VERGLEICHENDE UNTERSUCHUNG VON REINFORCEMENT LEARNING VERFAHREN BEIM TRAINING EINES DIALOG MANAGERS

Stefan Hillmann, Tilo Himmelsbach, Benjamin Weiss

Technische Universität Berlin
stefan.hillmann@tu-berlin.de

Kurzfassung: Unter Verwendung des Frameworks PLATO wurde die Policy eines Dialogmanagers mit drei verschiedenen Reinforcement Learning Algorithmen trainiert. Es zeigt sich, dass REINFORCE etwas schneller lernt als WoLF-PHC und deutlich schneller als Q-Learning. Auch hinsichtlich des erreichten Aufgabenerfolgs liegen REINFORCE und WoLF-PHC in etwa gleich auf und beide deutlich vor Q-Learning.

1 Einleitung

Das Trainieren eines Agenten mittels Reinforcement Learning soll diesen Agenten befähigen mit einer gegebenen Umgebung derart zu interagieren, dass die Interaktion, bezogen auf ein gegebenes Ziel hin, optimiert wird. Das Ziel der und die Möglichkeit zur Optimierung ist von der jeweiligen Aufgabe und Umgebung abhängig. Des Weiteren bestimmt die verwendete Reward-Funktion auf welche Faktoren hin optimiert werden soll, z. B. kurze/lange Interaktionen bzw. das Erreichen oder Halten eines Zustands. Bei aufgabengetriebenen Interaktionen mit einem Sprachdialogsystem (SDS) hat der Dialogmanager (DM) das Ziel einen Dialog in kurzer Zeit zu einem erfolgreichen Abschluss (z. B. dem Nutzer eine gesuchte Information zu übermitteln) zu bringen. Ein weiteres Ziel kann ein möglichst geringer Zusatzaufwand für den Nutzer sein [1].

Wenn, bei gleichem Verhalten der Umwelt (Datenbank mit Informationen und der simulierte Nutzer) sowie der selben Reward-Funktion, das verwendete Verfahren des Reinforcement Learnings variiert wird, zeigen sich Unterschiede in der Effizienz und Effektivität des Trainings [2]. Effizienz adressiert dabei die Geschwindigkeit mit welcher der Agent lernt (z. B. Anzahl von Trainingsdialogen) während sich die Effektivität aus der Leistungsfähigkeit des trainierten DM nach dem Training ergibt. Diese kann u. a. über die durchschnittliche Länge der Dialoge und die Dialogerfolgsrate, aber auch mittels des kumulierten Reward, gemessen werden.

Im Folgenden beschreiben wir die Evaluation von 3 verschiedenen Verfahren des Reinforcement Learning (RL), mit denen jeweils die Policy eines Dialogmanagers trainiert wurde. Dazu stellen wir in Abschnitt 2 das Framework PLATO vor, welches wir für das Training und die Evaluation verwendet haben. In Abschnitt 3 beschreiben wir zunächst die Anwendungsdomäne für die der Dialogmanager bzw. die Policies trainiert wurden. Danach beschreibt der gleiche Abschnitt wie das Training und die Evaluation durchgeführt wurden. Zusätzlich stellt Abschnitt 3.4 kurz unsere alternative Implementierung des REINFORCE Algorithmus für PLATO vor. Abschnitt 4 stellt die Ergebnisse des Trainings und der Evaluation der trainierten Dialogpolicies vor. Zum Abschluss enthält Abschnitt 5 eine Diskussion der Ergebnisse und auffälliger Eigenschaften von PLATO.

		Informationsart		
		Nutzer		System
Feld	Beschreibung	Wissen	Erfragbar	Erfragbar
ECTS	ECTS Punkte eines Kurses	x		x
Grad	Master- oder Bachelorkurs	x		x
Studiengang	Name des Studiengangs	x		x
Tag	Tag an dem der Kurs stattfindet	x	x	x
Lehrstuhl	Name des Lehrstuhls bzw Fachbereichs	x	x	x
Nachname	Nachname des Kursleiters	x	x	x
Vorname	Vorname des Kursleiters		x	
Büro	Raumnummer des Büros des Kursleiters		x	
Webseite	URL des Webseite des Fachbereichs		x	

Tabelle 1 – Datenfelder aus der Alex-Datenbank [10] die in unserer Untersuchung verwendet wurden.

