SEMI-AUTOMATISCHE GENERIERUNG UND REINFORCEMNET LEARNING BASIERTES TRAINING EINES DIALOGMANAGERS

Stefan Hillmann[1], *Klaus-Peter Engelbrecht*[2], *Benjmain Weiss*[1]

[1]*Technische Universität Berlin,* [2]*Deutsche Telekom AG*
stefan.hillmann@tu-berlin.de

Kurzfassung: In diesem Beitrag beschreiben wir unseren Ansatz, einen Dialogmanager aus einer Ontologie heraus zu generieren und diesen dann mittels Q-Learning und einer Nutzersimulation zu trainieren. Dieser Ansatz wird auf die Domäne eines Troubleshooter angewendet, für den aktuell eine händisch definierte und gewartete Implementierung existiert. Die gelernte Dialog Policy senkt signifikant die Länge der Dialoge und Höhe der Nutzeraufwände.

1 Einleitung

Für begrenzte Domänen kann das Verstehen von Sprache (Natural Language Understanding, kurz NLU) für sprachbasierte Auskunfts- und Hilfesysteme anhand weniger Beispielsätze händisch definiert und mit Machine Learning (ML) robust trainiert werden, was den manuellen Aufwand durch Experten reduziert. Für die Dialogsteuerung, insbesondere für komplexere Dialoge, wie sie von menschlichen Mitarbeitern problemlos umgesetzt werden können, ist der Einsatz von ML zur Steuerung des Dialogverlaufs jedoch noch nicht in der Breite angekommen. Dabei sind ML-Ansätze mit Bedarf an großen Datenmengen echter Dialoge unpraktikabel, da hier die Herausforderung besteht, solche Daten zu erhalten und aufzubereiten, als auch eine Generalisierung oder einen Domänentransfer zu ermöglichen. Als Lösung bietet sich die Nutzung von Nutzersimulatoren an, die auf einer abstrakten Repräsentationsebene agieren, und deren erzeugte Gesprächssequenzen für die Optimierung des Dialogmanagers mittels ML auszuwerten. Das Ziel dieses Beitrags ist es deshalb, auf Basis einer existierenden Beispiel-Ontologie, die eine Anwendungsdomäne beschreibt, zunächst semi-automatisch einen Dialogmanager zu generieren und diesen anschließend mittels Reinforcement Learning (RL) zu trainieren, um die Verwendung eines verallgemeinerbaren Ansatzes zu demonstrieren.

2 Stand der Forschung

Gegenüber dem Stand der wirtschaftlich verwendeten Technik gibt es für aufgabenorientierte Domänen verschiedene Forschungsansätze zur (semi-)automatischen Erstellung von statistischen Dialogmanagern. Neben einem sequence-to-sequence Lernen für generative Modelle aus bestehenden (meist Chat-) Daten [1] steht der Ansatz von RL mit simulierten Daten [2], der im Folgenden näher beschrieben wird.

2.1 Reinforcement Learning zum Training von Dialogmanagern

Es kann zwischen drei verschiedenen Ansätzen für ML unterschieden werden: Supervised Learning (SL), Unsupervised Learning (UL) sowie Reinforcement Learning (RL). SL und UL benötigen für das Training vorab gesammeltes Datenmaterial, bei SL muss dieses zusätzlich gelabelt sein. Für den Fall, dass ein Dialogmanager auf Basis von SL oder UL trainiert werden

	Aktion			
Zustand	a_1	a_2	a_3	a_4
s_1	**80**	7	0	20
s_2	**60**	23	11	40
s_3	0	**98**	3	2
s_4	50	**70**	60	30
s_5	13	0	**99**	0

Tabelle 1 – Beispiel für eine trainierte q-Matrix für ein System mit 5 Zuständen 4 Aktionen.

soll, müssten also Korpora mit dem Dialogverhalten von menschlichen Dialogpartnern (z.B. Callcenter-Mitarbeiter) vorliegen.

