SIMULATION VON NUTZERVERHALTEN BEI DER INTERAKTION MIT SPRACHDIALOGSYSTEMEN MITTELS KÜNSTLICHER NEURONALER NETZWERKE

Stefan Hillmann[1], *Benjamin Weiss*[1], *Thilo Michael*[1], *Sebastian Möller*[1]

[1]*Technische Universität Berlin*
stefan.hillmann@tu-berlin.de

Kurzfassung: Die Simulation von Nutzerverhalten wird überwiegend für das Training und die Evaluation von Sprachdialogsystemen verwendet. Insbesondere wenn ein Dialogmanager evaluiert oder trainiert werden soll, bietet sich eine Nutzersimulation auf der Ebene von Dialogakten und Konzepten an. Dazu untersuchen wir ein feedforward Netzwerk und 3 Varianten eines rekurrenten Netzwerks (RNN) auf ihre Fähigkeit Nutzerakte und Konzepte im BoRIS-Korpus zu schätzen. Für die Schätzung von Konzepten zeigen sich Vorteile bei RNN. Der Beitrag zeigt zudem wie neuronale Netze als Basis von Nutzersimulation verwendet werden können.

1 Einführung

Bei der Entwicklung von Sprachdialogsystemen (SDS) wird die Simulation von Nutzerverhalten im Wesentlichen für zwei Zwecke verwendet, zum einen für das Training von Dialogmanagern, die auf statistischen Ansätzen oder maschinellem Lernen beruhen. Zum anderen können solche Simulationen für die Evaluierung von Mensch-Computer-Interaktionen verwendet werden [1, 2]. Zur Evaluation werden Dialoge zwischen simulierten Nutzer und SDS, meist auf der Ebene der Intention, generiert und anschließend analysiert. Intention meint, dass der Dialog nicht in natürlicher Sprache, sondern abstrakt mittels Dialogakten und ausgetauschten Informationen abgebildet wird (z.B. adressierte Konzepte und entsprechende Werte). In dieser Arbeit untersuchen wir, inwieweit sich neuronale Netze (feedforward und rekurrent) als Basis für eine Nutzersimulation auf Konzeptebene eignen.

Zunächst gibt Abschnitt 2 einen kurzen Überblick zu verbreiteten Simulationsansätzen. In Abschnitt 3 beschreiben wir unsere Modelle und in Abschnitt 4 deren Evaluation sowie die Evaluationsergebnisse. Eine Diskussion der Ergebnisse findet sich in Abschnitt 5.

2 Ansätze zur Nutzersimulation

Zu den ersten Ansätzen in der Nutzersimulation zählen Bi-Grammmodelle [3] (es werden Tupel von System- und Nutzeräußerungen auf Konzeptebene betrachtet), die zu immer komplexeren (z. B. [4, 5]), aber dennoch stets probabilistischen oder regelbasierten Simulationen weiterentwickelt wurden. Die darauffolgende Generation von Nutzersimulationen verwendet Ansätze des maschinellen Lernens, wie bspw. Bayessche Netze [6], Hidden Markov Models [7] oder Reinforcement Learning [8, Kap. 3].

Die auch zum maschinellen Lernen zählenden neuronalen Netze haben seit den ca. 2010er Jahren eine große Verbreitung gefunden, da die Verfügbarkeit notwendiger Rechenkapazitäten und die Entwicklung effektiver Trainingsmethoden zusammenfielen. Im Kontext von SDS bilden künstliche neuronale Netze (KNN) heute den Stand der Wissenschaft im Bereich der Spracherkennung und des Sprachverstehens. Auch für die Modellierung von Dialogverläufen

Feature	Label
Dialogakte System	bye, explConfirm, indicateValues, indicateValues1, indicateValues2, inform, informAndOfferMore, offerModification, offerRefinement, repetitionRequest, request
Dialogakte Nutzer	accept, affirm, negate, neglect, no_value, provide
Konzepte System	ALL, bye, errorMessage
Konzepte Nutzer	action
Konzepte Beide	date, field, foodtype, localization, logical, no_value, price, time

Tabelle 1 – Im BoRIS Korpus verwendete Label für Dialogakte und Konzepte.

