Steadforce GmbH
Westendstraße 193
80686 München

STEADFORCE

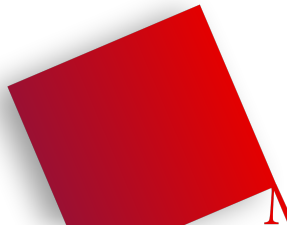
Masterarbeit
Datenaugmentierung und der Vergleich verschiedener Lernvarianten im Dialog Management von Gesprächsassistenten

Dominic Karnehm
Matrikelnummer: 48365015

21. September 2020

Prüfer:
Prof. Dr. David Spieler
Zweitprüfer:
Prof. Dr. Alfred Nischwitz

Betreuer:
Stephan Schiffner
Dr. Jonathan Boidol
Simon Schmitz



HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

Eidesstattliche Erklärung

Ich, Dominic Karnehm, versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Unterschrift:

_____

Ort und Datum:

_____

Zusammenfassung

Ein Chatbot oder ein Sprachassistent, welcher auf eine gestellte Frage unzusammenhängend antwortet, kann zu einer Abkehr des Benutzers von den Diensten des Unternehmens führen. Erhält der Anwender eine Liste an Hotels auf die Anfrage nach einem Zug in die nächste Stadt, wird dieser den Gesprächsassistenten womöglich nicht weiter nutzen oder gegebenenfalls zu einem Konkurrenzprodukt wechseln. Neben einer fehlerhaften Erkennung der Intentionen des Benutzers ist ein inkorrektes Dialog Management ein möglicher Grund für unzutreffende Reaktionen des Gesprächsassistenten.

In der hier vorliegenden Masterarbeiten werden mit Vertauschung von Abschnitten des Dialogs, Einfügen von Chitchat und Generierung neuer Dialoge drei Algorithmen für die Datenaugmentierung vorgestellt. Das Training des dem Dialog Management zugrundeliegenden Modells wird mit den zwei Lernvarianten Supervised Learning und Reinforcement Learning realisiert und verglichen. Für das Supervised Learning wird das State of the Art Modell Transformer Embedding Dialogue (TED) [72] des Rasa Frameworks genutzt. Der Deep-Q-Learning Algorithmus [42] wird zur Umsetzung des Reinforcement Learnings angepasst und evaluiert. Um die Nutzbarkeit der verschiedenen Verfahren festzustellen, werden innerhalb der Evaluation automatisierte Tests durchgeführt, um Änderungen in der Genauigkeit des Dialog Management unter Anwendung der Lernvarianten und Augmentierungen zu überprüfen. Das Einfügen von Chitchat erweist sich in der Evaluation als vielversprechendstes der drei Augmentierungsverfahren. Es konnten Verbesserungen der Accuracy von bis zu 1,6% und im F_1 Maß von 1,8% im Supervised Learning erzielt werden. Deutliche Verbesserungen von bis zu 6,6% Accuracy und 7,4% F_1 Maß werden mit dieser Augmentierung im Reinforcement Learning erreicht. Vor- und Nachteile der Augmentierungen und Lernvarianten werden diskutiert. Abschließend werden zukünftige Entwicklungsmöglichkeiten diskutiert.

Abstract

A chatbot or a virtual assistant that answers a question in an inconsistent manner may lead to the user turning away from the company's services. If the user receives a list of hotels in response to a request for a train to the next city, he may not continue to use the assistant or may switch to a competing product. In addition to incorrect detection of the user's intentions, incorrect dialog management is a possible reason for incorrect reactions of the conversational assistant.

In this master thesis, three algorithms for data augmentation are presented: swapping sections of the dialog, inserting chitchat and generating new dialogs using a dialog graph. The training of the model on which dialog management is based is realized and compared with the two learning variants Supervised Learning and Reinforcement Learning. For the supervised learning the state of the art model Transformer Embedding Dialogue (TED) [72] of the Rasa Framework is used. The Deep-Q-Learning algorithm [42] is adapted and evaluated for the implementation of reinforcement learning. In order to determine the usability of the different procedures, automated tests are performed within the evaluation to check for changes in the accuracy of the dialog management using the learning variants and augmentations. In the evaluation, the insertion of chitchat proves to be the most promising of the three augmentation methods. Improvements in accuracy of up to 1.6% and in the F_1 score of 1.8% were achieved in supervised learning. Significant improvements of up to 6.6% accuracy and 7.4% F_1 score are achieved with this augmentation in reinforcement learning. The advantages and disadvantages of augmentation and learning variants are discussed. Finally, future development possibilities are discussed.

