Using the Proxel Method to build EHMM for behaviour reconstruction

Pascal Krenckel*, Claudia Krull

Institut für Simulation und Graphik, Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Deutschland; **krenckel@ovgu.de*

Abstract.
Die Proxel-Methode wurde an der Otto-von-Guericke-Universität Magdeburg entwickelt, und wird dazu verwendet partiell-beobachtbare Systeme mit diskretem Zustandsraum zu analysieren und das Systemverhalten zu rekonstruieren. Dabei können nur die Forward-Wahrscheinlichkeiten berechnet werden. In diesem Paper stellen wir ein Verfahren vor, dass mithilfe der Proxel-Methode ein Expanded-Hidden-Markov-Model erzeugt. So können nicht nur die Forward-Wahrscheinlichkeiten, sondern auch die Backward-Wahrscheinlichkeiten berechnet werden. Dies ermöglicht bessere Zustandsrekonstruktionen. Dabei konzentrieren wir uns nur auf das Decoding als Problemstellung. Unsere Experimente zeigen, dass diese Methode in Genauigkeit mit der Proxel-Methode mithalten kann.

Einführung

Virtuelle Stochastische Sensoren (VSS) können für verschiedene Probleme angewendet werden. So können sie zum Beispiel zur Aktivitäts-Erkennung im Bereich des Ambient Assisted Living verwendet werden [1]. Auch zur Gestenerkennung von 3D- oder Touch-Gesten können Virtuelle Stochastische Sensoren verwendet werden. [2, 3, 4] Die Proxel-Methode kann zur Verhaltensrekonstruktion partiell-beobachtbarer stochastischer Systeme verwendet werden, insbesondere, wenn diese nicht die Markov-Eigenschaft erfüllen. Dabei kann diese Methode nur die Forward-Wahrscheinlichkeiten berechnen. Doch auch die Backward-Wahrscheinlichkeiten können von Interesse sein. So kann z.B. mithilfe der Forward- und Backward-Wahrscheinlichkeiten berechnet werden, wie Wahrscheinlich ein Zustand s zum Zeitpunkt t bei einer Beobachtungssequenz T ist. Dabei kann $|T|$ auch größer als t sein. Dies ist notwendig, wenn ein bestimmter früherer Systemzustand rekonstruiert werden soll. Wir präsentieren hier eine Methode auf Basis der Proxel-Methode und der Expanded-Hidden-Markov-Modellen, die nicht nur die Forward-Wahrscheinlichkeiten, sonder auch die Backward-Wahrscheinlichkeiten berechnen kann.

1 Hintergrund

1.1 Virtuelle Stochastische Sensoren

VSS wurden 2011 von Claudia Krull eingeführt. [5] Ein VSS modelliert einen doppelt stochastischen Prozess, bei dem nicht nur der interne Systemzustand von einem stochastischen Prozess abhängt, sondern auch die Ausgabe der „physischen Sensoren“ kann von einem stochastischen Prozess abhängen. Die häufig verwendeten Hidden-Markov-Modelle sind auch Virtuelle Stochastische Sensoren. Es können aber auch Modelltypen mit gedächtnisbehaftetem Verhalten wie Augmented Stochastic Petri Nets verwendeten werden. Ein VSS besteht immer aus einem Modell und einem Lösungsverfahren. Dabei gibt es drei Zielstellungen: Evaluierung, Decoding und Training. [5, 6]

Das Evaluierungsproblem versucht die Frage zu beantworten, wie wahrscheinlich bestimme Ereignisse sind - z.B. eine gegebene Beobachtungssequenz. Dadurch kann entschieden werden, welches System wahrscheinlich eine bestimmte Beobachtungssequenz erzeugt hat. Das Ziel des Decodings ist es, bei einer gegebenen Sequenz von Beobachtungen (Trace/Spur) die Sequenz der internen Zustände (Path/Pfad) zu ermitteln. Dabei gibt es verschiedene Herangehensweisen. Die drei meist genutzten sind:

1. Der wahrscheinlichste Pfad
2. Die Sequenz der wahrscheinlichsten Zustände
3. Die Sequenz der wahrscheinlichsten Zustände, die immer noch ein Pfad ist.

Bei Hidden-Markov-Modellen heißen die entsprechenden Algorithmen die diese Sequenzen berechnen: Viterbi, Posterior und Posterior-Viterbi.
Beim Training wird versucht das Parameterset eines Modells zu finden, das die höchste Wahrscheinlichkeit hat eine gegebene Menge von Beobachtungen zu erzeugen.

Für Modelle mit diskreten Zustandsräumen existieren bereits Lösungsverfahren die das Evaluierungsproblem und das Decoding-Problem lösen können. Die Proxel-Methode von Graham Horton (2002) und deren Erweiterungen ermöglichen das Lösen, ohne dass Differenzialgleichungen gelöst werden müssen. Dafür wird die Zeitdimension diskretisiert und in Zeitschritten gerechnet. [7, 8, 9, 10]

1.2 Proxel-Methode

Die Proxel-Methode dient zur Analyse von stochastischen Systemen mit diskreten Zuständen, wie Petri-Netze, Warteschlangensysteme oder Hidden-non-Markovian-Modellen auf Basis der zusätzlichen Variablen (supplementary variables) [7, 11]. Dabei werden die kontinuierlichen Systeme diskretisiert und der Zustandsraum exploriert. Die Systeme müssen nicht die Markov-Eigenschaft erfüllen.