2 PLATO Framework

Die von uns ab Abschnitt 3 beschriebenen Arbeiten bauen auf dem PLATO[1]-Framework [3] auf, welches im Folgenden kurz beschrieben wird.

Das Framework adressiert verschiedene Teilprobleme der gesamten Verarbeitungskette konversationeller Nutzerschnittstellen. Es lassen sich Module zur Spracherkennung (ASR) und Sprachgenerierung (TTS) einbinden, der Fokus der Arbeit [3] liegt jedoch auf dem Training zweier Agenten, welche währen des Lernvorganges miteinander über natürliche Sprache interagieren. Hierzu werden zunächst die neuronalen Modelle für das Sprachverstehen (NLU) und Sprachgeneration (NLG) auf annotierten Daten vortrainiert und anschließend mittels Reinforcement-Learning (RL) nachtrainiert. Die angewandten RL-Algorithmen sind das Q-Learning, das auf diesem aufbauende "Win or Lose Fast Policy Hill Climbing"(WoLF-PHC) sowie der REINFORCE-Algorithmus [4].

3 Methodik

Innerhalb dieser Arbeit fokussieren wir uns auf die Analyse des Lernverhaltens verschiedener RL-Algorithmen. Hierbei beschränken wir uns auf eine Nutzer-System-Interaktion auf Basis von Dialogakten. Der System-Agent interagiert mit einer agendabasierten Nutzersimulation nach dem Vorbild von Schatzmann [5]. Da der System-Agent und die Nutzersimulation einzig über Dialogakte interagieren und die Agenda der Nutzersimulation für den System-Agenten verborgen ist, lässt sich dieses Dialogsetting auch als „partially observable Markov decision process“ (POMDP) interpretieren [6][7]. Das Q-Learning sowie der REINFORCE-Algorithmus sind in [8] beschrieben. Der auf dem Q-Learning aufbauende WoLF-PHC-Algorithmus nutzt eine variable Lernrate um besseres Konvergierungsverhalten zu erzielen [9].

3.1 Anwendungsdomäne

Die verwendete Domäne ist das elektronische Modul- und Vorlesungsverzeichnis der TU Berlin (TUB), für das bereits ein, auf Hidden Markov Models basierender, Chatbot (Alex) implementiert wurde [10]. Im Rahmen dieser Arbeit wurde eine Datenbank erstellt, die regelmäßig mit den in den Informationssystemen der TUB verfügbaren Informationen aktualisiert wird. Ein

[1] https://github.com/uber-research/plato-research-dialogue-system

Auszug aus dieser Datenbank, der alle Kurse aber nicht alle verfügbaren Datenfelder enthält, wurde für das Training und die Evaluation verwendet. Insgesamt enthält die in das PLATO-Framework importierte Datenbank 27.205 Datensätze. Hierbei muss berücksichtigt werden, dass ein Kurs in mehreren Datensätzen adressiert sein kann, z. B. wenn er in mehreren Studiengängen anrechenbar ist.

In Tabelle 1 sind alle Datenfelder aufgelistet die durch den Dialogmanager behandelt werden können. Bei Feldern die dem Wissen des Nutzer zugeordnet sind, kann der Nutzer den entsprechenden Wert auf Nachfrage des Systems nennen. Vom Nutzer erfragbare Felder sind solche die vom Nutzer, gewöhnlich nach dem Anbieten einer Lösung durch den Dialogmanager, zusätzlich erfragt werden können. Vom System erfragbare Felder beschreiben die Felder, die vom System erfragt werden können, um mit diesen Kriterien eine Lösung in der Datenbank zu suchen. Damit entsprechen die vom System erfragbaren Felder, den Slots die durch den Dialogmanager gefüllt werden sollen.

Um das Untersuchungsszenario einfach zu halten, wurde die Zuordnung der Felder so gewählt, dass das Wissen des Nutzers in jedem Fall die notwendigen Informationen zur Lösung einer Aufgabe enthält.