		Akt		Konzepte	
	Äußerung	Label	Kodierung	Label	Kodierung
System	Sie möchten also in der Innenstadt asiatisch essen gehen?	explConfirm	$a_s = 01000000000$	no_value	$k_s = 00000000010$
Nutzer	Ja, am Samstag Abend.	affirm provide	$a_n = 00000010$	date time logical	$k_n = 010100100$

Tabelle 2 – Beispiel für die Kodierung eines Dialogschritts. Die Bedeutung von a_s, a_n, k_s und k_n ist in Abschnitt 3.1 beschrieben.

bilden sie eine interessante Möglichkeit, da KNN prinzipiell dazu geeignet sind, die im Dialogverlauf und dem gegebenen Kontext implizit enthaltenen Informationen abzubilden. Dies gilt vor allem für Rekurrente Neuronale Netze (RNN) und LSTM-RNN (Long Short-Term Memory RNN) [9], welche die über die Zeit abgebildeten Einflussgrößen auf den Dialogverlauf (z.B. häufige Rückfragen durch das System) erlernen können [10, 11].

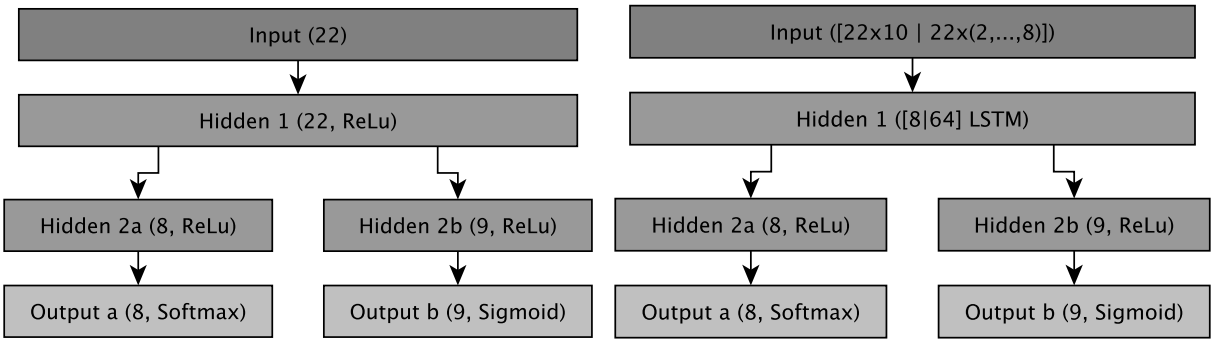
3 Daten und Modelle

Dieser Abschnitt stellt zum einen kurz das für das Training und die Evaluation verwendete annotierte Dialogkorpus vor, und zum anderen die Architektur der vier evaluierten Netzwerkvarianten sowie die verwendeten Dialogfeatures. Zum Abschluss wird die, auf einem probabilistischen Ansatz beruhende, Baseline erläutert.

3.1 BoRIS Korpus und Featurekodierung

Das genutzte Dialogkorpus stammt aus empirischen Versuchen mit dem Bochumer Restaurant Informationssystem (BoRIS) [12, S. 241–244]. Die 196 verwendeten Dialoge wurden im Rahmen einer von Möller 2009 beschriebenen Arbeit [12, S. 137 ff.] erhoben. Für unsere Untersuchung wurden die in dem Korpus annotierten Dialogakte und Konzepte verwendet, deren Label für System und Nutzer in Tabelle 1 gegeben sind. Die Konzepte in der letzten Zeile der Tabelle werden sowohl in durch Nutzer als auch System verwendet. Im Wesentlichen können die Konzepte als die in BoRIS adressierbaren Slots angesehen werden, weitergehende Beschreibungen zu den annotierten Dialogakten und Konzepten werden in [1] und [2, S. 150–151] gegeben.