Ein Proxel ist ein Tupel, das den Systemzustand und die Wahrscheinlichkeit speichert. Für jedes Proxel können die nachfolgenden Proxel berechnet werden. Dazu wird die Wahrscheinlichkeit berechnet, dass das System aus dem Zustand in den neuen Zustand innerhalb eines Zeitschrittes wechselt.

Ein interner Systemzustand stellt dabei einen Zustand des Systems dar. Dieser muss nicht beobachtbar sein, aber er ist diskret beschreibbar. Bei einem Warteschlangenproblem könnten dies z.B. die Anzahl der Personen in der Warteschlange sein. Liegt das Model in Form eines Petri-Netzes vor, so stellt das Marking einen internen Systemzustand dar.

Ein Erweiterter Systemzustand enthält dabei noch einen τ-Vektor. Dieser enthält für alle Transitionen, wie lange sie aktiv waren, ohne dass sie ausgelöst haben. Dies ist notwendig, um die Wahrscheinlichkeit zu berechnen, dass eine Transition in einem bestimmten Zeitschritt feuert.

Der erweiterte Zustand eines Proxels, kann dabei noch weitere Informationen enthalten, die zur Analyse benötigt werden. In diesem Paper ist dies z.B. eine Pfad-ID, mit der der Pfad rekonstruiert werden kann. Ein Pfad ist dabei eine Sequenz von internen Zuständen.

Bisher ist es nur möglich vorwärts zu rechnen, und so von einem Eltern-Proxel die Kinderproxel zu bestimmen. Das entspricht dem Forward-Algorithmus. Für das Rückwertsrechnen (Backward-Algorithmus) sind die erweiterten Endzustände nicht bekannt, da die Proxel-Methode diese on-the-fly berechnet. Auch das Rekonstruieren der Elternproxel aus den Kinderproxel ist nicht ohne weiteres möglich, da der ursprüngliche τ-Wert, wenn eine Transition gefeuert hat, nicht bekannt ist. Für das Decoding bedeutet dies, dass nur der Viterbi-Algorithmus (Wahrscheinlichste Pfad) und nicht der Posterior (Sequenz wahrscheinlichster Zustände) oder der Posterior-Viterbi-Algorithmus (Pfad wahrscheinlichster Zustände) verwendet werden kann. [8, 9, 7]

Bei Expanded-Hidden-Markov-Modellen existieren hingegen bereits Algorithmen zum Berechnen der Forward- und Backward-Wahrscheinlichkeiten.

1.3 Expanded-Hidden-Markov-Modell

2007 passte Claudia Krull die Hidden-Markov-Modelle an. [12] Im Gegensatz zu den HMM werden die Ausgaben bei Expanded-Hidden-Markov-Modellen (EHMM) nicht in den Systemzuständen, sondern in den Übergängen erzeugt. Da die resultierenden Berechnungen identisch sind, sind beide Modelle ineinander umwandelbar. Die von uns untersuchten Modelle erzeugen ihre Ausgaben in den Übergängen. Daher ist ein EHMM wesentlich kompakter als das entsprechende HMM. Mit EHMMs lässt sich sowohl der Viterbi-Pfad als auch der Posterior- und Posterior-Viterbi-Pfad berechnen, da diese die Forward- und die Backward-Wahrscheinlichkeiten berechnen können. [12]

Das Berechnen der Backward-Wahrscheinlichkeiten ermöglicht weitere Anwendungsfälle, wie die Rekonstruktion wahrscheinlicher Zustände. Dies ist für den Posterior und den Posterior-Viterbi Algorithmus notwendig.

Insbesondere zeigen frühere Arbeiten, dass für das Decoding-Problem je nach Modell der Posterior bzw. der Posterior-Viterbi-Algorithmus bessere Ergebnisse

erzielen, als der Viterbi-Algorithmus. [13]

Das Ergebnis der Decoding-Algorithmen ist ein einzelner rekonstruierter Pfad aus einer Beobachtungssequenz. Um die Güte dieser Rekonstruktion zu messen, verwenden wir Distanzmaße zwischen dem echten Pfad und dem rekonstruiertem Pfad.

1.4 Distanzmaße für Sequenzen

Die in diesem Paper verwendeten Distanzmaße sind die Hamming-Distanz und die Damerau-Levenshtein-Distanz. Die Hamming-Distanz ist ein weit verbreitetes Distanzmaß. Diese entspricht der Anzahl der Unterschiede zwischen zwei Zeichenketten. Dabei werden nur Zeichen mit gleichem Index miteinander verglichen. Dies hat zur Folge, dass nur Zeichenketten mit gleicher Länge verglichen werden können. Vorteil der Hamming-Distanz ist, dass sie sehr einfach und ressourcensparend zu berechnen ist.