Im Fall von Q-Learning (QL) [3][4, S. 131 f.], als eine Art des RL, sind solche Daten, aus denen der Dialogmanager das gewünschte Interaktionsverhalten lernt, nicht notwendig. Vielmehr beruht das Training im Wesentlichen auf dem Bewerten des Ergebnisses einer Aktion, die der Dialogmanager in einem gegeben Zustand des Dialogs ausgeführt hat. Diese Bewertung erfolgt mittels einer Rewardfunktion, mit welcher der nun aktuelle, durch die Aktion erreichte, Dialogzustand bewertet wird [5]. Eine einfache Rewardfunktion verwendet bspw. die Dialoglänge und den Aufgabenerfolg als Eingangsparameter. Es können aber auch, neben anderen, Parameter wie die Bewertung durch einen Nutzer oder der Aufwand für den Nutzer zur Beantwortung einer Systemaktion zum Einsatz kommen. Alleinstellungsmerkmal sind hierbei Rewards, die sich an dem (tatsächlichen oder erwarteten) Ende einer Sequenz von Aktionen ergeben.

Während des Trainings wird also das später genutzte Systemverhalten, die sogenannte Dialog Policy (im Folgenden Π oder Strategie genannt), gelernt bzw. anhand des Reward optimiert. Beim Q-Learning wird Π als Matrix (q-Matrix) repräsentiert, in der jede Zeile einem der möglichen Dialogzustände S und jede Spalte einer der Systemaktion A zugeordnet ist. Dabei wird angenommen, dass die Dialogzustände voneinander unabhängig sind. Eine Zelle Π enthält den q-Wert für die entsprechende Zustand-Aktion-Kombination und wird als $Q(S,A)$ bezeichnet. Dieser q-Wert gibt den derzeitig erwarteten finalen Reward für diese Aktion an.

Tabelle 1 zeigt eine beispielhafte q-Matrix für eine trainierte Strategie Π in der, der jeweils höchste q-Wert einer Zeile **fett** hervorgehoben ist. Ist $Q(S,A) = 0$, wie zum Beispiel für $Q(s_3, a_1)$ in Tabelle 1, dann bedeute dies gewöhnlich, dass A während des Trainings aus Zustand s heraus nicht ausgeführt wurde. Dies ist der Fall, wenn die Aktion in diesem Zustand nicht zur Verfügung steht, oder nicht genug Trainingsläufe durchgeführt wurden, um diese Aktion für diesen Zustand im Training ausführen zu können. Bei der Nutzung von Π im laufenden Betrieb bestimmt der Dialogmanager zunächst den Zustand in dem sich der Dialog befindet (im Fall eines slotbasierten Systems z. B. anhand der aktuellen Belegung der Slots) und führt dann die Aktion mit dem höchsten q-Wert in der entsprechenden Zeile aus (z. B. nach dem Wert für einen bestimmten, noch nicht belegten Slot fragen).

In unserer Implementierung des q-Learning erfolgt das Update von Π nach Gleichung 1, wie von Sutton und Barto beschrieben [4, S. 131]. Deren Beschreibung des Q-Learning beruht wiederum auf den Arbeiten von Watkins [3]. Bei uns beträgt die Lernrate (engl. *learning rate*) $\alpha = 0,1$ und der Diskontierungsfaktor (engl. *discount factor*) $\gamma = 0,5$. Währen des Trainings erfolgt die Auswahl der Aktion A in Zustand S immer zufällig, d.h. für den ε-greedy Ansatz [4, S. 27 f.] beträgt $\varepsilon = 1$. Sowohl die verwendete Rewardfunktion zur Bestimmung von r_t in Gleichung 1, als auch Details des Trainings, die für Sprachdialogsysteme spezifisch sind, werden in