3.2 Nutzersimulation und Dialogmanager

Es wird die in PLATO bereitgestellte Nutzersimulation verwendet um in Interaktion mit dem Dialogmanager Dialoge zu erzeugen die wiederum zum Training einer Policy genutzt werden können, oder um anhand der erzeugten Dialoge die Leistungsfähigkeit einer bereits trainierten Policy zu beurteilen. Die Policy wird vom Dialogmanager verwendet, um auf Basis des aktuellen Dialogzustands zu entscheiden welche Systemaktion als nächstes ausgeführt wird.

Die in PLATO verwendete Nutzersimulation ist entsprechend der von Schatzmann vorgestellten agendabasierten Nutzersimulation (ANS) implementiert [5, 3]. Für die Simulation eines Dialogs verwendet die ANS eine Agenda die als Stack implementiert ist und dazu genutzt wird die Dialogakte, die zum Dialogmanager gesendet werden sollen, zu verwalten. Die Agenda wird initial befüllt, indem aus der Datenbank, auf welcher der Dialogmanager arbeitet (vgl. Abs. 3.1, zufällig ein Datensatz gelesen wird und damit, wiederum in zufälliger Reihenfolge, Dialogakte in die Agenda geschrieben werden. Welcher Dialogakttyp für welche Information verwendet wird, hängt u.a. von der in Tabelle 1 gezeigten Zuordnung von Datenfeldern zu Informationsarten ab. Dadurch, dass die Aufgabe aus der Datenbank erzeugt wird, lässt sich nach Abschluss eines Dialogs bestimmen, ob dieser Dialog erfolgreich war oder nicht, da die vom Dialogmanager angebotene Lösung dem Datensatz entsprechen muss auf dem die initiale Agenda des ANS basiert.

Für unsere Untersuchung fand die Interaktion zwischen der ANS und dem Dialogmanager ausschließlich auf der Dialogaktebene statt, d. h. Informationen wurden über Dialogakte (z. B. inform(tag: montag)) ausgetauscht und nicht in Form natürlicher Sprache.

Die in PLATO implementierte ANS beinhaltet eine Fehlersimulation, die eine Vertauschung von Konzepten oder dem Wert eines Konzepts in den von der ANS gesendeten Dialogakten erlaubt. Beide Fehlerraten wurden jeweils auf eine Auftretenswahrscheinlichkeit von 4 % konfiguriert.

Ein Dialogzustand wird im Dialogmanager durch die aktuelle Belegung der Slots (vom System erfragbar in Tabelle 1) sowie dem zuletzt vom System gesendeten Dialogakt und zuletzt empfangen Dialogakt des Nutzers repräsentiert.

3.3 Rewardfunktion

Je nach Dialogerfolg oder -misserfolg wird ein positiver bzw. negativer Reward von +20 bzw. -1 gezählt. Pro Dialogschritt (Turn) erhält der Agent einen Strafreward von -0,05. Der Dialog wird als erfolgreich gewertet, wenn die vom System angebotene Lösung (in unseren Daten ein Modul aus dem Vorlesungsverzeichnis) allen vom Nutzer gesetzten Anforderungen genügt und weiterhin alle vom Nutzer erfragten Information zu dieser Lösung durch das System beantwortet wurden.

3.4 Implementierung REINFORCE

Da mit der PLATO-Implementierung des REINFORCE-Algorithmus kein plausibles Lernverhalten erzeugt werden konnte, entschieden wir uns für eine eigene, auf dem PyTorch-Framework [11] basierende, Umsetzung. Hierbei orientierten wir uns stark an einer existierenden Beispielimplementierung[2]. Eine nennenswerte Abweichung von der Methodik im Beispiel-Code stellt unser Verzicht auf eine Normierung der Returns[3] dar, weil eben diese bei Dialogerfolg alle Dialogschritte, ausgenommen den letzten, mit einem negativen Return belegen würde.

Im Gegensatz zur ursprünglichen Implementierung von REINFOCRE in PLATO nutzen wir zur Kodierung des Dialogzustands die gleiche Logik wie PLATOs Implementierung beim Q-Learning, damit Dialogzustände durch die selben Informationen repräsentiert werden.

3.5 Training

Mit den in Abschnitt 2 eingeführten Verfahren wurden 3 Policies (π_Q (Q-Learning), π_W (WoLF-PHC) und π_R (REINFORCE) trainiert. Das generelle Vorgehen beim Training wird im Folgenden kurz dargestellt. Soweit nicht explizit erwähnt sind die Parameter für das Training für alle 3 Verfahren gleich gewählt worden.