Tabelle 2 zeigt anhand eines Beispiels aus dem BoRIS-Korpus die Kodierung der Features, die in den nachfolgend beschriebenen Modellen verwendet wurden. In jedem Dialogschritt



(a) Feedforward-Netzwerk

(b) Rekurrentes Netzwerk (Varianten mit 8 bzw. 64 LSTM-Zellen ersten Hidden Layer)

Abbildung 1 – Schematische Gegenüberstellung der verwendeten Netzwerke. In jeder Schicht sind Klammern jeweils die Anzahl der Neuronen und die Aktivierungsfunktion angegeben.

hat zuerst das System eine Äußerung generiert und der Nutzer danach geantwortet. Alle Äußerungen sind mit den in Tabelle 1 gezeigten Labeln annotiert. Eine Systemäußerung hat immer genau ein Aktlabel, während eine Nutzeräußerung mit ein oder zwei Labeln annotiert ist. Neben den 6 einzelnen Labeln für Akte des Nutzers in Tabelle 1, treten auch die Kombinationen *accept provide* und *affirm provide* auf. Sowohl die 11 Systemakttypen als auch die insgesamt 8 Nutzerakttypen wurden one-hot kodiert (genau ein Wert in dem 11- bzw. 8-stelligen Vektor ist 1) während die verwendeten Konzepte, bei System und Nutzer, in eine binäre Kodierung überführt wurden. Dort kann, entsprechen der Anzahl der verwendeten Konzepte, an mehreren Stellen des 11- (System) oder 9-stelligen (Nutzer) Vektors der Wert 1 auftreten. Die Variablen a_s, a_n, k_s sowie k_n werden im folgenden als Abkürzungen genutzt, um die kodierenden Vektoren für Akte (a) und Konzepte (k) von System (s) und Nutzer (n) zu referenzieren.

3.2 Modellierungsansätze

Im Folgenden werden die vier neuronalen Netze und ein rein probabilistischer Ansatz vorgestellt, die alle den Zweck haben, aus einem bekannten Systemakt und den dabei durch das System verwendeten Konzepten, die darauf folgende Nutzerreaktion in Form von Nutzerakt und verwendeten Konzepten zu schätzen. Es soll also mit einer Funktion h geschätzt werden, welches Tupel $(\hat{a}_n, \hat{k}_n)$ auf das Tupel (a_s, k_s) folgt: $h(a_s, k_s) = (\hat{a}_n, \hat{k}_n)$.

Um die Generalisierbarbeit der Ansätze zu erhöhen, wurde bewusst darauf verzichtet die konkreten Werte, die zu einem Konzept genannt wurden, zu modellieren. Dies bedeutet für das Beispiel aus Tabelle 2, dass die Nutzeräußerung “Ja, am Samstag Abend.” äquivalent zu der Äußerung “Ja, am Montag Morgen.” kodiert wird.

3.2.1 Grundlegende Struktur der Neuralen Netze

Die vier in diesem Abschnitt beschriebenen Ansätze beruhen jeweils auf einem neuronalen Netzwerk mit jeweils einer einheitlichen Eingabeschicht und einer geteilten Ausgabeschicht. Das bedeutet, die Ausgabeschicht wurden jeweils in die zwei Abschnitte *Output a* und *Output b* unterteilt, die entsprechenden Schemata sind in Abbildung 1 gezeigt. In jeder Variante repräsentiert *Output a* den durch das Netzwerk klassifizierten Dialogakt des Nutzers, während *Output b* je Neuron ein durch den Nutzer adressiertes Konzept repräsentiert. Nachstehend werden nun die beiden Teile der Ausgabeschicht (hier auch insbesondere die Aktivierungsfunktionen und die Interpretation bei der Klassifizierung) beschrieben und im Anschluss, für jedes Netzwerk, die Eingabeschicht und die ihr nachfolgenden Schicht (Hidden 1 in Abbildung 1). Architektonisch unterscheiden sich die Netze ausschließlich in diesen beiden Schichten.

Output a repräsentiert den durch das Netzwerk geschätzten Nutzerakt und arbeitet als multi-class-Klassifikator (d. h. 1-aus-n), der genau einen der 6 einzelnen und 2 kombinierten Nutzerakte (s. Abschnitt 3.1) ausgibt. Die Neuronen in *Output a* nutzen Rectifier [13] als Aktivierungsfunktion (rectified linear unit, kurz ReLU). Sowohl beim Training, als auch bei der Vorhersage (Klassifikation), wird auf alle Neuronen zusätzlich die Softmax-Funktion angewendet, womit die Aktivierungen der Neuronen als relative Wahrscheinlichkeiten interpretiert werden können, da die Summe der Werte in *Output a* sich zu 1 aufaddieren lässt. Für die Bestimmung des entsprechenden Vektors a_n (Nutzerakt), wird die Stelle in a_n auf 1 gesetzt die dem Neuron mit der höchsten Aktivierung in *Output a* entspricht. Das heißt konkret, dass auf *Output a* die *arg max* Funktion angewendet wird. Alle andere Stellen in a_n werden auf 0 gesetzt, wodurch der klassifizierte Nutzerakt eindeutig kodiert ist (vgl. Abschnitt 3.1 und Tabelle 2).