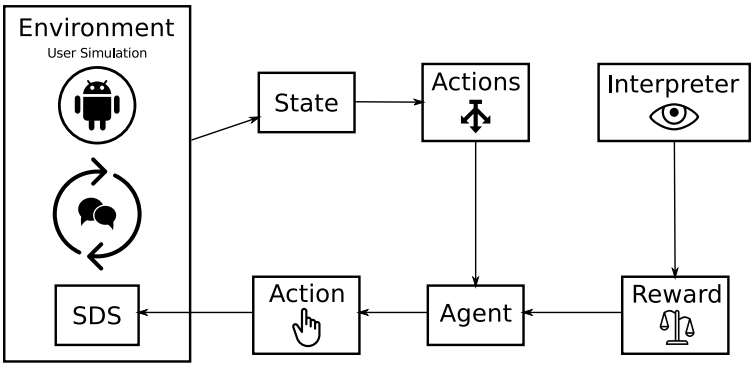

Abbildung 1 – Schematische Darstellung des Reinforcement Learning Zyklus.
(State: Zustand, Actions: Aktion(en), Environment: Umgebung, SDS: Sprachdialogsystem).

Abschnitt 4 beschrieben.

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[r_t + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)] \quad (1)$$

Abbildung 1 zeigt schematisch den zyklischen Ablauf beim Training der Strategie eines Dialogmanagers mittels Reinforcement Learning. Der Dialog befindet sich in einem bestimmten Zustand (State, S_t in Gleichung 1) in dem verschiedenen Aktionen (Actions) möglich sind. Der Agent wählt eine Aktion (A_t) aus und lässt diese durch das System ausführen. Die Ausführung der Aktion in der Umgebung (Environment) der Anwendung führt in einen neuen Zustand (S_{t+1}). Ein Interpreter, der die Umgebung und die Auswirkung von Aktionen darin beobachtet, bestimmt mit Hilfe eine definierten Rewardfunktion den Reward (r_t) den der Agent für die Aktion erhält. Mit r_t führt der Agent zum Abschluss ein Update auf Π aus indem der neue Wert für $Q(S_t,A_t)$ berechnet wird (s. Gleichung 1).

Bei einem Sprachdialogsystem (SDS) hängt die Zustandsänderung nicht nur von dem aktuellen Zustand und der gewählten Aktion ab, sondern auch von der Reaktion des Nutzers, mit dem das SDS in Interaktion steht. Um ein effizientes Training zu ermöglichen, interagiert das SDS nicht mit echten Nutzern, sondern mit einer Nutzersimulation, die eine hohe Anzahl von Trainingsläufen in kurzer Zeit erlaubt. Die Anzahl der notwendigen Trainingsläufe hängt mit der Anzahl von Zuständen und Aktionen zusammen. Für ein hypothetisches System, in dem ein Zustand über 10 Variablen mit jeweils 3 möglichen Werten (z. B. 1, 2, 3) beschrieben werden kann, beträgt die Anzahl der möglichen Zustände bereits $|S| = 3^{10} = 59.049$. Wenn nun im System 5 Aktionen möglich sind, wären rechnerisch $|S| * |A| = 3^{10} * 5 = 295.245$ Trainingsläufe notwendig, um jedes $Q(A,S)$ wenigstens einmal zu aktualisieren. Um den exponentiellen Wachstum des Zustandsraums zu begegnen, wurden verschiedene Methoden entwickelt auf die hier nicht weiter eingegangen werden soll. Eine Übersicht dazu geben u. a. Rieser und Lemon [5, S. 40 f.]

Da eine Nutzersimulation zum Einsatz kommt, muss der Reward aus Parametern der Interaktion geschätzt werden, bspw. aus der Dialoglänge und dem Aufgabenerfolg. Die Details der von uns verwendeten Rewardfunktion sind in Abschnitt 4.4 beschrieben.

3 Anwendungsdomäne

Unseren Ansatz einen Dialogmanager semi-automatisch aus einer strukturierten Wissensbasis zu generieren und dann mittels einer Nutzersimulation und Q-Learning zu trainieren, haben wir an dem Anwendungsfall eines Chatbots implementiert. Der Chatbot ist ein Assistent, der seine Nutzer bei der der Lösung von Problemen mit einer Internetverbindung unterstützen soll. Die Interaktion mit dem Chatbot (kurz Bot) verläuft systemgetrieben, da der Bot dem Nutzer diagnostische Fragen stellt, um dass Problem einzugrenzen und am Ende des Dialogs Vorschläge