Die Damerau-Levenshtein-Distanz ist die Anzahl der Operationen, die es benötigt, um die Zeichenketten in einander umzuformen. Die Operationen sind: Einfügen, Ersetzen oder Löschen eines Zeichens, sowie das Tauschen zweier benachbarter Zeichen. Die kleinste benötigte Anzahl an Operationen entspricht dann der Distanz. Werden nur Ersetzungsoperationen verwendet, so ist die Damerau-Levenshtein-Distanz gleich der Hamming-Distanz. Aber im Gegensatz zur Hamming-Distanz ermöglicht die Definition als Minimierungsproblem die Verwendung von Zeichenketten unterschiedlicher Länge. Zusätzlich kann das Löschen und Einfügen von Zeichen die Distanz stark senken. Abb. 1 zeigt dieses Problem.

A	B	A	A	B	A	B	A	B	B	A	B	A	A	B	A -
A	B	A	A	B	A	B	A	B	***A***	B	A	B	A	A	B	A

Abb. 1: Zwei Zeichenketten mit Damerau-Levenshtein-Änderung (rot) und Hamming-Übereinstimmung (grün)

Die Damerau-Levenshtein Distanz in Abb. 1 ist eins, da lediglich das hervorgehobene „A“ hinzugefügt bzw. gelöscht werden muss. Die Hamming-Distanz ist hingegen sieben. Dabei wurde eine angepasste Version der Hamming-Distanz verwendet, die „Null“-Zeichen am Ende hinzufügt. Dadurch haben beide Zeichenketten die gleiche Länge. Insbesondere bei der Verwendung von Virtuellen Stochastischen Sensoren und der Berechnung von Ereignissequenzen treten solche Unterschiede sehr häufig auf.

Ein Nachteil der Damerau-Levenshtein-Distanz ist, dass die Laufzeit in $O(n^2)$ liegt. Zudem sind beide Distanz-Maße nicht direkt vergleichbar, da es auf den Anwendungsfall oder die Fragestellung ankommt.[14, 15]

2 Algorithmus

Die Proxel-Methode exploriert den Zustandsraum. Der dabei erzeugte Proxel-Baum entspricht einer Markov-Kette. Auf diesen Gedanken baut die hier vorgestellte Methode auf. Anstatt einer Markov-Kette wird, der mit der Proxel-Methode explorierter Zustandsraum, in ein EHMM umgewandelt. Auf diesem EHMM werden dann die Decoding-Algorithmen angewendet. Das EHMM kann gespeichert werden, da das EHMM systemspezifisch und unabhängig vom Trace ist. In diesem Kapitel stellen wir die Anpassung der Proxel-Methode vor. Abbildung 2 zeigt den Ablauf des Algorithmus zur Erzeugung der EHMM.

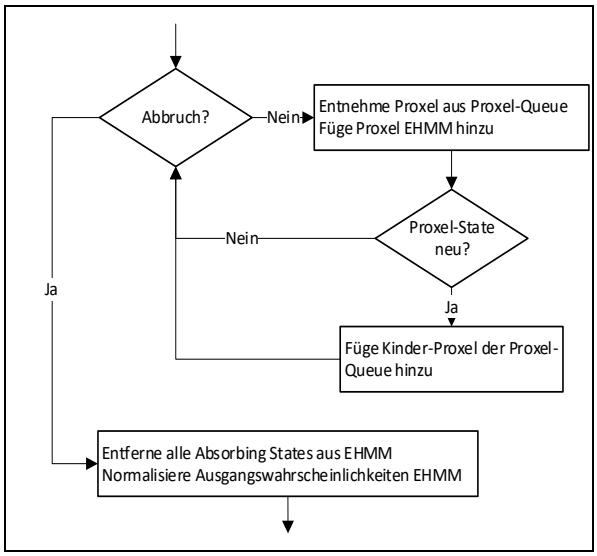

Abb. 2: Programmablaufplan zur Erzeugung der EHMM

2.1 Abbruchbedingungen

Da der Zustandsraum durchaus unendlich sein kann, ist eine Abbruchbedingung notwendig. In diesem Paper

wurde als Abbruchbedingung die Anzahl der Zustände im EHMM gewählt, da diese maßgeblich Einfluss auf Rechenzeit und Genauigkeit der Ergebnisse hat. Andere mögliche Abbruchbedingungen sind Zeitschritt des jüngsten oder ältesten Proxels, sowie die Wahrscheinlichkeit des wahrscheinlichsten Proxels in der Proxel-Queue oder die Gesamtwahrscheinlichkeit, die in der Proxel-Queue noch verbleibt. Es wird abgebrochen, wenn ein bestimmter Grenzwert unterschritten wird.