Output b fungiert nicht als multi-class-Klassifikator, sondern als multi-lable-Klassifikator (m-aus-n). Da der Nutzer prinzipiell beliebige Kombinationen von Konzepten in einem Dialogschritt adressieren kann, wären bei der Nutzung eines 1-aus-n Klassifikators 511 Neuronen in *Output b* notwendig gewesen ($2^9 - 1$ Kombinationen in der binären Kodierung). Eine Sigmoidfunktion wird in den 9 Neuronen von *Output b* als Aktivierungsfunktion verwendet. Die jeweilige Aktivierung kann als Maß für die Wahrscheinlichkeit angesehen werden, dass das mit dem Neuron korrespondierende Konzept in der geschätzten Nutzeräußerung verwendet wird. Die Aktivierung kann theoretisch in dem Intervall $(-1,1)$ liegen, wobei in der Praxis Werte zwischen 0 und 1 vorkommen. Zur Bestimmung des binären Vektors k_n wird der Schwellwert λ verwendet. Für jede Aktivierung $\hat{\theta}_i$ in einem der Neuronen in *Output b*, für die $\hat{\theta}_i \geq \lambda$ gilt, wird die entsprechende Stelle in k_n auf 1 gesetzt und alle anderen Stelle auf den Wert 0 (s. Gleichung 1). Im Folgenden haben wir stets mit $\lambda = 0,5$ gearbeitet.

$$k_n^i = \begin{cases} 1, & \text{wenn } \hat{\theta}_i \geq \lambda \\ 0, & \text{wenn } \hat{\theta}_i < \lambda \end{cases} \tag{1}$$

Alle in dieser Arbeit vorgestellten Netzwerke wurden unter Verwendung von Keras 2.1.1 (`github.com/fchollet/keras`) mit Tensorflow 1.4.0 (in der GPU-fähiger Variante, `www.tensorflow.org`) als Backend definiert, trainiert und evaluiert. Als Laufzeitumgebung kam Python 3.5.2 (`www.python.org`) auf einem Ubuntu 16.04.3 LTS (`www.ubuntu.com`) basierten System zum Einsatz.

3.2.2 Feedforward-Netzwerk

Abbildung 1a zeigt die Struktur des verwendeten feedforward-Netzwerks (FFN im Folgenden). Der Eingabeschicht mit 22 Neuronen folgt direkt eine verborgene Schicht (engl. hidden layer) der gleichen Größe, hinter der sich das Netzwerk in zwei Stränge verzweigt und somit ein sogenanntes multi-ouput Netzwerk bildet. Jeder Strang enthält nochmals eine verborgene Schicht, deren Größe der jeweiligen Ausgabeschicht entspricht. Alle verborgenen Schichten nutzen ReLU zur Berechnung der Aktivierungen.

3.2.3 Rekurrente Netzwerke

Es wurden zwei Varianten eines rekurrenten Netzwerks (s. Abbildung 1b) evaluiert. Beide Varianten unterscheiden sich in der Definition der Eingabeschicht und dann nochmal jeweils individuell in der ersten verborgenen Schicht von dem FFN aus dem vorherigen Abschnitt. Die Eingabeschicht kann in den rekurrenten Varianten Zeitreihen der Länge 10 verarbeiten, d. h. Sequenzen von bis zu 10 Dialogschritten können zur Schätzung des nächsten Nutzakts und der dabei verwendeten Konzepte berücksichtigt werden.

Zeitreihen		Zeitreihen (fortg.)	
Nr.	Sequenz	Nr.	Sequenz
1	$\_, \_, s_1$	4	s_2, s_3, s_4
2	$\_, s_1, s_2$	5	s_3, s_4, s_5
3	s_1, s_2, s_3		

(a) Beispiel für die Erzeugung von Zeitreihen für die Fenstergröße $w = 3$ und dem Dialog s_1, s_2, s_3, s_4, s_5 (5 Dialogschritte).