Inhaltsverzeichnis

1 Motivation

»Gott sprach: Es werde Licht. Und es wurde Licht.« So heißt es in Gen 1,3 [4]. Wäre die Bibel ein Werk der heutigen Zeit, würde es eher lauten "Gott sprach: 'Alexa mach das Licht an'. Und Alexa antwortete: 'Ok Gott. Ich habe das Licht angemacht'". Sprachassistenten oder Chatbots zu nutzen für einfache Befehle wie das Steuern des Lichtes im Smarthome oder das Beantworten von leichten Fragen ist heute für eine breite Masse der Bevölkerung normal. »Die Beratungsfirma Deloitte schätzt, dass der Markt für künstlichen [sic] Intelligenzen, mit denen man sich unterhalten kann, bis zum Jahr 2024 auf mehr als 15 Milliarden US-Dollar wächst« [34] (Grund für sic: korrekte Schreibweise "künstliche"). Die Erstellung von Gesprächsassistenten mit denen komplexere Gespräche geführt werden können, stellt Entwickler auch heute noch vor eine Vielzahl von Problemen. Ein Chatbot, der aufgrund von Unverständnis nicht mehr korrekt auf Eingaben des Benutzers reagieren kann, frustriert den User ungemein. Unerwartete Zwischenfragen können beispielsweise zu solch einer Unstimmigkeit führen.

Ein Sachverhalt welcher in vielen Gebieten des maschinellen Lernens auftritt, ist die Beschaffung ausreichender Daten und die Beurteilung der Qualität dieser Daten. Dieses Problem stellt sich so auch im Bereich der Gesprächsassistenten. Ist es nicht möglich bereits vorhandene Daten zu nutzen, sind andere Mittel zu ergreifen. Bei Gesprächsassistenten handelt es sich bei diesen Daten beispielsweise um Dialoge, welche als Grundlage für das maschinelle Lernen dienen. Es kann möglich sein sich auf offen zugängliche Daten zu stützen. Eine weitere Möglichkeit ist die Erstellung neuer Dialoge. Ist auch dies aufgrund der Gegebenheiten nicht möglich oder ausreichend, können Dialoge mit Hilfe von verschiedenen Verfahren augmentiert werden. Das Cambridge Dictionary definiert Augmentierung als einen Prozess, um die Größe, den Wert oder die Qualität von etwas zu erhöhen, indem etwas hinzugefügt wird [11]. In dieser Arbeit wird Augmentierung als Vergrößerung der Menge und Änderung der ursprünglichen Daten angesehen, um so die Qualität der Daten für das maschinelle Lernen zu erhöhen.

Neben der zu geringen Anzahl an Trainingsdaten stellt der Wechsel zwischen unterschiedlichen Kontexten, wie es bei menschlicher Kommunikation üblich ist, die Entwickler von solchen Systemen vor Probleme. Für Menschen ist es völlig normal, innerhalb eines Gesprächs zwischen verschiedenen Themen zu springen. Solche Wendungen stellen jedoch Gesprächsassistenten vor Probleme. Es ist notwendig, diesen Wechsel zu erkennen und den korrekten Bezug zu dem jeweiligen Kontext herzustellen, um eine passende Reaktion durch den Gesprächsassistenten sicherzustellen. Hierbei gibt es verschiedene Alternativen dies umzusetzen. Eine Möglichkeit ist die Anpassung der Algorithmen oder der Lernmethoden auf diese Problemstellung. Auch besteht die Alternative, schon in den Trainingsdaten solche Zwischenfragen oder Themenwechsel einzubauen, um diese bereits von Anfang an vorzusehen [79].

2 Ziele der Arbeit

Durch die zunehmende Relevanz von digitalen Assistenten steigen gleichermaßen die Anforderungen an solche Systeme. So ist neben der Spracherkennung und der Erkennung der Intention des Benutzers auch das Dialog Management ein wichtiges Modul von Assistenzsystemen. Dialog Management wählt anhand der mit dem User bereits durchgeführten Kommunikation und der Intention des Benutzers, die nächsten durchzuführenden Aktionen. Die Auswahl kann algorithmisch oder mit Methoden des maschinellen Lernens erfolgen. Algorithmische Ansätze sind in dieser Arbeit nicht zu beachten. Verbesserungen bezüglich der Genauigkeit des Dialog Management können neben optimierten Architekturen auch durch eine höhere Qualität der Trainingsdaten erzielt werden. Um festzustellen, welche Rolle dabei die Augmentierung der Trainingsdaten spielen kann, werden im Rahmen dieser Arbeit mehrere Fragen betrachtet.

Explizit ausgeschlossen ist die Durchführung von Tests durch menschliche Benutzer. Alle durchzuführenden Tests erfolgen anhand zuvor definierter Testsets.

Um Aussagen zum Anwendungsgebiet von Augmentierungen zu treffen, soll geklärt werden,

- welche Augmentierungen im Dialog Management möglich sind,
- welche Lernvarianten im Dialog Management nutzbar sind,
- welche Auswirkungen die unterschiedlichen Augmentierungen auf die Ergebnisse der Lernvarianten aufweisen,
- inwieweit die betrachteten Augmentierungen zur Optimierung des Dialog Managements von Gesprächsassistenten führen und
- welche Vor- und Nachteile sich durch die Durchführung einer Augmentierung von Daten ergeben.