Entity	Beschreibung	Werte
Internetproblem	Die Art der wahrgenommenen Internetstörung	z. Bsp. zu langsam, keine Verbindung
Symptom	Das Symptom für die Ursache der Störung	z. Bsp. Kein Wifi-Symbol in der Statusleiste oder rot leuchtende LED am Router etc.
Ursache	Die Ursache für die Internetstörung	z. Bsp. Wifi deaktiviert, Wifi Verbindung instabil, DSL-Kabel defekt etc.
Lösung	Eine Lösung zur Beseitigung der Ursache	z. Bsp. Wifi-Repeater installieren, defektes LAN-Kabel tauschen etc.

Tabelle 2 – Übersicht zu den Entities in der Anwendungsdomäne.

zur Lösung des Problems präsentiert.

Tabelle 2 listet die im Dialog verwendeten Entities mit einer kurzen Beschreibung und exemplarischen Werten, mit denen das Entity belegt werden kann, auf. Ein Ausschnitt einer Ontologie die uns zur Verfügung gestellt wurde haben wir, in Form von RDF[1], zum Erstellen der Zustandsrepräsentation im Dialogmanager verwendet und um den möglichen Dialogzustandsraum (DZR) daraus abzuleiten. Da die Ontologie die Beziehungen zwischen den Entities eindeutig beschreibt (z. B. Ursache X verursacht Internet Problem Y) können daraus, über geeigneten SPARQL-Anfragen [6], generisch der DZR als auch ein Zustandstracker, der durch den Dialogmanager verwendet wird, erzeugt werden.

4 Dialogmanager und Trainingsumgebung

4.1 Dialogzustandsraum

Der Dialogzustandsraum (DZR) ist als gerichteter Graph modelliert, bei dem ein Knoten jeweils einen möglichen Zustand bzw. State (*S*) und jede Kante eine mögliche Aktion (*A*) des Dialogmanagers repräsentiert. Jeder Zustand wird darüber definiert, ob ein konkreter Wert, den ein Entity annehmen kann, bereits gefragt wurde (0 = nein, 1 = ja). Um nicht jeden Zustand über 32 binär belegte Variablen beschreiben zu müssen, was in Kombination zu einem DZR mit $2^{32} (\approx 4 * 10^9)$ Zustände führen würde, wurde die Anzahl der Zustände reduziert,indem für jedes Entity eine eigene relevante Untermenge von Zuständen verwendet wird. Dies reduziert die Anzahl der Zustände im DZR auf 8.414 und ist in Abbildung 2 schematisch dargestellt.

Die 40 verschiedenen Aktionen im DZR können durch den Dialogmanager ausgeführt werden und fragen entweder nach der Bestätigung für einen konkreten Wert (z. B. *Sie haben eine langsame Internetverbindung?*) oder nach dem Wert für ein Entity (z. B. *Welches Problem haben Sie mit Ihrem Internetanschluss?*

4.2 Dialogmanager

Der Dialogmanager (DM) besteht aus einem Modul, das über Dialogakte mit einer Nutzerschnittstelle kommunizieren kann, sowie einem Zustandstracker der für alle (hier 32) möglichen Werte festhält, welche im Dialog bereits behandelt wurden. Weiterhin enthält der DM Slots, die für jedes Entity den vom Nutzer entweder bestätigten oder genannten Wert festhalten.

Die verwendeten Dialogakte entsprechen den CUED Standard Dialogakten [7] und werden von dem DM benutzt, um mit menschlichen Nutzern (über Module zum Sprachverstehen und zur Spracherzeugung) oder direkt mit einer Nutzersimulation zu kommunizieren. Das Training