Variante	w	*#LSTM*
FFN	1	–
RNN_8^{10}	10	8
RNN_{64}^{10}	10	64
RNN_{64}^{2-8}	2, 3, 4, 5, 6, 7, 8	64

(b) Übersicht zur Anzahl von LSTM Zellen und Fenstergröße (w) in den verschiedenen Varianten.

Tabelle 3 – Ein Beipiel zur Erzeugung von Zeitreihen (a) sowie die Parameter der evaluierten Netzwerke (b).

Die erste verborgene Schicht der einen Variante (RNN_8^{10} im Folgenden) setzt sich aus 8 LSTM Zellen zusammen, während die zweite Variante (RNN_{64}^{10} im Folgenden) dort 64 LSTM Zellen verwendet.

Eine weitere Evaluation fand unter Verwendung von RNN_{64}^{10} statt, jedoch mit Zeitreihen der Länge 2 bis 8. Diese Bedingung wird im Folgenden als RNN_{64}^{2-8} bezeichnet. Tabelle 3b gibt einen Überblick zu den beschriebenen Varianten. Die Unterschieden bei der Aufbereitung der Dialogdaten, hinsichtlich der Zeitreihenlängen, werden im nachfolgenden Abschnitt erläutert.

3.2.4 Fenstergröße

Die Methode zum Zerlegen eines Dialogs in Teilsequenzen war für alle Netzwerke die gleiche, lediglich die verwendete Fenstergröße (w), mit der über den jeweiligen Dialog (modelliert als Abfolge von Dialogschritten) geglitten wurde, wurde variiert. Tabelle 3b gibt dazu eine Übersicht.

Das in Tabelle 3a gezeigte Beispiel für $w = 3$ illustriert das Vorgehen bei der Erzeugung der Zeitreihen, und nachfolgend wird der verwendete Algorithmus kurz beschrieben. Das Dialogschrittfenster gleitet Schritt für Schritt von links (Anfang des Dialogs) nach rechts (Ende des Dialogs) über die Sequenz der Dialogschritte. Dabei wird das Fenster von rechts aufgefüllt und es wird mit dem ersten Dialogschritt begonnen. Die noch freien Stellen des Fensters werden mittels links-padding aufgefüllt (_ wurde als Pad verwendet). Das Padding ist für die ersten $w-1$ Dialogschritte notwendig. Danach ist das Fenster stets komplett gefüllt und wandert über die Sequenz bis der letzte Dialogschritt erreicht ist. Ein right-padding wurde nicht durchgeführt, da die rechte Fensterseite nicht über den letzten Schritt hinausgeschoben wurde. Jede, in einem Fensterschritt erzeugte Teilsequenz wurde später für Training und Evaluation verwendet.

Für das FFN liegt ein Spezialfall vor, da hier $w = 1$ gilt. Die Dialoge wurde also faktisch in ihre einzelnen Schritte zerlegt, ein Padding war nicht notwendig. Im Fall von RNN_{64}^{2-8} wurden Zeitreihen für $w \in \{1,2,3,4,5,6,7,8\}$ erzeugt, mit der Absicht die Verallgemeinerbarkeit des damit trainierten Netzwerks zu erhöhen.

3.3 Baseline

Die Baseline (*BL*) orientiert sich an der von Eckert 1997 [3] beschriebenen stochastischen Modellierung des Nutzerverhaltens. Die Aktion des Nutzer wird dabei als Wahrscheinlichkeitsfunktion modelliert, die sich nur auf den vorhergehenden Akt des System bezieht und somit zeitinvariant ist. Wie auch bei der Modellierung der neuronalen Netze wurde die *BL* in zwei Abschnitte unterteilt. Eine erste Funktion h_1 schätzt die Akte des Nutzers (a_n) und eine zweite Funktion h_2 schätzt die durch den Nutzer verwendeten Konzepte (k_n). Dabei schätze h_1 den Nutzerakt aus den Akten und Konzepten des Systems: $h_1(a_s, k_s) = \hat{a_n}$, während h_2 die Konzepte des Nutzer aus dem Akt und Konzepten des Systems sowie dem durch h_1 gegebenen Nutzerakts

schätzt: $h_2(a_s, k_s, \hat{a_n}) = \hat{k_n}$.