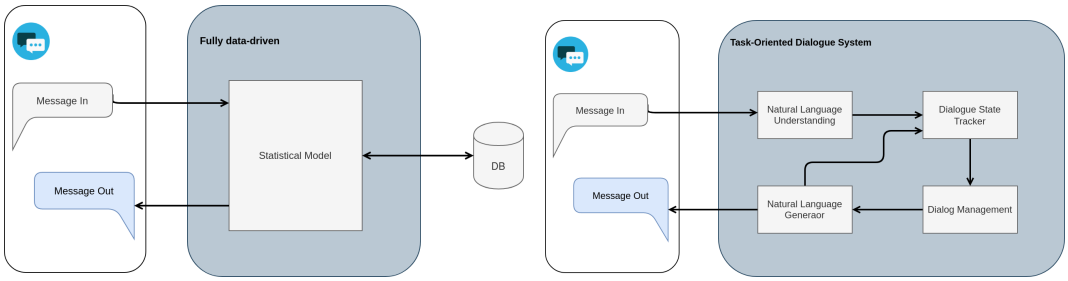
Hierzu wird zunächst im Kapitel 3 auf der nächsten Seite eine Einführung in das Dialog Management und die damit verbundenen Themenbereiche, wie Architekturen, Algorithmen und bereits umgesetzte Augmentierungen gegeben. Das Kapitel 4 auf Seite 18 stellt die verwendeten Korpora da, die Trainingsdaten für das Dialog Management erläutern. Drei Algorithmen der Augmentierung werden im Kapitel 5 auf Seite 20 eingeführt. Eine Beschreibung der genutzten Lernvarianten folgt im Kapitel 6. Eine Evaluierung der Effekte der Augmentierung auf die genutzten Lernvarianten erfolgt im Kapitel 7 auf Seite 34. Im Anschluss daran werden die Ergebnisse diskutiert und zum Abschluss ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

3 Stand der Technik

Um die in Kapitel 2 auf der vorherigen Seite beschriebenen Ziele der Arbeit zu diskutieren, sind zunächst vorangegangene Entwicklungen zu erwähnen. In diesem Kontext wird auf die Architektur moderner Dialogsysteme und deren Komponenten, schon verwendete Algorithmen des maschinellen Lernens und verschiedene Ausprägungen der Trainingsdaten eingegangen werden. Des Weiteren folgt eine Darstellung bereits diskutierter Varianten der Augmentierungen und eine Vorstellung von drei Frameworks aus dem Bereich der Dialogsysteme.

3.1 Architektur

Architekturen von Dialogsystemen lassen sich in zwei Kategorien aufteilen [18]. Es wird zwischen Aufgaben-orientierten und datengesteuerten Systemen unterschieden. Nach Aussage von [18] brachte ein Trend vollständig datengesteuerte Architekturen hervor, welche ein einzelnes neuronales Netz nutzen, um die Eingaben des Benutzers direkt auf Ausgaben des Gesprächsassistenten abzubilden. Eine Darstellung dieser Architektur ist der Abbildung 3.1a zu entnehmen.



(a) Datengesteuerte Dialoge nach [18] (b) Aufgaben-orientierter Dialog nach [38]

Abbildung 3.1: Zwei Architekturen für Dialogsysteme

Im Gegensatz hierzu beinhaltet die Aufgaben-orientierte Architektur mehrere miteinander interagierende Komponenten. Durch diese Aufteilung können spezialisierte Komponenten beispielsweise für das Sprachverständnis, das Dialog Management des Assistenten und die Sprachgenerierung unabhängig voneinander entwickelt und verbessert werden. Dadurch ist es möglich einzelne Bereiche eines Gesprächsassistenten an die äußeren Umstände anzupassen. Beispielsweise können so die Spracherkennung und Sprachgenerierung auf die jeweilige Sprache für den jeweiligen Zweck geändert werden, wobei das Dialog Management nicht notwendigerweise anzupassen ist. Außerdem können Fehlerquellen durch diese Strukturierung leichter ermittelt werden. Einen Nachteil ist der größere Entwicklungsaufwand, welcher aufgrund der höheren Anzahl an Komponenten anzunehmen ist. Die Abbildung 3.1b stellt eine Aufgaben-orientierte Architektur da, wie sie in [38] beschrieben wird. Diese Darstellung wurde gewählt, da sie alle Komponenten beinhaltet, die ebenso in vergleichbaren Implementierungen vorzufinden sind [33, 78, 5]. Abweichungen bei der Bezeichnung der Komponenten können jedoch festgestellt werden. So wird das Natural Language Understanding (NLU) auch als Natural Language Interpreter (NLI) bezeichnet. Überdies lassen sich nicht immer alle Komponenten in der identischen Ausprägung finden [33]. Die in der Architektur in Abbildung 3.1b dargestellten Komponenten sind:

- *Natural Language Understanding*
  Die Aufgabe des NLU besteht darin, Intentionen des Benutzers zu erkennen und die darin beinhalteten Informationen zu extrahieren und für die weitere Verarbeitung aufzubereiten.
- *Dialogue State Tracker*
  Der Dialogue State Tracker verfolgt die Historie des Gesprächs zwischen dem Nutzer und dem Gesprächsassistenten. Diese Komponente erhält die eingehenden Intentionen mit den dazugehörigen Informationen in Form von Entitäten des Benutzers und die ausgehenden Aktionen des Systems.
- *Dialog Management*
  Diese Komponente ermittelt anhand der durch die NLU erkannten Intention und der im Tracker hinterlegten Historie die nächsten zu tätigenden Aktionen des Gesprächsassistenten.
- *Natural Language Generator (NLG)*
  Der NLG erzeugt die Nachricht, die vom Gesprächsassistenten an den Benutzer gesendet wird.