[1] `https://www.w3.org/RDF/`

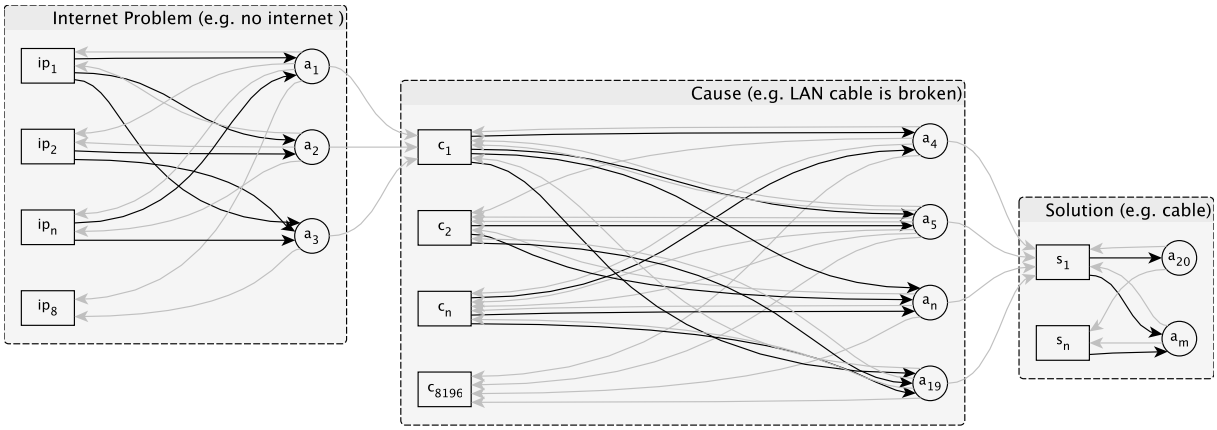

Abbildung 2 – Schematische Darstellung des kaskadenartig modellierten Dialogzustandsraum. Zustände (rechteckige Knoten) beziehen sich auf jeweils ein Entity. Ein schwarzer Pfeil beschreibt, dass in einem Zustand die referenziert Aktion genutzt werden kann. Graue Pfeile verweisen auf potentielle Folgezustände einer Aktion.

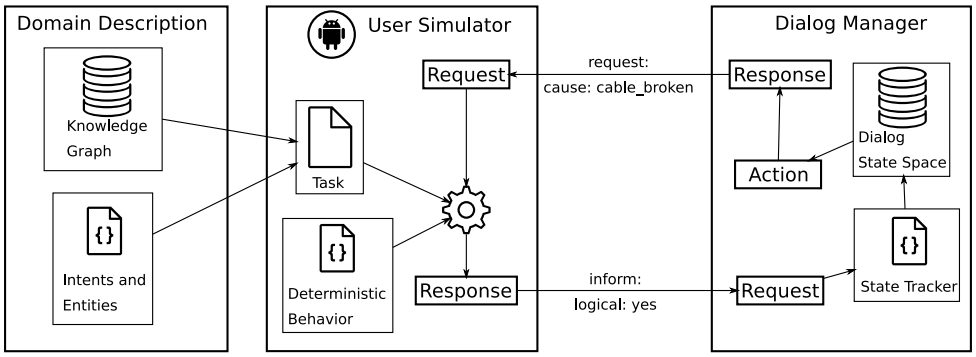

Abbildung 3 – Interaktion zwischen Nutzersimulation (User Simulator) und dem Dialogmanager über Dialogakte.

und die Evaluation, die in diesem Beitrag beschrieben werden, wurden mit einer Nutzersimulation durchgeführt.

4.3 Interaktion mit der Nutzersimulation

Der Interaktionsverlauf zwischen einer Nutzersimulation (NS) [8, 9] und dem DM ist in Abbildung 3 schematisch dargestellt. Im Rahmen eines Turns wählt der DM eine der im aktuellen Zustand verfügbaren Interaktionen aus und sendet der NS den entsprechenden Dialogakt (z. B. *Confirm(internet_problem: no_internet)*). Die NS antwortet dem DM mit einem passenden Dialogakt (z. B. *Affirm()* als explizite Bestätigung). Auf Basis der Antwort der NS führt der DM ein Update des Zustandtrackers aus und verwendet anschließend dessen Belegung, um im DZR den passenden Zustand zu suchen, also den Zustand mit der identischen Wertebelegung für das aktuell im Dialog behandelte Entity. Nun kann der DM wieder eine der möglichen Aktionen auswählen und damit den nächsten Turn starten.

