Alltages aufgezeichnet worden sind, allerdings wurden lediglich Daten aus Helsinki verwendet. Daher sollte der Algorithmus auch in anderen Städten und auch mit weiteren öffentlichen Verkehrsmitteln getestet werden. Dazu befanden sich keine Aufzeichnungen von Autofahrten unter den Testdaten. Da Auto- und Busfahrten leicht zu verwechseln sind, sollten aufgezeichnete Autofahrten getestet und überprüft werden, inwiefern fälschlicherweise Linien erkannt werden.

Die SLD dieser Arbeit kann in andere Systeme integriert werden. Zunächst ist eine Live-Erkennung denkbar, die für kontextsensitive Applikationen relevant ist. Für eine Live-Erkennung muss lediglich nach jedem Schritt des Viterbi-Algorithmus das Backtracking durchgeführt. Dafür empfiehlt es sich, nicht auf das Ausstiegsevent zu warten, um so erste Ergebnisse direkt zu erhalten. Aktuell wird immer nur eine Zustandssequenz ausgegeben, allerdings könnten bei gleichen Wahrscheinlichkeiten

mehrere Zustände gespeichert werden und somit beim Backtracking beachtet werden. Dadurch wäre eine Auflistung der Linien möglich, die der Algorithmus erkannt hat.

Anhang

Die mitgelieferte CD enthält die Daten, die in dieser Masterarbeit verwendet oder erarbeitet worden sind. Die folgenden Daten sind auf dieser CD zu finden:

root

Im Stammverzeichnis befindet sich eine *readme.txt*, welche die hier aufgeführten Informationen enthält.

Ordner *workspace*

Enthält die JAVA-Projekte, die im Rahmen dieser Arbeit entwickelt worden sind. Dies sind die folgenden drei Projekte:

- ServiceLineDetectionGtfs
- AreaCalculatorGtfs
- onebusaway-gtfs

Ordner *Testdaten*

Enthält die Daten, mit denen der Algorithmus getestet worden ist.

Ordner *GTFS*

Enthält die GTFS-Datei, mit der die Tests durchgeführt worden sind.

Ordner *Ergebnisse*

Enthält die Ergebnisse beider Testphasen.

Ordner *Dokument*

Enthält die digitale Version dieser Masterarbeit im .pdf-Format.

Ordner *Bibliotheken*

Enthält die notwendigen Bibliotheken in Form von .jar-Dateien, die für das Programm benötigt werden.

9 Literaturverzeichnis

[1] R. C. Shah, C.-y. Wan, H. Lu und L. Nachman, „Classifying the Mode of Transportation on Mobile Phones using GIS Information," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Seattle, Washington, 2014.

[2] L. Stenneth, O. Wolfson, P. S. Yu und B. Xu, „Transportation Mode Detection using Mobile Phones and GIS Information," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, Illinois, 2011.

[3] P. Zhou, Y. Zheng und M. Li, „How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing," *IEEE Transactions on Mobile Computing,* Bd. 13, Nr. 6, pp. 1228-1241, 2014.

[4] [Online]. Available: http://liveandgov.eu. [Zugriff am 10 03 2015].

[5] Google, „GTFS," [Online]. Available: https://developers.google.com/transit/gtfs/. [Zugriff am 01 03 2015].

[6] M. Ltd., „Mattersoft Oy," [Online]. Available: http://www.mattersoft.fi/. [Zugriff am 27 02 2015].

[7] H. R. Transport, „Helsingin seudun liikenne," [Online]. Available: https://www.hsl.fi/. [Zugriff am 27 02 2015].

[8] A. Klenke, Wahrscheinlichkeitstheorie, Springer, 2013.

[9] Movable Type Ltd, „Movable Type - Information Design & Management," [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html. [Zugriff am 10 03 2015].

[10] Wikipedia, „Wikipedia," [Online]. Available: http://de.wikipedia.org/wiki/Arkustangens_und_Arkuskotangens#Der_.E2.80.9EArkustangens.E2.80.9C_mit_zwei_Argumenten_.28atan2.29. [Zugriff am 22 03 2015].

[11] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE,* vol. 77, no. 2, pp. 257-286, 2 1989.

[12] A. Honkela, Nonlinear Switching State-Space Models, 2001.

10 Abbildungsverzeichnis

11 Tabellenverzeichnis