Durch die hier beschriebene Architektur wird nicht festgeschrieben, mit welcher Technologie oder mit welchen Algorithmen die einzelnen Komponenten umgesetzt werden. So ist es beispielsweise möglich, das Dialog Management regelbasiert oder mit Mitteln des maschinellen Lernens umzusetzen. Auf Algorithmen aus dem Bereich des maschinellen Lernens, welche im Dialog Management Anwendung finden, wird im folgenden Abschnitt eingegangen.

3.2 Angewendete Algorithmen im Dialog Management

In diesem Abschnitt wird nicht näher auf regelbasiertes Dialog Management eingegangen. RavenClaw als beispielhaftes Framework, welches regelbasiertes Dialog Management umsetzt und die damit verbundene Vor- und Nachteile werden im Abschnitt 3.6.2 auf Seite 14 beschrieben. Dieser Abschnitt erläutert mit *Partially Observable Markov Decision Processes (POMDP)* und neuronalen Netzen zwei verschiedene Ausprägungen des maschinellen Lernens. Es wird darauf eingegangen, wo diese verwendet werden und welche Aspekte aus heutiger Sicht für oder gegen ihre Verwendung sprechen. Die Umsetzung mathematischer Methoden wird nicht beschrieben. Es folgt lediglich eine Erläuterung der Prinzipien der Algorithmen und Architekturen. Die verschiedenen Varianten des Lernens werden in Abschnitt 3.3 auf Seite 6 behandelt.

Partially Observable Markov Decision Processes (POMDP)

Eine Art von mathematischen Modell, das im Dialog Management genutzt wird ist Partially Observable Markov Decision Processes [66, 76, 75]. Bei POMDP handelt es sich um eine Generalisierung von Markov-Ketten [44]. Um POMDP zu erläutern ist zunächst ein Grundverständnis von Markov-Ketten zu schaffen. Grundlegende Begriffe und Funktionsweisen zu Markov-Ketten lassen sich anhand eines Beispiels erläutern. Hierbei wird das Wetter an einem Ort mittels einer Markov-Kette beschrieben [40]. Das Wetter an einem Tag kann sonnig, wolkig und regnerisch sein. Damit kann das Wetter genau drei sogenannte Zustände annehmen. Die Änderung des Wetters von einem auf den anderen Tag lässt sich in Form einer Matrix darstellen, welche die möglichen Zustandsänderungen und deren Wahrscheinlichkeiten beschreibt. Die Wahrscheinlichkeiten von einem Zustand in einen anderen zu gelangen hängen bei einer Markov-Kette lediglich vom aktuellen Zustand ab und sind unabhängig von vergangenen Zuständen. Es muss jedoch der aktuelle Zustand exakt bekannt sein. Im oben beschriebenen Beispiel: Das aktuelle Wetter. Eine ausführliche Erläuterung von Markov-Ketten lässt sich [40] entnehmen. Im Dialog Management handelt es sich bei der In-

tention des Benutzers um den Zustand. Der Zustand ist jedoch nicht exakt bestimmbar und somit ist es unsicher, wodurch die Verwendung von Markov-Ketten ausgeschlossen werden kann.

POMDP lassen eine Unsicherheit über den Zustand der Markov-Kette zu und ermöglichen die Modellierung der Beschaffung von Informationen. Eine Markov-Kette im generellen ist so definiert, dass die Wahrscheinlichkeit vom Zustand s_1 in den Zustand s_2 zu wechseln lediglich von s_1 abhängt. Die Änderung des Zustands erfolgt über die Zustandsänderung a_1. Der Zustand s_0, der Vorgänger von s_1, ist für diese Wahrscheinlichkeit irrelevant. Im Umfeld von Gesprächsassistenten ist dies jedoch nur sehr selten der Fall. Bei Gesprächsassistenten spielt auch der bereits durchgeführte Dialog für den weiteren Verlauf eine Rolle. Dies kann anhand eines Beispiels erläutert werden. Der Benutzer wird vom System gefragt, für wie viele Personen ein Tisch im Restaurant zu reservieren ist. Daraufhin antwortet der Benutzer mit "Für 5 Personen". Die Antwort des Benutzers hat somit einen direkten Bezug auf die Frage des Assistenten und somit besteht ein Bezug zwischen diesen zwei Zuständen. Um nun eine korrekte Aktion des Gesprächsassistenten zu ermitteln, muss dieser Bezug möglich sein. Für sich alleine hat die Aussage des Nutzers keinen Informationsgehalt. Im Zusammenhang mit der zuvor gestellten Frage entsteht jedoch dieser Kontext. POMDP ermöglichen es, einen Bezug zu vorhergegangenen Zuständen zu bilden. Es wird jedoch dabei nicht der Zustand s selbst betrachtet, sondern die Beobachtung o, welche sich aus dem Zustand s ergibt. Durch diese Generalisierung ist ein Zustand s_2 nicht mehr nur von s_1 abhängig, sondern auch von den Beobachtungen o_1 und o_0, welche zuvor gemacht wurden. Dieser Zusammenhang ist graphisch in der Abbildung 3.2 aufbereitet. Bei dieser Darstellung wird bereits deutlich, dass bei zunehmendem historischen Bezug, die Komplexität des Graphen exponentiell steigt. Auch bei einer steigenden Anzahl möglicher Zustände erhöht sich die Komplexität so enorm, dass dies ein Problem darstellt [77]. Letztlich ist der zeitaufwendige Lernprozess bei POMDP noch zu erwähnen.