Für das Modell h_1 wird für jeden möglichen Nutzerakt a_n^i die bedingte Wahrscheinlichkeit $p(a_n^i) = P(a_n^i | a_s, k_s)$ aus einem Set an Trainingsdaten berechnet. Beim Schätzen des Nutzeraktes wird dann das Maximum aus den errechneten Wahrscheinlichkeiten genutzt, um den wahrscheinlichsten Nutzerakt zu bestimmen: $argmax_i(p(a_n^i)) = \hat{a_n}$. Für den Fall, dass keine Daten für eine Kombination aus (a_s, k_s) vorliegen, wird die Klassifizierung als fehlerhaft gewertet.

Für das Modell h_2 wird für jedes mögliche Konzept des Nutzers k_n^i die bedingte Wahrscheinlichkeit $p(k_n^i) = P(k_n^i | a_s, k_s, a_n)$ aus dem gleichen Set an Trainingsdaten berechnet. Beim Schätzen der Konzepte wird dann jedes Konzept mit einer Wahrscheinlichkeit von mindestens $0,5$ in der Schätzung verwendet. Dies wird analog zu der Berechnung in Gleichung 1 berechnet, nur dass statt θ_i die Wahrscheinlichkeit $p(k_n^i)$ verwendet wird. Für den Fall dass keine Daten für eine Kombination aus (a_s, k_s, a_n) vorliegen wird, im Rahmen des Modells, angenommen, dass kein Konzept vom Nutzer genannt wurde.

4 Evaluation der Modelle

Dieser Abschnitt beschreibt zunächst das Evaluierungsverfahren, das auf die oben beschriebenen Modelle angewendet wurde, sowie die verwendeten Evaluationsmaße. Danach werden die, für die einzelnen Maße, erzielten Werte einander gegenüber gestellt und erläutert.

4.1 Kreuzvalidierung und Performanzmaße

Alle im vorherigen Abschnitt vorgestellten Modelle und ihre Varianten wurden jeweils im Rahmen einer 10-fachen Kreuzvalidierung evaluiert. Für die neuronalen Netze mit einer festen Fenstergröße (s. Tabelle 3b), wie auch die Baseline (*BL*), standen jeweils 2001 Datensätze (einzelne Dialogschritte oder Zeitreihen) zur Verfügung. Für die Variante RNN_{64}^{2-8} waren es $2001 * 7 = 14.007$ Zeitreihen. In jedem Durchlauf der Kreuzvalidierung wurden 90 % der Daten für das Training und 10 % für die Evaluation verwendet, wobei jeder Datensatz neunmal für das Training und einmal für die Evaluation genutzt wurde. Die Batchgröße (batch size) betrug für Training und Evaluation stets 1.

In den, nach der Beschreibung in Abschnitt 3.2.4 erzeugten, Datensätzen wurde vor der Evaluation die Reihenfolge der erzeugten Zeitreihen (ohne einzelne Zeitreihen zu ändern) zufällig verändert. Für die Varianten RNN_8^{10} und RNN_{64}^{10} waren die in der Reihenfolge geänderten Datensätze identisch.

Zur Bewertung der Performanz der einzelnen Modelle, hinsichtlich ihrer Fähigkeit aus einem Systemakt und den dabei verwendeten Konzepten den darauf folgenden Nutzerakt (inklusiver der genutzten Konzepte) zu schätzen, wurden die Modelle anhand der Genauigkeit (s. Gleichung 2), Cohens Kappa (κ) [14] und dem Hamming Loss (η) [15] verglichen. Die Maße wurden jeweils für einen Durchlauf der Kreuzvalidierung berechnet und im Rahmen der Auswertung gemittelt.

$$\text{Genauigkeit} = \frac{\text{Anzahl korrekte Schätzungen}}{\text{Anzahl aller Schätzungen}} \tag{2}$$

Neben der Genauigkeit wurde κ auf die geschätzten Nutzerakte angewendet, da die Klassen (Nutzerakttypen) starke Unterschiede in ihren Häufigkeiten aufweisen. Da die Genauigkeit auf absolute Übereinstimmung testet, wurde η verwendet um den Multi-Label-Klassifikator zum Schätzen der Nutzerkonzepte weitergehend zu analysieren. Hier gibt der Hamming Loss Auskunft darüber, ob zumindest ein Teil der geschätzten Konzepte (z. B. eins von zwei geschätzten) pro Dialogakt korrekt war.