2.2 Proxel-Queue

Im ursprünglichen Proxel-Algorithmus wurde jeder Zeitschritt vollständig abgeschlossen, bevor der nächste exploriert wurde. Dies ist bei der vorgestellten Methode nicht ratsam. Für dieses Paper wurden die Proxel in einer Priority-Queue mit deren Wahrscheinlichkeiten als Sortierparameter gespeichert. Das wahrscheinlichste Proxel wird zuerst entnommen. Dadurch werden auch die wahrscheinlichsten Zustände zuerst exploriert und sind somit im EHMM enthalten.

2.3 EHMM

Das EHMM enthält alle Zustände und Transitionen. Beim Hinzufügen des Proxels zum EHMM enthält das Proxel alle notwendigen Informationen. Dazu gehört die Transition, die den Zustandswechsel verursacht hat, sowie deren Wahrscheinlichkeit. Informationen über die Ausgabesymbole sind transitionsspezifisch und modellabhängig aber konstant. Auch der Elternzustand ist im Proxel gespeichert, sodass der Zustand oder die Transition dem EHMM hinzugefügt werden können.

Ein Problem sind Senken. Das sind Zustände, die keine Ausgangstransitionen besitzen. Dies trifft auf die letzten hinzugefügten Zustände zu, da deren Kinderzustände noch nicht exploriert wurden. Bei der Berechnung „absorbieren“ diese die Wahrscheinlichkeit. Da diese in der Realität keine Senken sind, müssen sie aus dem EHMM entfernt werden. Dadurch können neue Senken entstehen, die ebenfalls entfernt werden müssen.

Desweiteren müssen auch alle Ausgangswahrscheinlichkeiten normiert werden, damit keine Wahrscheinlichkeit verloren geht. Der Grund für die fehlende Ausgangswahrscheinlichkeiten ist derselbe, wie bei den Senken. Einige Kinderproxel wurden nicht exploriert, weshalb der Link zu den Zuständen fehlt.

3 Experimente

Für die Experimente wurde ein einfaches, nicht markov‘sches Modell durchgerechnet. Sowohl die Proxel-Methode als auch die erzeugten EHMMs können das Evaluations- und das Dekodierungsproblem lösen. In diesem Paper betrachten wir nur das Dekodierungsproblem.

Für das Modell wurden 1000 Pfade mit dazugehörigem Trace generiert. Ziel ist es den echten Pfad aus dem Trace zu dekodieren.

Als Distanzmaß wurde zum Vergleich die Damerau-Levenshtein-Distanz verwendet, sowie die Hamming-Distanz. Da sich aufgrund der Art der Pfade die Hamming-Distanz und die Damerau-Levenshtein Distanz kaum unterscheiden, werden alle Diagramme in diesem Paper nur die Damerau-Levenshtein Distanz enthalten. Die Hamming Distanz ist um rund 10% größer als die Damerau-Levenshtein Distanz.

Als Vergleichsparameter zwischen der hier vorgestellten Methode und der ursprünglichen Proxel-Methode werden relative Distanz zwischen echtem und rekonstruiertem Pfad sowie die Rechenzeit verwendet. Für den Posterior bzw. Posterior-Viterbi-Algorithmus wurde nur die relative Distanz verwendet.

Für die Proxel-Methode wurden die Anzahl der Proxel pro Zeitschritt limitiert, für die EHMMs wurden die Anzahl der Zustände limitiert. Dabei wurden die Modelle mit 10k Zuständen und 100k Zuständen durchgerechnet. Durch das Entfernen aller Senken resultieren diese in einem EHMM mit 7k beziehungsweise 84k Zuständen. Da die EHMMs nur einmal erzeugt werden müssen und dann für jeden Trace wiederverwendet werden können, wurde die Erzeugung nicht in die Rechenzeit mit einbezogen. Diese enthält nur die reine Zeit, die zum Dekodieren benötigt wurde.

3.1 Model

Das System stellt den Bedienprozess einer Autovermietung dar. Die Autovermietung hat im Grundsystem eine Tür. Alle Kunden betreten und Verlassen die Autovermietung durch diese Tür. Passiert

ein Kunde die Tür, so wird dies über einen Sensor registriert. Es gibt Premium-Kunden, die mit Priorität behandelt werden, und eine längere Bedienzeit haben. Es gibt einen Angestellten. Ziel des VSS ist es, die Länge der Warteschlangen anhand des Türprotokolls zu bestimmen.

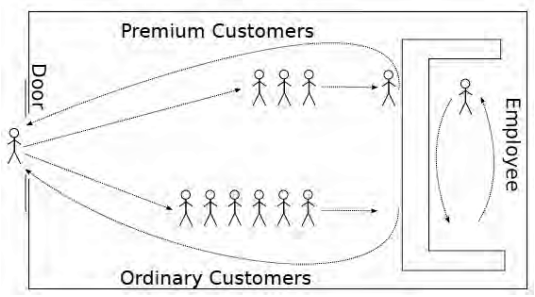

Abb. 3: Autovermietung mit zwei getrennten Warteschlangen für VIP und normale Kunden und einem Angestellten