4.4 Rewardfunktion

Das Ziel unserer Rewardfunktion ist es die Strategie des Dialogmanagers hinsichtlich von 3 Kriterien zu optimieren: 1) Der Dialog soll erfolgreich sein, 2) die Dialoglänge soll minimiert werden, 3) der Aufwand des Nutzers soll minimiert werden.

Der Aufwand des Nutzers ist über eine Kostenfunktion modelliert, die für eine vom DM ausgeführte Aktion beschreibt, wie hoch der geschätzte Aufwand eines Nutzers ist, um auf die Systemaktion zu antworten. Bspw. bedeutet es einen geringen Aufwand zu Prüfen ob ein

Netzwerkkabel in einen Router eingesteckt ist, jedoch ist es ein hoher Aufwand die korrekte Funktion des DSL-Ports eines DSL-Modems zu prüfen. Bei der Berechnung des Reward (im folgenden Absatz beschrieben), wird zwischen hohen ($k_t = 10$) und niedrigen ($k_t = 5$) Kosten unterschieden, die für alle Aktionen vordefiniert sind.

Die entsprechende, beim Training verwendete, Rewardfunktion, um r_t aus Gleichung 1 zu berechnen, ist in Gleichung 2 gegeben. In ihr wird durch den Term $100 - t_t$ die Anzahl der Turns (d. h. die Dialoglänge) berücksichtigt, die benötigt werden, um das Aufgabenziel zu erreichen. Die Anzahl der bisher im Dialog durchgeführten Turns wird durch t_t gegeben, d. h. je mehr Turns benötigt werden, um das Aufgabenziel zu erreichen, um so geringer fällt der globale Reward aus. Der Parameter k_t gibt die Kosten an, die aus der vom System durchgeführten Aktion resultieren.

$$r_t = \begin{cases} 100 - t_t + 10 - k_t, & \text{wenn das Aufgabenziel erreicht wurde.} \\ 10 - k_t, & \text{wenn das Aufgabenziel } \textit{nicht} \text{ erreicht wurde.} \end{cases} \tag{2}$$

Ein weiterer Parameter $t_{max} = 20$ ist im DM und gibt an wie lang ein Dialog maximal werden darf, um noch erfolgreich sein zu können. Der DM bricht den Dialog nach t_{max}-vielen Schritten ab.

5 Evaluation

Dieser Abschnitt beschreibt die Evaluation des in Abschnitt 4 beschrieben Dialogmanagers, dessen Strategie zu Aktionsauswahl mittels Q-Learning trainiert wurde. Zunächst wird die Evaluationsmethodik beschrieben. Anschließend werden in Abschnitt 5.2 die Evaluationsergebnisse gezeigt und erläutert.

5.1 Methodik

Das Training mittels Q-Learning optimiert die Auswahl der nächsten Systemaktion durch den Dialogmanager in einem bestimmten Zustand des Dialogs. Um den Einfluss des Trainings auf die Leistung des Dialogmanagers zu prüfen, wurden mittels der oben beschrieben Nutzersimulation zwei Dialogkoropora mit jeweils 1.000 Dialogen erzeugt. Die Implementierung des Dialogmanagers war in beiden Fälle dieselbe. Variiert wurde ausschließlich die Strategie zur Auswahl der nächsten Systemaktion aus der Menge der in einem gegeben Dialogzustand verfügbaren Aktionen. Zum Einen fand die Auswahl über Π_z statt, zum Anderen über die trainierten Strategie Π_q. Bei der Strategie *zufällig* (Π_z), sind die Ausfallwahrscheinlichkeiten für alle im Zustand S verfügbaren Aktionen gleich groß. Das Vorgehen ist hier analog zur Auswahl von Aktionen beim Training des Dialogmanagers. Bei der Nutzung von $\Pi_q$wird in S diejenige Aktion mit dem höchsten trainierten q-Wert gewählt. Ist für mehr als eine Aktion der gleiche höchste q-Wert vorhanden, wird unter diesen Aktionen zufällig gewählt.