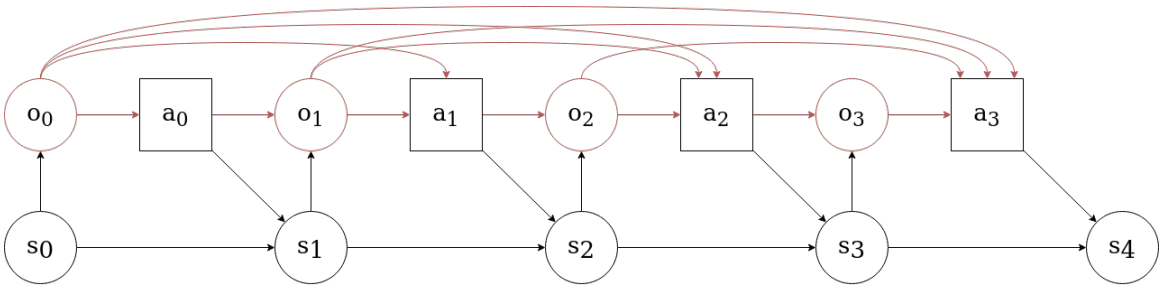


Abbildung 3.2: Darstellung einer Partially Observable Markov Decision Processes (POMDP) nach [47, 15]

Neuronale Netze

Neben dem Ansatz das Dialog Management mit POMDP zu realisieren etablierten sich auch mehrere Varianten von neuronalen Netzen, um diese Komponente der Dialogsysteme umzusetzen [28, 33]. Mit Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) und Transformer finden vor allem drei Typen von neuronalen Netzen eine weite Verbreitung.

Recurrent Neural Network (RNN)
RNNs bieten eine gute Möglichkeit, um mit sequenziellen Daten umzugehen [58]. Diese Eigenschaft ermöglicht die Nutzung in diesem Bereich [73], da vor allem der zeitliche Bezug einen großen Aspekt für die korrekte Vorhersage im Dialog Management darstellt. Jedoch stellt auch diese Eigenschaft eine Limitierung in der Nutzbarkeit da. Die standardmäßige Umsetzung von RNN hat ein Problem damit mehrere parallele Gesprächsfäden zu realisieren [72]. Ein weiteres

Problem eines RNN besteht darin, dass es nicht vorgesehen ist den zeitlichen Bezug zu vergessen. Für dieses Problem bietet Long Short-Term Memory, eine spezielle Architektur der RNN, eine Lösung.

Long Short-Term Memory (LSTM)
Long Short-Term Memory [27] besteht aus drei Varianten von Gates. Bei einem Gate handelt es sich um eine Struktur, welche eine Sigmoidfunktion[1] und eine punktweise Multiplikationsoperation beinhaltet. Die Gates dienen dazu den Informationsfluss in und aus der Zelle zu steuern. Jede LSTM Zelle besteht aus einem *Input-Gate*, *Output-Gate* und *Forget-Gate*. Das Forget-Gate ermöglicht es der LSTM Zelle Zeitpunkte zu ermitteln, an denen der Bezug zu Vergangenem vergessen werden soll [19]. Durch diese Funktionalität bietet sich diese Art von Architektur für Dialog Management an [50]. Dies liegt unter anderem daran, dass wie bei RNN der zeitliche Bezug möglich ist und wenn es gewünscht ist, wie bei dem Wechsel des Kontextes im Gespräch, der vergangene Kontext vergessen werden kann. Dadurch können Kontexte im Gespräch beendet und neu hergestellt werden. Dies lässt sich mit einem RNN nicht realisieren. Jedoch ist es nicht möglich mehrere Kontexte gleichzeitig zu führen. So kann nicht zwischen mehreren Gesprächsfäden gewechselt werden.