Das Training einer Policy π erfolgte in allen 3 Fällen mit jeweils 40.000 Dialogen (D) die iterativ generiert und zum Training von π verwendet wurden:

1. Generiere 500 initiale Dialoge in D
2. Trainiere π dreimal wie folgt:
   (a) Ziehe zufällig 500 Dialoge aus D und überschreibe damit D_s
   (b) Für jeden Dialog d in D_s führe ein Update von π aus.
3. Prüfe ob insgesamt 40.000 Dialog generiert wurden ($|D| \leq 40.000$)
   - Falls ja: Beende das Training und speichere π
   - Falls nein: Generiere 50 neue Dialoge unter Verwendung von π, füge diese zu D hinzu und gehe zu Schritt 2.

Kurz gefasst, wurde zunächst ein Batch von 500 Dialogen erzeugt. Im Anschluss wurden mit Hilfe der Nutzersimulation, des Dialogmanagers und π jeweils 50 Dialoge generiert und π dann mit zufällig gewählten Dialogen (aus der Menge aller Dialoge die bis dahin erzeugt wurden) nachtraininert. Nach der Erzeugung von 40.000 Dialogen endete das Training.

Die gewählten Werte für die Batchgröße ($B = 500$ Dialoge), das Trainingsintervall ($I = 50$ Dialoge), Anzahl der Trainingssessions je Intervall ($S = 3$) und die Zielgröße von $|D| = 40.000$

[2] `https://github.com/pytorch/examples/blob/master/reinforcement_learning/reinforce.py`
[3] der Return zu einem Dialogschritt stellt die diskontierte Summe zukünftiger Rewards dar

Dialogen führen dazu, dass π insgesamt 1.185.000 mal mit einem zufällig aus D gewähltem Dialog trainiert wurde (s. Gleichung 1).

$$\frac{|D| - B}{I} \times S \times B = \frac{40.000 - 500}{50} \times 3 \times 500 = 1.185.000 \quad (1)$$

Zu Beginn des Trainings betrug die Explorationsrate 100 % ($\gamma = 1$). Während des Trainings wurde γ nach jedem Trainingsintervall mit dem Faktor $0,9974016$ multipliziert. Diese Absenkung wurde durchgeführt bis γ den Wert $0,01$ erreicht hatte. Dieser Wert wurde nach 30.000 generierten Trainingsdialogen erreicht. Im Fall der Exploration hat der Dialogmanager entweder eine regelbasierte Policy (π_B, s. 3.6) verwendet (Warm-up) oder zufällig eine mögliche Aktion ausgeführt. Diese Entscheidung (Warm-up oder zufällige Aktion) wurde zufallsbasiert getroffen, wobei die Auswahlwahrscheinlichkeit jeweils 50 % betrug.

3.6 Evaluation

Zur Evaluation der trainierten Policies wurden mit jeder jeweils ein Datensatz von 1.000 Dialoge erzeugt und die Datensätze hinsichtlich der Maße Reward, Dialoglänge und Aufgabenerfolg verglichen. Zusätzlich wurden auch 1.000 Dialoge mit einer rein regelbasierten und auf die Implementierungen des Dialogmanagers und der ANS hin optimierten Policy (π_B) erzeugt. Dieser Datensatz dient als Baseline bei der Evaluation. Die Verwendung von π_B führt, in unserem Setting, immer zu einem erfolgreichen Dialog, jedoch nicht notwendiger Weise zum schnellstmöglichen (im Sinne der Anzahl der notwendigen Dialogschritte) Dialog.

4 Ergebnisse

Im Folgenden werden zunächst einige Informationen zum Trainingsverlauf der einzelnen Policies gegeben und im Anschluss die Ergebnisse der vergleichenden Evaluation vorgestellt.