Die Zwischenankunftszeit der normalen Kunden ist Normal-Verteilt: $\mathrm{N}_{\mathrm{ord}}(22.5, 5)$. Die Zwischenankunftszeit der VIP-Kunden ist ebenfalls Normal-Verteilt: $\mathrm{N}_{\mathrm{vip}}(45{,}5)$. Die Bearbeitungszeit pro Kunde ist Weibull-Verteilt: $\mathrm{W}_{\mathrm{ord}}(12{,}2)$ und $\mathrm{W}_{\mathrm{vip}}(16{,}2)$

Die Bearbeitungszeit pro Kunde (11.8 Zeiteinheiten) ist im Schnitt geringer als deren Zwischenankunftszeiten (15 Zeiteinheiten). Dies ist wichtig, da sich sonst die Kunden aufstauen würden. Ein Zustand enthält die Anzahl der Personen in den jeweiligen Warteschlangen sowie den Zustand des Angestellten. Zur Diskretisierung wurde ein Zeitschritt von 5 Zeiteinheiten gewählt.

3.2 Rechenzeit

Um die Rechenzeit zu vergleichen wurde die Zeit, die die jeweiligen Algorithmen benötigen, gemessen. Hier zeigt sich, dass die Proxel-Methode schneller als die Hybride-Proxel-Methode ist. Dies liegt daran, dass die Modelle mit deutlich weniger Zuständen/Proxel berechnet werden können, ohne Einbuße der Genauigkeit. Bei gleicher Anzahl an Proxel/Zuständen ist die Hybride-Proxel-Methode schneller. Die Proxel-Methode mit 1k-Zuständen ist doppelt so schnell, wie das EHMM. Das EHMM mit 84k Zuständen ist 34x langsamer als das EHMM mit 7k Zuständen, zur besseren Übersichtlichkeit ist diese daher nicht in Diagramm 1 enthalten. Die Werte in Diagramm 1 wurde normiert zum EHMM mit 7k Zuständen, um das Verhältnis der Methoden besser darzustellen. Beide Methoden haben linearen Rechenaufwand in Abhängigkeit zu Pfadlänge.

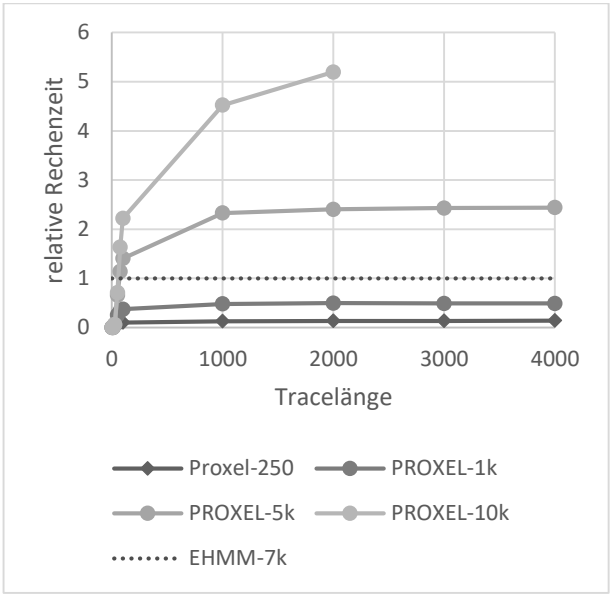

Diagramm 1: relative Rechenzeit normiert zu EHMM-7k

3.3 Genauigkeit

Die Hybride-Proxel-Methode zeigt ähnliche Ergebnisse, wie die Proxel-Methode. Die Ergebnisse für die EHMM mit 7k Zuständen sind geringfügig schlechter als die Proxel-Methode.

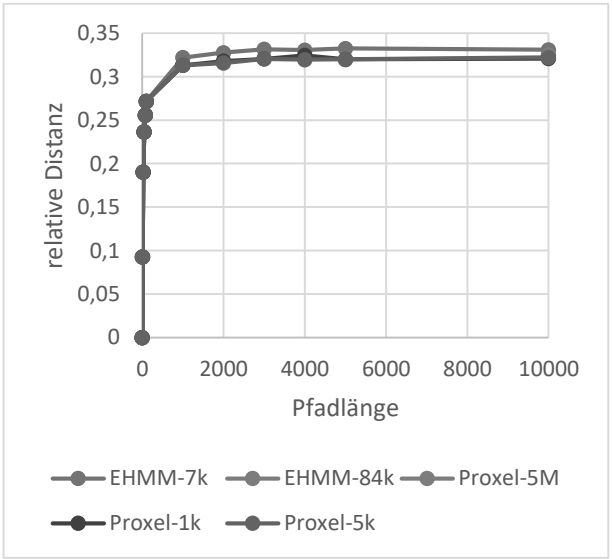

Diagramm 2: Durchschnittliche relative Distanz zum richtigen Pfad für die Proxel-Methode und die Hybride-Proxel-Methode (EHMM) mit verschiedener Begrenzung des

Zustandsraums.