Transformer
Mit Transformer [69] etabliert sich neben RNN eine weitere Architektur. Es handelt sich um eine Architektur basierend auf einem Decoder-Encoder. In der Vorstellung Vaswani u. a. [69] dieser Architektur wird das Übersetzen von Texten als Anwendungsbeispiel genannt. Für die Durchführung einer korrekten Übersetzung ist es wichtig den Kontext korrekt zu interpretieren. So kann das Wort Bank in verschiedenen Kontexten unterschiedliche Bedeutungen aufweisen. Es kann sich entweder um die Finanzinstitution oder um die Sitzgelegenheit handeln. Um festzustellen, ob es sich im jeweiligen Kontext um das eine oder andere handelt, wird ein Aufmerksamkeitsmechanismus genutzt. Dieser Mechanismus ermöglicht es beispielsweise die Relevanz von einzelnen Wörtern in einem Satz zueinander zu ermitteln. Die Bedeutung von Relevanz kann an folgendem Beispiel gezeigt werden. Es wird der Satz “Er saß auf der Bank im Park” betrachtet. Anhand der Wörter “saß” und “Park” kann der Leser feststellen, dass es sich bei dem Wort “Bank” um die Sitzgelegenheit handeln muss. Somit sind diese zwei Wörter von hoher Relevanz für das Wort “Bank”, wohingegen die restlichen Wörter eine geringere Relevanz aufweisen. In Verbindung mit diesen Wörtern ist Bank also als “bench” und nicht als “bank” in das Englische zu übersetzen. Dieser Mechanismus ist auch für ein korrektes Dialog Management essenziell. Es wird ermöglicht, einen korrekten Bezug innerhalb des Dialogs herzustellen. Einem Transformer ist es möglich irrelevante Eingaben, beispielsweise belanglose Zwischenfragen des Benutzers, für die Ermittlung weiterer Vorhersagen zu ignorieren. Eine ausführliche Erläuterung der Funktionsweise dieser Variante der Neuronalen Netze wir in dieser Arbeit nicht durchgeführt. Es wird an dieser Stelle auf weiterführende Literatur verwiesen [29, 69].

3.3 Varianten des maschinellen Lernens

Nachdem im vorherigen Abschnitt auf die verschiedenen Algorithmen und Architekturen eingegangen wurde, beschäftigt sich dieser mit den verwendeten Varianten des maschinellen Lernens. Es werden lediglich Varianten behandelt, welche im Bereich der Gesprächsassistenten Verwendung finden. So werden Supervised Learning, Transfer Learning und Reinforcement Learning erläutert.

[1] $\mathrm{sig}(t) = \frac{1}{1+e^{-t}}$

3.3.1 Supervised Learning

Bei Supervised Learning wird dem Lernalgorithmus ein Set an gelabelten Beispielen als Trainingsdaten übergeben. Anhand dieser Daten werden Hypothesen über noch ungesehene Daten aufgestellt. So ist es möglich eine Vorhersage über noch nicht bekannte Daten zu treffen [43]. Es handelt sich bei Supervised Learning um das häufigste Szenario im Zusammenhang mit Klassifizierungs-, Regessions- und Rankingproblemen. Supervised Learning als Variante des Lernens findet in Dialogsystemen ein hohes Maß an Verwendung [59, 38, 72, 71]. Diese Variante des Lernens wird vor allem dann verwendet, wenn eine große Anzahl an Daten für das Training vorhanden ist. Supervised Learning findet neben der Wissenschaft auch in Frameworks wie beispielsweise Rasa Verwendung (siehe Kapitel 3.6.3 auf Seite 15).

Supervised Learning kann auch dazu verwendet werden, um einen Algorithmus vorab zu trainieren und mit Reinforcement Learning zu verbessern [36, 35]. Auf diese Variante des Lernens wird im Weiteren eingegangen.

3.3.2 Reinforcement Learning

Seit das Unternehmen Google DeepMind mit [41, 42] gezeigt hat, wie es möglich ist, mittels Reinforcement Learning (deutsch: bestärkendes Lernen) ein neuronales Netz so zu modellieren und trainieren, dass es im Stande ist Atari Spiele selbstständig zu lernen und auf überdurchschnittlichem Niveau zu spielen, findet dieser Ansatz eine immer größere Beachtung. Spätestens mit AlphaGo [62] und AlphaGo Zero [63] stellte DeepMind die Möglichkeiten dieser Lernvariante unter Beweis. Hierbei handelte es sich um neuronale Netze, welche fähig waren das Brettspiel “Go” so gut zu erlernen, dass selbst Profispieler besiegt wurden. So ist es nicht unerwartet, dass auch im Umfeld der Dialogsysteme die Anwendbarkeit dieser Art des Lernens diskutiert wird [78, 74, 24, 33, 35, 66]. Folgend wird auf Reinforcement Learning und Ausprägungen davon eingegangen.