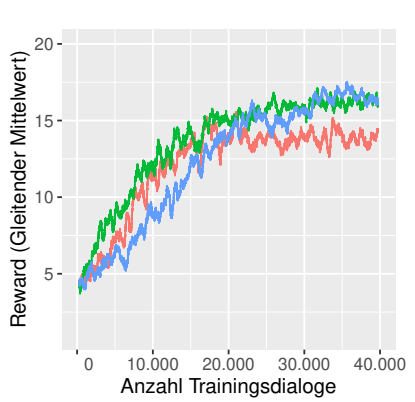
4.1 Training

Die Abbildungen 1a und 1b zeigen die Entwicklung des mittleren Rewards und des Mittelwertes des Aufgabenerfolgs anhand eines gleitenden Mittelwerts mit einer Fenstergröße von 500 Dialogen. Die auf der X-Achse aufgetragene Anzahl der bisher im Training erzeugten Dialoge stellt einen Anhaltspunkt für den Fortschritt des Trainings dar, die Anzahl der tatsächlichen Trainingsiterationen liegt jedoch deutlich höher (vgl. Abschnitt 3.5).

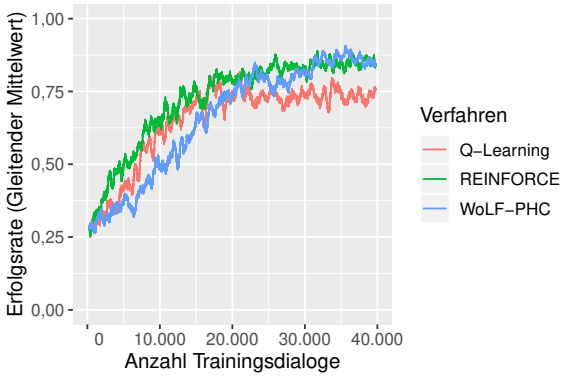
Die Verläufe der Graphen in den Abbildungen 1a und 1c zeigen, dass REINFORCE und Q-Learning zunächst schneller lernen als WoLF-PHC. Q-Learning stagniert jedoch ab ca. 20.000 Trainingsdialogen unterhalb eines mittleren Rewards von 15, während REINFORCE und WoLF-PHC den mittleren Reward steigern und zum Ende des Trainings hin in etwa die selben Werte erzielen.

Das in Abbildung 1b die Verläufe der Graphen zur Aufgabenerfolgsrate recht ähnlich zu denen in Abbildung 1a sind liegt an dem hohen Gewicht des Aufgabenerfolgs bei der Berechnung des Rewards (s. Abschnitt 3.3).

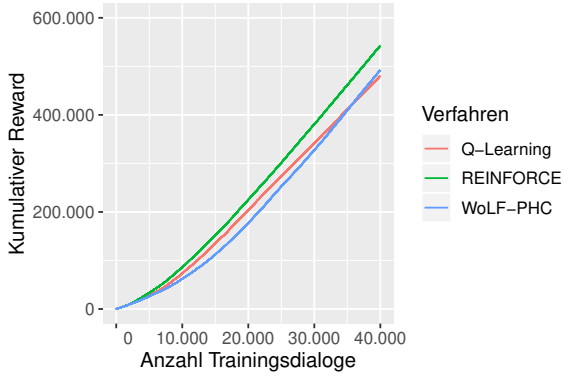
Der für REINFORCE insgesamt höhere kumulative Reward, der sich aus der Aufsummierung der Rewards aller bis zu einem bestimmten Zeitpunkt generierten Dialoge ergibt, hat seine Ursache in dem früheren Erreichen höher Rewards bei dem Training mit REINFORCE.



(a) Gleitender Mittelwert des Rewards (Fenstergröße: 500 Dialoge)



(b) Gleitender Mittelwert der Dialogerfolgsrate (Fenstergröße: 500 Dialoge)



(c) Kumulativer Reward über alle beim Training generierten Dialoge

Abbildung 1 – Entwicklung des Rewards und des Aufgabenerfolgs über den Verlauf des Trainings hinweg. Die oberen beiden Abbildungen zeigen gleitende Mittelwerte (Fenstergröße 500 Dialoge) während die untere Abbildung den kumulierten Reward der generierten Dialoge zeigt. Die Daten sind auf der X-Achse in der Reihenfolge in der die Dialoge erzeugt wurden aufgetragen.

4.2 Evaluation

Die Ergebnisse der Evaluation der trainierten Policies sind in Tabelle 2 aufgelistet. Die rein regelbasierte Policy π_B, die auch für das Warm-up beim Training verwendet wurde, erzielt den höchsten Reward da sie eine Erfolgsrate von 100 % erreicht. Warum eine generelle Verwendung von π_B nicht sinnvoll ist, wird in der Diskussion (Abschnitt 5) erläutert.