Die Proxel-Methode liefert für dieses Modell sogar noch bis 250 Proxel pro Zeitschritt bessere Ergebnisse als die EHMM mit 7k Zuständen, wenn der Viterbi Algorithmus verwendet wird. Die Ergebnisse zwischen 250 Proxeln und 5M Proxeln unterscheiden sich dabei nicht. Hier ist die Proxel-Methode eindeutig überlegen.

Verwendet man allerdings den Posterior- oder den Posterior-Viterbi Algorithmus, so erzielt das EHMM bessere Ergebnisse. Diagramm 3 zeigt die durchschnittliche relative Distanz zu den jeweiligen echten Pfaden. Die Hybride-Proxel-Methode mit 7k hat die höchste Distanz mit dem Viterbi-Algorithmus. Die Hybride-Proxel-Methode mit 84k Zuständen und dem Viterbi-Algorithmus und die Proxel-Methode liefern etwas schlechtere Ergebnisse. Die geringste Distanz liefert der Posterior/Posterior-Viterbi-Algorithmus. Die Ergebnisse zwischen 7k und 84k und zwischen Posterior und Posterior-Viterbi unterscheiden sich dabei nicht.

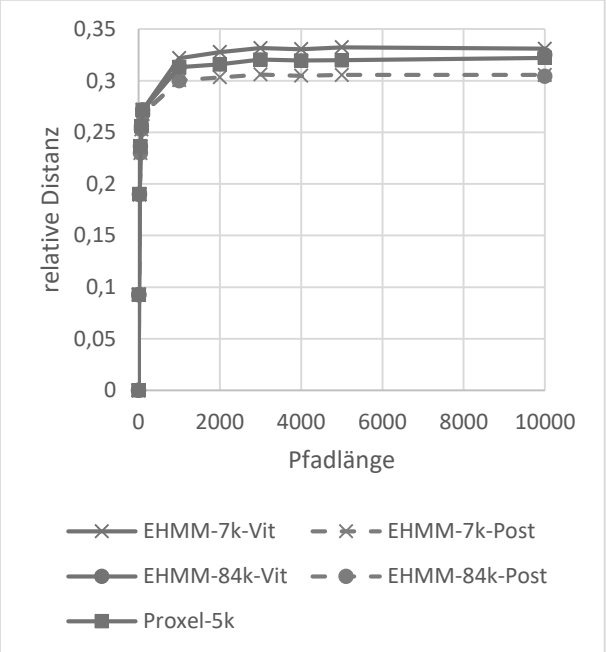

Diagramm 3: Durchschnittliche relative Distanz zum richtigen Pfad für die Proxel-Methode und die Hybride-Proxel-Methode (EHMM)

4 Result Discussion

Das Konvertieren der Proxel-Methode in ein EHMM ist nicht nur möglich, sondern bietet auch einige Vorteile. Im Gegensatz zur Proxel-Methode können nicht nur der Forward-Algorithmus, sondern auch der Backward-Algorithmus ausgeführt werden. Dadurch kann nicht nur der Viterbi-Algorithmus zum Dekodieren verwendeten werden, sondern auch Algorithmen wie Posterior oder Posterior-Viterbi-Algorithmus.

Unsere vorgestellte Methode zeigt nur kleine Einbuße bezüglich des Erkennungsgrades. Betrachtet man den Posterior bzw. Posterior-Viterbi-Algorithmus, so erzielt das EHMM sogar bessere Ergebnisse. Ob allerdings der Posterior oder Posterior-Viterbi-Algorithmus angewendet werden kann, ist Problem abhängig. Insbesondere der Posterior-Algorithmus muss nicht zwangsläufig einen echten Pfad ausgeben: Für zwei aufeinanderfolgende Zustände in der Ergebnissequenz muss es keinen Übergang von ersteren in den zweiten geben.

Das Berechnen der Forward und Backward-Wahrscheinlichkeiten hat allerdings noch andere Vorteile. So kann dadurch die Frage beantworten werden, wie Wahrscheinlich ein Zustand s zum Zeitpunkt t ist, unter der Bedingung das ein Trace T beobachtet wurde. Dabei kann die Trace-Länge auch größer als t sein. Die Proxel-Methode kann diese Frage nicht beantworten.

Im Bezug auf Rechenleistung hingegen kann das EHMM nicht mit der Proxel-Methode mithalten. Selbst der Viterbi-Algorithmus benötigt bis zu 8x mehr Rechenzeit bei vergleichbaren Ergebnissen. Der Posterior-Algorithmus benötigt entsprechend 16x mehr Rechenzeit. Dies liegt daran, dass die Proxel-Methode nur valide Zustände exploriert. Das verwendete Modell erzeugt in jedem Zeitschritt eine Ausgabe, die τ-Werte hängen daher direkt mit den beobachteten Ausgaben zusammen. So ist der kleinste τ-Wert z.B. die Anzahl Zeitschritte seit der letzten Ausgabe. In den EHMM sind alle Zustände unabhängig von den Beobachtungen gespeichert. Zudem wurde die Zeit zum Erzeugen der EHMMs nicht mit eingerechnet. Muss nur ein einzelner kurzer Pfad dekodiert werden, so ist der Overhead für die Konvertierung zu groß.