Um Informationen zu sammeln, interagiert der Lernende, auch als *Agent* bezeichnet, aktiv mit der Umgebung, auch *Environment* genannt, und beeinflusst dadurch die Umgebung und erhält für Aktionen eine direkte Belohnung [43, S. 7], einen *Reward*. Diese Belohnung wird üblicherweise durch einen *Interpreter* vergeben. Der Zusammenhang zwischen Agent, Environment und Interpreter ist in der Abbildung 3.3 dargestellt. Das Ziel des Lernenden ist es, den Reward über eine Reihe von Aktionen und Wiederholungen mit der betreffenden Umgebung zu maximieren. Die Umgebung liefert jedoch kein langfristiges Belohnungsfeedback. Gleichzeitig steht der Lernende vor dem Problem der Exploration gegenüber der Ausbeutung, da er sich zwischen der Erforschung unbekannter Aktionen zur Gewinnung weiterer Informationen und der Ausnutzung der bereits gesammelten Informationen entscheiden muss.

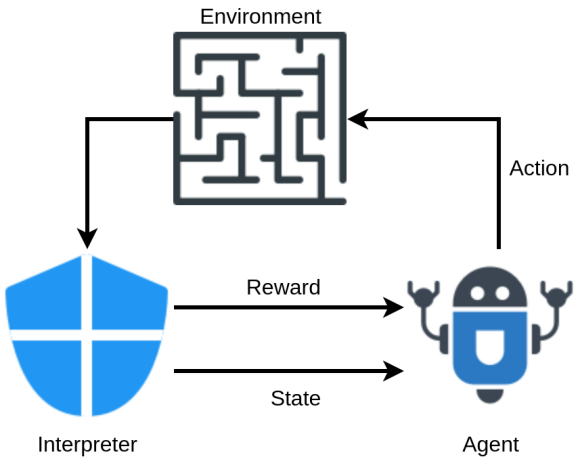


Abbildung 3.3: Zusammenhang zwischen Agent, Environment und Interpreter

Ein Beispiel, welches dieses Verfahren anschaulich beschreibt ist ein Kind, das nach dem Essen seinen Eltern hilft den Tisch abzuräumen. Wenn das Kind sein Geschirr in die Spülmaschine räumt, darf es sich ein Stück Schokolade nehmen. Das Kind erhält einen Reward. Auch das nächste mal

erhält das Kind noch diesen Reward, doch irgendwann wird es nur noch dankende Worte durch die Eltern hören. Das Kind muss neue Wege finden um eine Schokolade als Reward zu erreichen. Das Kind wählt den Weg der Exploration und hilft die Spülmaschine auszuräumen. Es erhält wieder als Belohnung ein Stück Schokolade. So hat das Kind gelernt, es erhält eine Belohnung, einen Reward, wenn es dabei Hilft die Spülmaschine ein- und auszuräumen.

Es gibt verschiedene Ansätze, die verfolgt werden, um Reinforcement Learning für das Training von Chatbots umzusetzen. So ist ein simulierter User als Interpreter nutzbar [33, 66, 74]. Das Problem, auf welches sie dabei stoßen, ist die hohe Komplexität, die bei der Implementierung dieses simulierten Users auftritt. Es ist bereits notwendig, in der Simulation ein Dialog Management umzusetzen. Es stellen sich die Frage, welche Reaktion durch die Simulation erfolgt, wenn der Bot eine unerwartete Antwort gibt, welche aber grundsätzlich als korrekt zu deuten ist. Somit verschiebt sich ein hohes Maß der Implementierungsaufwände auf diese Simulation. [33, 66] greifen für die Simulation auf die im Kapitel 3.4 auf der nächsten Seite beschriebene Technik der Datengenerierung zurück, die von Schatzmann und Young in [56, 54] beschrieben wurde.

Eine zweite Variante besteht darin, das Feedback von Benutzern als Reward zu nutzen [78, 36, 35, 24]. Hier wird auf die Implementierung eines simulierten Users verzichtet und der Bot interagiert direkt mit Menschen, welche ein Feedback an den Bot übermitteln. Es ist somit nicht nötig, die Komplexität in einer Simulation zu implementieren. Eine breite Anzahl an Beta-Testern ist jedoch notwendig, sollte auf ein anfängliches Supervised Learning verzichtet werden. Der Verzicht kann beispielsweise aufgrund fehlender Trainingsdaten notwendig sein. Die praktikablere Variante besteht darin den Bot mit Trainingsdaten mittels Supervised Learning zu trainieren und das Feedback für ein späteres Training zu nutzen [24].

3.3.3 Transfer Learning

Ein Mensch wendet meist bereits bekanntes Wissen in einem neuen Umfeld an. Er transferiert also sein bekanntes Wissen in eine neue Domäne. Dadurch ist es ihm möglich schneller in neuen Umgebungen korrekt zu interagieren. Das Ziel bereits bekanntes Wissen aus einer anderen Domäne als Grundlagenwissen für eine neue, aber ähnliche Domäne anzuwenden, verfolgt das Transfer Learning.