Beim Vergleich der trainierten Policies zeigt sich, dass die Erfolgsrate (E in Tabelle 2) bei π_q mit 72,4 % deutlich geringer ist als bei π_R (86,5 %) und π_W (87,2 %). Der durchschnittliche Reward ($\bar{R}$) ist, auch aufgrund der verwendeten Rewardfunktion (s. Abschnitt 3.3), bei π_q (13,6) deutlich geringer als bei π_R (16,6) und π_W (16,8).

Bei der Betrachtung der Dialoglänge ($\bar{L}$) fällt auf, dass der Unterschied zwischen π_Q und π_R nur 0,3 beträgt, während der Unterschied zwischen π_R und π_W 1,5 Dialogschritte beträgt.

		Dialoglänge (L)					Reward (R)					
		$\bar{L}$	σ	$\tilde{L}$	min	max	$\bar{R}$	σ	$\tilde{R}$	min	max	E
π_Q	Q-Learning	12,8	4,01	12	3	17	13,6	9,47	19,2	-1,85	19,9	72,4 %
π_B	Regelbasiert	10,4	1,77	10	8	15	19,5	0,09	19,6	19,3	19,6	100 %
π_R	REINFORCE	12,5	4,01	13	3	17	16,6	7,17	19,4	-1,85	19,8	86,5 %
π_W	WoLF-PHC	11	4,68	10	3	17	16,8	7,03	19,4	-1,85	19,9	87,2 %

Tabelle 2 – In der Evaluation ermittelte Werte für Dialoglänge (L), Reward (R) und Dialogerfolgsrate (E): $\bar{L}$ und $\bar{R}$ = Mittelwert, σ = Standardabweichung, $\tilde{L}$ und $\tilde{R}$ = Median.

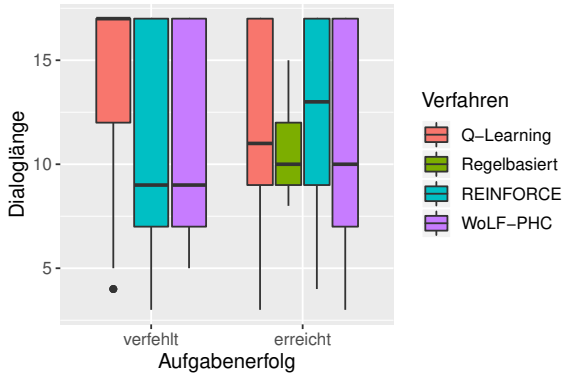


Abbildung 2 – Boxplot der Dialoglängen gruppiert nach dem Aufgabenerfolg (verfehlt oder erreicht)

Dies weißt daraufhin, dass das Training mit REINFORCE und WoLF-PHC zwar zu sehr ähnlichen Erfolgsraten (0,7 Prozentpunkte Unterschied) führt, aber WoLF-PHC im Mittel kürzere Dialoge erzeugt. Der nach dem Aufgabenerfolg gruppierte Boxplot in Abbildung 2 zeigt, dass der Unterschied in der Dialoglänge in den erfolgreichen Dialogverläufen begründet ist.

5 Diskussion

In unserem Beitrag haben wir untersucht wie sich 3 verschiedene Methoden des Reinforcement Learnings beim Trainieren einer Policy für einen Dialogmanager verhalten. Es war nicht unser Ziel ein tatsächliches Dialogsystem zu realisieren, sondern anhand einer bekannten und plausiblen Anwendungsdomäne das Lernrverhalten der Verfahren als auch die damit trainierten Policies zu evaluieren. Dazu wurde die Interaktion zwischen einem simulierten Nutzer und dem Dialogmanager auf der Ebene von Dialogakten realisiert.

Die verwendete Implementierung des PLATO Frameworks (Stand Dezember 2019) hat noch die Einschränkung, dass Nutzer und Dialogmanager jeweils nur einen Dialogakt mit einem adressierten Slot senden und empfangen können. Dies ist jedoch ausreichend, um das Verhalten beim Lernen der Policies zu beobachten.