Modell	Genauigkeit			κ	η	η_1	η_2	η_3	η_4
	Akt	Konzepte	Kombiniert						
BL	0,945	**0,852**	0,817	0,899	**0,148**	**0,073**	0,892	0,979	1
FFN	0,945	0,829	0,812	0,902	0,171	0,098	0,858	1	1
RNN_8^{10}	0,948	0,817	0,802	0,906	0,183	0,115	0,703	1	0,983
RNN_{64}^{10}	0,958	0,835	0,824	0,925	0,165	0,098	0,718	0,963	**0,933**
RNN_{64}^{2-8}	**0,960**	0,840	**0,828**	**0,930**	0,160	0,096	**0,567**	**0,960**	0,944

Tabelle 4 – Evaluationsergebnisse für die Baseline und die vier Netzwerkvarianten. Es sind jeweils die Mittelwerte nach einer 10-fachen Kreuzvalidierung angegeben. Cohens Kappa (κ) ist für die klassifizierten Dialogakte angegeben während sich der Hamming Loss (η) auf das automatische Labeln von Konzepten bezieht.

4.2 Ergebnisse

Die mittels der einer 10-fachen Kreuzvalidierung, die im vorherigen Abschnitt beschrieben wurde, gewonnen Ergebnisse werden in diesem Abschnitt erläutert. In Tabelle 4 sind die Mittelwerte, gerechnet über die 10 Durchläufe der Kreuzvalidierung, für die gemessenen Performanzmaße (s. vorheriger Abschnitt) für jedes Modell aufgelistet. Der für ein Maß (d.h. innerhalb einer Tabellenspalte) jeweils beste Wert ist in Tabelle 4 fett markiert.

Insgesamt zeigt sich, dass für ein Maß entweder *BL* oder RNN_{64}^{2-8} den jeweils besten Wert erreicht. Bezogen auf die Genauigkeit bei der Kombination aus Nutzerakt und Konzepten (beides wurde korrekt für einen Dialogakt geschätzt) ist RNN_{64}^{2-8} minimal im Vorteil (ca. 1 %). Für κ (gemessen für die geschätzten Nutzerakte) beträgt der Unterschied zwischen *BL* und RNN_{64}^{2-8} ca. 3 %. Werte von $\kappa > 0,8$ werden beim Vergleich menschlicher Annotatoren als eine sehr gute Übereinstimmung gedeutet.

Wie die Genauigkeit, liegt η immer im Intervall $[0,1]$. Da allerdings mit η ein Unterschied gemessen wird, ist hier ein kleinerer Wert besser. η wurde sowohl für alle Schätzungen von Konzepten berechnet, als auch für Dialogschritte bei denen die Nutzer im BoRIS-Korpus 1, 2, 3 oder 4 Konzepte nannten (s. Spalten η_1 bis η_4 in Tabelle 4). Bei der Betrachtung aller Fälle (Spalte η) und der mit einem genannten Konzept (η_1) zeigt *BL* die besten Werte, wobei der Unterschied zu RNN_{64}^{2-8} beide Male ca. 2,3 % beträgt. Sehr auffällig ist das Ergebnis für η_2. Hier zeigt *BL* den höchsten und RNN_{64}^{2-8} den geringsten Wert (und damit einen geringeren Unterschied zu den Testdaten). Für RNN_{64}^{2-8} ist $\eta_2 = 0,567$, was zwar eine beträchtliche Abweichung von den Testdaten bedeutet, aber deutlich besser ist, als für die anderen rekurrenten Modelle (ca. 13 %), *BL* und *FFN* (ca. 30 %). Für η_3 und η_4 ist die Schätzleistung aller Modelle sehr schlecht.