Zudem kann bei besonders langem Pfaden der Speicherverbrauch auch ein limitierender Faktor sein. Der Speicherverbrauch des Forward- und Backward-Algorithmus liegt in $\mathrm{O}(n * |T|)$, wobei n die Anzahl der Zustände des EHMMs ist und $|T|$ die Tracelänge. Bei sehr langen Traces ist daher der Speicherverbrauch der

limitierende Faktor, da Viterbi und Posterior jeweils auf den Forward/Backward-Algorithmus aufbauen. Diese können, wie der Name suggeriert, nur in eine Richtung durchlaufen werden. Ein Zurückführen zu früheren Werten ist nicht möglich. Für das Backtracking werden alle Werte in der umgedrehten Reihenfolge benötigt. Daher müssen alle Werte für die Berechnung gespeichert werden. Um den Speicherverbrauch zu reduzieren haben wir auch eine angepasste Variante der Algorithmen entwickelt. Anstelle alle Berechnungen zu speichern, werden nur die Ergebnisse jedes $\sqrt{|T|}$ -tem Schritt gespeichert. Die Werte können dann vom letzten Zwischenergebnis aus berechnet werden. Der Speicherverbrauch reduziert sich so auf $O(n * \sqrt{|T|})$. Für den Viterbi-Algorithmus bedeutet dies ungefähr eine Verdopplung der Rechenzeit, da alle Berechnungen zweimal ausgeführt werden müssen. Der Rechenaufwand des Posterior-Algorithmus steigt um 50%, da nur entweder der Forward- oder der Backward-Algorithmus angepasst werden muss.

Die verwendeten Distanz-Maße Damerau-Levenshtein, sowie die Hamming-Distanz liefern ähnliche Ergebnisse. Dies ist der Art des Models und der Pfade geschuldet. Da alle Ereignisse beobachtbar sind, sind die Pfadlängen und die Veränderung in den einzelnen Zuständen zeitlich genau bestimmbar. Dadurch verhält sich die Damerau-Levenshtein ähnlich zur Hamming-Distanz. Das Einfügen eines Zustandes und Löschen eines anderen Zustandes ist hier nicht hilfreich, um einen Pfad in einen anderen umzuwandeln. Auch das Vertauschen von Zuständen verringert die Distanz nicht, da nicht die Ereignisse, sondern die internen Zustände im Pfad enthalten sind. Werden zwei Events in falscher Reihenfolge erkannt, so ist der Anfangs und Endzustand dennoch derselbe.

Einschränkend ist allerdings zu sagen, dass nur ein Modell zum Vergleich verwendet wurde. Die Einsetzbarkeit und der Nutzen der Proxel-Methode, als auch der Konvertierung in ein EHMM hängen stark vom Modell und Anwendungsfall ab. Das verwendete Model ist stationär, was dem EHMM entgegen kommt. Das Modell fällt immer wieder in dieselben Zustände zurück. Wäre dies nicht der Fall, dann würden viele der Zustände des EHMMs bei längeren Traces nur einmal benötigt werden. Das EHMM ist dann deutlich größer als notwendig, was die Rechenzeit erhöhen würde. Zudem erzeugt jedes Ereignis eine Ausgabe. Bei nicht beobachtbaren Ereignissen, kann es sein, dass auch die Proxel-Methode deutlich mehr Zustände pro Zeitschritt benötigen würde.

Bezüglich der Geschwindigkeit ist anzumerken, dass die Experimente auf i7-6700HQ ausgeführt wurden. Der Prozessor ist schon mehrere Jahre alt. Auf neueren Prozessoren können die Ergebnisse anders ausfallen. Auch die Möglichkeit der Parallelisierung wurde nicht untersucht. Für die Proxel-Methode ist ein parallelisieren innerhalb eines Zeitschrittes nur schwer möglich. Beo den EHMMs ist dies allerdings sehr einfach, da es innerhalb eines Zeitschrittes pro Zustand nur Concurrent-Reads und keine Concurrent-Writes auf derselben Speicheradresse gibt.

Die sauberere Neuimplementation der EHMMs ist zudem 4x schneller als die in diesem Paper verwendete Version, da ein unnötiger Hash-Table-Lookup entfernt wurde. Damit können die EHMMs auch im Bereich Geschwindigkeit mit der Proxel-Methode mithalten. Auch ein Viterbi auf Basis eines Dijkstra kann zu zusätzlichen deutlichen Geschwindigkeitssteigerungen führen. Erste Tests legen nahe, das auch die Verwendung von Sparse-Vectors einen erheblichen Geschwindigkeitsvorteil bringt. Ähnlich wie bei der Proxel-Methode werden dann auch nur die Zustände berechnet, deren Wahrscheinlichkeit nicht 0 ist. Wir haben uns aber gegen die neue Version für dieses Paper entschieden, um sicher zu gehen, dass wir nicht eine einseitige Optimierung für den neuen Algorithmus vornehmen. Auch bei der Proxel-Methode kann noch viel Optimierungspotenzial vorhanden sein. Neben der Optimierung der Datenstrukturen, scheint vor allem Step-Size-Control hier eine vielversprechende Idee zu sein.