5.2 Ergebnisse

Im Folgenden steht l_d für die Länge des Dialog d gemessen in Turns, wobei ein Turn aus einem Systemturn gefolgt von einem Nutzerturn besteht. Weiterhin steht N für die Anzahl von Dialogen in einer Bedingung (hier die jeweils evaluierte Strategie). Um die mit zwei verschiedenen Strategien erzeugten Dialogkorpora zu vergleichen und damit Rückschluss auf die Leistung des jeweils verwendeten Strategie des Dialogmanagers zu erhalten, wurde jeweils die mittlere Dialoglänge ($\widetilde{l}$), die mittleren Dialogkosten ($\widetilde{k_d}$) sowie die mittleren Kosten je Turn ($\widetilde{k_t}$) berechnet. Die Gleichungen in Zeile 3 zeigen die formale Berechnung der genannten Werte. Dort

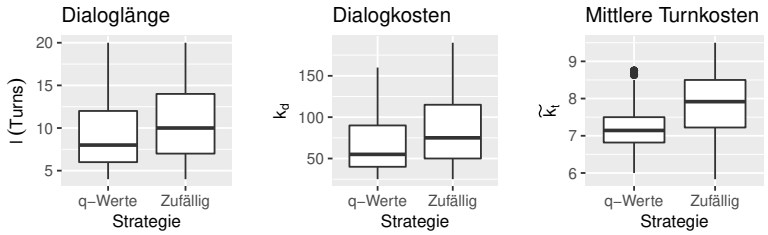

Abbildung 4 – Boxplots für die Länge aller Dialoge (links), die Aktionskosten aller Dialoge (mitte) und die Aktionskosten je Turn für alle Dialoge.

	Dialoglänge		Dialogkosten		Kosten je Turn		
Strategie	$\widetilde{l}$	s_l	$\widetilde{k_d}$	s_{k_d}	$\widetilde{k_t}$	s_{k_t}	Erfolgsrate
Π_z (zufällig)	10,80	4,76	87,16	43,67	7,83	0,79	90,3 %
Π_q (q-Werte)	9,06	4,36	66,54	35,21	7,19	0,60	95,8 %

Tabelle 3 – Mittelwert $\widetilde{x}$ und empirische Standardabweichung s [10, S. 31] für Dialoglänge l, Dialogkosten k_d und Kosten je Turn k_t sowie die Erfolgsrate, für die zufällige und gelernte Aktionsauswahl.

beschreibt $k_{t,d}$ die Summe der Aktionskosten aller Turns in einem Dialog, wobei k_i die Kosten für den i-ten Turn sind.

$$\widetilde{l} = \frac{\sum_{d=1}^{N} l_d}{N},\ \widetilde{k_t} = \frac{\sum_{d=1}^{N} k_{t,d}}{N},\ k_{t,d} = \frac{\sum_{i=1}^{l_d} k_i}{l_d} \tag{3}$$

Die Ergebnisse dieser Berechnungen werden Abbildung 4 gezeigt und sind in Tabelle 3 aufgelistet. Diese Tabelle enthält zusätzlich die Angabe der Erfolgsrate. Diese gibt hier an in wie viel Prozent der Dialoge eine zur Aufgabe passende Lösung gefunden werden konnte, bevor die maximal zulässige Anzahl ($t_{max} = 20$, s. Abschnitt 4.4) von Turns erreicht wurde.

Für die Verteilungen von l, k_d und k_t ($\widetilde{l}$, $\widetilde{k_d}$ und $\widetilde{k_t}$ in Tabelle 3) kann mit einem einseitigen Wilcoxon-Rangsummentest jeweils gezeigt werden, dass die gemessenen Werte für die Strategie Π_z signifikant größer sind, als für die Strategie Π_q. Die Ergebnisse der Unterschiedstests sind $W = 619680$, $p < 0,00001$ für l, $W = 648600$, $p < 0,00001$ für k_d und $W = 739640$, $p < 0,00001$ für k_t. Diese bedeutet, dass die mittels der trainierten Strategie Π_q gesteuerten Dialoge weniger Turns benötigten und geringere Kosten verursachten als die mittels Π_z gesteuerten Dialoge.