Die Ergebnisse zeigen, dass die mit Q-Learning trainierte Policy weniger Reward und geringen Aufgabenerfolg als WoLF-PHC und REINFORCE aufweist. WoLF-PHC und REINFORCE erzielen fast gleichen Aufgabenerfolg, jedoch realisiert WoLF-PHC kürzere Dialoge. REINFORCE kann bei unbekannten Dialogzuständen verallgemeinern, d. h. Wahrscheinlichkeiten für die beste Aktion ausgeben, während WoLF-PHC und Q-Learning keine Abschätzung vornehmen können. Allerdings lernt WoLF-PHC effizienter die optimale Aktion für im Training selten gesehener Dialogzustände als Q-Learning.

Die als Baseline verwendete und rein regelbasierte Policy π_B ist für das hier verwendete Szenario und auf die PLATO Implementierung optimal angepasst, um als Warm-up Policy beim Training genutzt werden zu können. Sie ist allerdings nicht verallgemeinerbar, da sie voraussetzt, dass der Dialogmanager und der Nutzer immer nur jeweils einen Dialogakt je Dialogschritt verwenden und keine Metakommunikation unterstützt wird.

6 Danksagung

Wir bedanken uns bei dem 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] HILLMANN, S., K.-P. ENGELBRECHT, und B. WEISS: *Semi-automatische Generierung und Reinforcement Learning basiertes Training eines Dialogmanagers*. In *ESSV 2019*, Bd. 93 d. Reihe *Studientexte Zur Sprachkommunikation*, S. 31–41. Dresden, 2019.

[2] DUAN, Y., X. CHEN, R. HOUTHOOFT, J. SCHULMAN, und P. ABBEEL: *Benchmarking Deep Reinforcement Learning for Continuous Control*. In *Proc. 33rd Int. Conf. on Machine Learning*, ICML'16, S. 1329–1338. 2016.

[3] PAPANGELIS, A., Y.-C. WANG, P. MOLINO, und G. TUR: *Collaborative Multi-Agent Dialogue Model Training Via Reinforcement Learning*. In *Proc. SIGDIAL 2019*, S. 92–102. Stockholm, Sweden, 2019.

[4] WILLIAMS, R. J.: *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. *Mach. Learn.*, 8(3–4), S. 229–256, 1992. doi:10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

[5] SCHATZMANN, J., B. THOMSON, K. WEILHAMMER, H. YE, und S. YOUNG: *Agenda-based user simulation for bootstrapping a POMDP dialogue system*. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, S. 149–152. Association for Computational Linguistics, Rochester, New York, 2007. URL `https://www.aclweb.org/anthology/N07-2038`.

[6] WEISZ, G., P. BUDZIANOWSKI, P.-H. SU, und M. GASIC: *Sample efficient deep reinforcement learning for dialogue systems with large action spaces*. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 26(11), S. 2083–2097, 2018. doi:10.1109/TASLP.2018.2851664. URL `https://doi.org/10.1109/TASLP.2018.2851664`.

[7] YOUNG, S., B. THOMSON, J. D. WILLIAMS, und ET AL.: *Pomdp-based statistical spoken dialogue systems: a review*. In *PROC IEEE*. 2013.

[8] SUTTON, R. S. und A. G. BARTO: *Reinforcement Learning: An Introduction*. The MIT Press, second edn., 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[9] BOWLING, M. und M. VELOSO: *Rational and convergent learning in stochastic games*. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, S. 1021–1026. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[10] MICHAEL, T., S. HILLMANN, und B. WEISS: *An Artifical Conversational Agent for Students at the TU Berlin*. In *ESSV 2017*, Bd. 86 d. Reihe *Studientexte Zur Sprachkommunikation*, S. 238–245. Saarbrücken, 2017.

[11] PASZKE, A., S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, A. DESMAISON, A. KOPF, E. YANG, Z. DEVITO, M. RAISON, A. TEJANI, S. CHILAMKURTHY, B. STEINER, L. FANG, J. BAI, und S. CHINTALA: *Pytorch: An imperative style, high-performance deep learning library*. In H. WALLACH, H. LAROCHELLE, A. BEYGELZIMER, F. D' ALCHÉ-BUC, E. FOX, und R. GARNETT (Hrsg.), *Advances in Neural Information Processing Systems 32*, S. 8024–8035. Curran Associates, Inc., 2019.