5 Conclusion and Future Work

Unsere Ergebnisse haben gezeigt, dass die Umwandlung in ein EHMM für das Dekodierungsproblem im Bereich Genauigkeit mit der Proxel-Methode mithalten kann. Im Bereich Rechenzeit erzielt unsere neue Methode schlechtere Ergebnisse. Allerdings zeigen bereits neue Untersuchungen, dass

dort noch deutlich Optimierungspotential besteht und die neue Methode durchaus mit der alten Methode auch im Bereich Rechenzeit mithalten kann.

Mit unserer vorgestellten Methode lassen sich allerdings nicht nur die Forward-Wahrscheinlichkeiten berechnen, sonder auch die Backward-Wahrscheinlichkeiten. Dadurch können deutlich mehr Anwendungsfelder abgedeckt werden. Zukünftige Untersuchungen müssen zeigen, ob dadurch sogar Smoothing und Training von Modellen möglich wird. Auch dies ist mit der Proxel-Methode bisher nicht möglich.

Allerdings wurde unsere Methode nur an einem Modell getestet. In wie weit die EHMMs für größere und nicht stationäre System anwendbar ist, muss noch untersucht werden. Ebenfalls ist es interessant wie die EHMMs mit nur teilweise-beobachtbaren Ereignissen performt. Während das Laufzeitverhalten der Proxel-Methode dadurch deutliche Verschlechterungen erfährt, wird das Laufzeitverhalten der EHMMs dadurch nicht beeinflusst. Aber ob dies auch für die Genauigkeit der Fall ist, muss noch untersucht werden.

Zudem wurde nur das Decoding untersucht. Ob die EHMMs auch das Evaluierungs-Problem genauso gut lösen können, wie die Proxel-Methode, müssen zukünftige Untersuchungen noch zeigen. Davon ist allerdings auszugehen, da der im Viterbi-Algorithmus verwendete Forward-Algorithmus sich nur wenig vom echten Forward-Algorithmus unterscheidet.

References

[1] L. Fialho Müller, *Feasibility and Applicability of Virtual Stochastic Sensors for Human Activity Recognition in the Context of Ambient,* 2022.

[2] T. Dittmar, C. Krull and G. Horton, *A new approach for touch gesture recognition: Conversive Hidden non-Markovian Models,* vol. 10, 2015, pp. 66-76.

[3] T. Dittmar, C. Krull und G. Horton, *AN IMPROVED CONVERSIVE HIDDEN NON-MARKOVIAN MODEL-BASED TOUCH GESTURE RECOGNITION SYSTEM WITH AUTOMATIC MODEL CREATION,* Bergeggi, 2015.

[4] T. Dittmar, C. Krull und G. Horton, *Evaluating a New Conversive Hidden non-Markovian Model Approach for Online Movement Trajectory Verification,* Porto, 2017.

[5] C. Krull, Virtual Stochastic Sensors: Formal Background and Example Applications, Magdeburg: Shaker Verlag Düren, 2021.

[6] C. Krull, R. Buchholz und G. Horton, „Virtual Stochastic Sensors: How to gain Insight into Partially Observable Discrete Stochastic Systems," in s *The 30th IASTED International Conference on Modelling, Identification and Control*, 2011.

[7] G. Horton, „A NEW PARADIGM FOR THE NUMERICAL SIMULATION OF STOCHASTIC PETRI NETS WITH GENERAL FIRING TIMES," in s *European Simulation Symposium*, Dresden, 2002.

[8] S. Lazarova-Molnar und G. Horton, *Proxel-Based Simulation of Stochastic Petri Nets,* SCS European Publishing House, 2004.

[9] S. Lazarova-Molnar und G. Horton, *Proxel-Based Simulation of Stochastic Petri Nets Containing Immediate Transitions.*

[10] F. Wickborn, C. Isensee, T. Simon, S. Lazarova-Molnar und G. Horton, *A New Approach for Computing Conditional Probabilities of General Stochastic Processes,* 2006.

[11] D. R. Cox, „The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables," *Mathematical Proceedings of the Cambridge Philosophical Society,* pp. 433-441, 1955.

[12] C. Krull und G. Horton, *EXPANDED HIDDEN MARKOV MODELS: ALLOWING SYMBOL EMISSIONS AT STATE CHANGES,* Prague, 2007.

[13] P. Fariselli und P. L. Martelli, *The posterior-Viterbi: a new decoding algorithm for hidden Markov models,* 2005.

[14] V. I. Levenshtein, „Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady,* Bd. Vol. 10, p. 707, February 1966.

[15] F. J. Damerau, „A technique for computer detection and correction of spelling errors," *Commun. ACM,* Bd. 7, pp. 171-176, March 1964.

git-Repository

Alle Quellcodes und Daten werden unter folgendem Link zur Verfügung gestellt: https://github.com/Virtual-Stochastic-Sensors