6 Diskussion und Ausblick

Der vorgestellte Ansatz zeigt beispielhaft ein Vorgehen, um aus einer domänenspezifischen Wissensbasis halbautomatisch einen Dialogmanager zu generieren und mittels Reinforcement Learning zu trainieren. Die funktionale Evaluierung, belegt eine signifikante Verbesserung der Effizienz (kürzere Dialog, geringere Dialogkosten) durch Reinforcement Learning mittels simuliertem Nutzerverhalten. Die Relevanz dieser Verbesserung und die Güte der erreichten Dialogstrategie muss im Folgenden auch subjektiv durch Evaluierungen mit menschlichen Nutzern überprüft werden.

Aus Usabilitysicht gilt es, diesen Ansatz bezüglich Anforderungen an Entwickler/Designer und benötigten Aufwände mit der manuellen Erstellung von Expertensystemen zu vergleichen.

In Folgearbeiten sollen die Rewardfunktionen besondere Aufmerksamkeit bekommen. Während die hier verwendete Berücksichtigung des fachlichen, zeitlich und kognitiven Nutzerauf-

wandes ein erster Schritt ist, müssen domänenspezifische und für Nutzer relevante Metriken identifiziert und verwendet werden. Zudem bietet sich hier die Möglichkeit, den Einfluss von Interaktionsdesignern im maschinellen Lernen zu untersuchen.

Weitere Untersuchung sind auch bei der verwendeten Methode des Reinforcement Learning Ansatzes (z. B. Sarsa [4, S. 129 f.] oder Deep q-Learning (DQL) [11]) und der Reduktion des Zustandsraums (ZR) notwendig. Insbesondere DQL würde eine explizite Reduktion des ZR vermeiden.

7 Danksagung

Wir bedanken uns für die Unterstützung durch das Bundesministerium für Bildung und Forschung (BMBF). welches die Arbeit an diesem Beitrag durch Förderung des Projekts OKS - Optimierung konversationeller Schnittstellen (Förderkennzeichen 16SV8151) unterstützt.

Literatur

[1] SERBAN, I. V., R. LOWE, L. CHARLIN, und J. PINEAU: *Generative deep neural networks for dialogue: A short review*. In *NIPS 2016 workshop on Learning Methods for Dialogue*. 2016.

[2] ULTES, S.: *Towards natural spoken interaction with artificial intelligent systems*. In *Proceedings der ESSV*. 2018.

[3] WATKINS, C. J. C. H.: *Learning from Delayed Rewards*. Ph.D. thesis, King's College, 1989.

[4] SUTTON, R. S. und A. G. BARTO: *Reinforcement Learning: An Introduction (Draft from March 21, 2018)*. The MIT Press, Cambridge, Massachusetts, 2018. URL `http://incompleteideas.net/book/bookdraft2018mar21.pdf`.

[5] RIESER, V. und O. LEMON: *Reinforcement Learning for Adaptive Dialogue Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-24942-6.

[6] DUCHARME, B.: *Learning SPARQL: Querying and Updating with SPARQL 1.1*. O'Reilly Media, Sebastopol, CA, second edition edn., 2013.

[7] YOUNG, S.: *CUED Standard Dialogue Acts*. Tech. Rep., 2009.

[8] SCHATZMANN, J. und S. YOUNG: *The Hidden Agenda User Simulation Model*. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4), S. 733–747, 2009. doi:10.1109/TASL.2008.2012071.

[9] HILLMANN, S.: *Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems*. Ph.D. thesis, Technische Universität Berlin, Berlin, 2017. doi:10.1007/978-3-319-62518-8.

[10] BORTZ, J. und C. SCHUSTER: *Statistik Für Human- Und Sozialwissenschaftler*, Bd. 2. Springer Medizin, 7 edn., 2010.

[11] MNIH, V., K. KAVUKCUOGLU, D. SILVER, A. GRAVES, I. ANTONOGLOU, D. WIERSTRA, und M. A. RIEDMILLER: *Playing atari with deep reinforcement learning*. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602. 1312.5602`.