5 Diskussion und Ausblick

Die Ergebnisse unserer Arbeit zeigen, dass neuronale Netzwerke auf dem von uns verwendeten BoRIS-Korpus Nutzerakte und -konzepte ähnlich gut schätzen können wie ein probabilistisches Modell (*BL*). Ein (leichte) Überlegenheit zeigt sich nur für das rekurrente Netz mit 64 LTSM-Zellen in Kombination mit gemischter Fenstergröße (RNN_{64}^{2-8}). Auffällig ist die deutlich bessere Performanz bei der Schätzung von zwei Konzepten mit RNN_{64}^{2-8} gegenüber den anderen Modellen. Ähnlich große Unterschiede lassen sich nicht in den anderen Maßen zeigen, jedoch war die Leistung der Baseline (*BL*) auch deutlich besser als erwartet.

Die Gründe für die geringen Unterschiede zwischen den konzeptionell unterschiedlichen Ansätzen (insb. probabilistisch vs. rekurrente Netze) haben vermutlich zwei wesentliche Gründe. Zum einen sind die Dialogverläufe im BoRIS-Korpus relativ deterministisch, was dem *BL*

Ansatz zugute kommt. Zum anderen erwarten wir von den neuronalen Netzen einen deutlicheren Vorteil, wenn eine höhere Felxibilität (d. h. Generalisierbarkeit) notwendig ist (größer Anteil an Daten für die Evaluation) und gleichzeitig mehr Datenmaterial für das Training verwendet werden kann. Dazu sollen die Modelle auf einem größeren Dialogkorpus getestet werden. Als zusätzliches Eingangsfeatures soll dann auch das Nutzerziel, ebenfalls auf Konzeptebene, berücksichtigt werden. Das Nutzerziel ist bei der Simulation auf Systemseite nicht verfügbar, aber durchaus bei Simulationen mit dem Zweck einen Dialogmanager zu evaluieren oder zu trainieren.

Literatur

[1] ENGELBRECHT, K.-P.: *Estimating Spoken Dialog System Quality with User Models*. Ph.D. thesis, Technische Universität Berlin, Berlin, 2011.

[2] HILLMANN, S.: *Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems*. Ph.D. thesis, Technische Universität Berlin, Berlin, 2017.

[3] ECKERT, W., E. LEVIN, und R. PIERACCINI: *User modeling for spoken dialogue system evaluation*. In *Proc of IEEE Automatic Speech Recognition and Understanding Workshop ASRU97*, S. 80–87. Ieee, 1997.

[4] HILLMANN, S. und K.-P. ENGELBRECHT: *Aufgabenmodellierung in der Simulation von Interaktionen mit Sprachdialogsystemen*. In P. WAGNER (Hrsg.), *Elektronische Sprachsignalverarbeitung 2013*, S. 20–27. TUDpress, 2013.

[5] SCHATZMANN, J., B. THOMSON, K. WEILHAMMER, H. YE, und S. YOUNG: *Agenda-based user simulation for bootstrapping a POMDP dialogue system*. In *NAACL-HLT 2007*, S. 149–152. 2007.

[6] PIETQUIN, O.: *A Framework for Unsupervised Learning of Dialogue Strategies*. Ph.D. thesis, Presses univ. de Louvain, 2004.

[7] 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.

[8] CUAYÁHUITL, H.: *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. thesis, 2009. URL `https://www.era.lib.ed.ac.uk/handle/1842/2750`.

[9] HOCHREITER, S. und J. SCHMIDHUBER: *Long Short-Term Memory*. *Neural Comput.*, 9(8), 1997.

[10] CROOK, P. und A. MARIN: *Sequence to Sequence Modeling for User Simulation in Dialog Systems*. S. 1706–1710. ISCA, 2017.

[11] ASRI, L. E., J. HE, und K. SULEMAN: *A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems*. In *Interspeech 2016*, S. 1151–1155. 2016.

[12] MÖLLER, S.: *Quality of Telephone-Based Spoken Dialogue Systems*. Kluwer Academic Publishers, Boston, 2005.

[13] HAHNLOSER, R. H. R., R. SARPESHKAR, M. A. MAHOWALD, R. J. DOUGLAS, und H. S. SEUNG: *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit*. *Nature*, 405(6789), S. 947–951, 2000.

[14] COHEN, J.: *A Coefficient of Agreement for Nominal Scales*. *Educational and Psychological Measurement*, 20(1), S. 37–46, 1960.

[15] SOKOLOVA, M. und G. LAPALME: *A systematic analysis of performance measures for classification tasks*. *Information Processing & Management*, 45(4), S. 427–437, 2009.