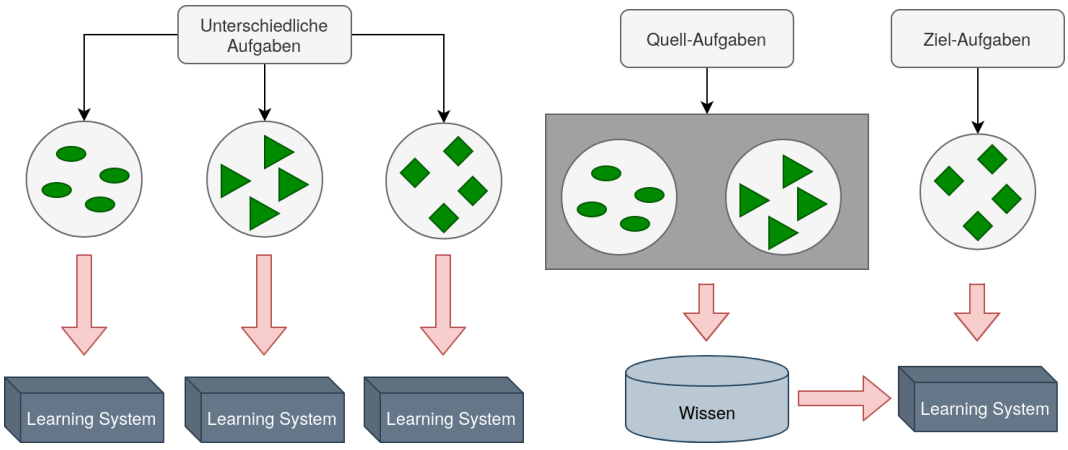


Abbildung 3.4: Vergleich zwischen maschinellem Lernen ohne Transfer Learning (links) und Transfer Learning (rechts) nach [46]

Im Transfer Learning ist eine Quelldomäne und eine Quellaufgabe vorhanden [46]. Diese dienen

als Trainingsdaten für Supervised Learning, womit eine Approximation der Funktion $f_S(\cdot)$ erfolgt. Transfer Learning soll dazu beitragen, das Lernen der Zielvorhersagefunktion $f_T(\cdot)$ zu verbessern. Es handelt sich hierbei um die approximierte Funktion der Zieldomäne und der Zielaufgabe. Um dies zu ermöglichen ermöglichen dient $f_S(\cdot)$ als Grundlage für das Lernen von $f_T(\cdot)$.
Zusammengefasst kann gesagt werden, dass Domäne und Aufgabe einer Quelle dazu verwendet werden, die Ergebnisse in einem Ziel zu verbessern. Diese Methode kommt zum Einsatz, wenn im Ziel nicht ausreichend Daten vorhanden sind, aber eine Quelle als Basis verwendet werden kann. Die Abbildung 3.4 auf der vorherigen Seite vergleicht das traditionelle maschinelle Lernen mit Transfer Learning.

Die Autoren von [28] nutzt eine Kombination aus Transfer Learning und Reinforcement Learning. Sie sprechen direkt das Problem der fehlenden oder nicht ausreichend vorhandenen Daten an, welches sie mit diesem Ansatz zu lösen versuchen. Dies kann gelingen, wenn es möglich ist von der Quelldomäne zu profitieren. Bereits beim Training der Quelle müssen die Ein- und Ausgabe-Variablen des Ziels bekannt sein, da sonst ein Training nicht zum gewünschten Ziel führt. Es wird ein neuronales Netz in zwei Durchläufen mit zwei Datenbeständen trainiert. Im in [28] beschriebenen Beispiel wird der erste Durchlauf, das Training der Quelldomäne, mit Daten zu Film-Reservierung durchgeführt. Der zweite Durchlauf erfolgt dann in der Zieldomäne, welche hier Reservierungen von Restaurants beinhaltet. Die Überschneidungen in beiden Domänen sind dann z.B. die Stadt, die Anzahl der Personen und das Datum. Unterschiedlich sind Namen des Films, Startzeitpunkt, Kino, Essen, Preislage und Adresse des Restaurants.
Es war ihnen mit der beschriebenen Methode möglich, mit einer geringen Anzahl an Trainingsdaten zu guten Ergebnissen zu gelangen. Der Chatbot wies bei dieser Art des Lernens auch dann eine bessere Performance auf, wenn für die Zieldomäne ausreichend Daten verfügbar waren und trotzdem auch die Quelldomäne für ein Training genutzt wurde.

3.4 Datenbeschaffung

Als Ersteller eines Gesprächsassistenten kann man sich folgende Fragen stellen: “Kann ich auf bereits vorhandene Daten zurückgreifen? Nuss ich Daten selbst erzeugen? Ist es mir möglich Dialoge zu generieren?” In diesem Kapitel werden Erfahrungen zu dieser Fragestellung dargestellt, um einen Rahmen für eine zukünftige Beantwortung dieser Fragen zu bieten.

3.4.1 Vorhandene Datensätze

Bei der Menge der Daten sind zwei Aspekte zu beachten [60]. Eine größere Menge bietet eine erhöhte Variabilität in der Struktur der einzelnen Dialoge. Andererseits erhöht sich mit mehr Daten die Komplexität der Sprache und Dialoge. Die erhöhte Komplexität kann dazu führen, dass Menschen besser verstanden werden. Es ist aber sicherzustellen, dass die verwendeten Algorithmen die Komplexität abbilden können. Es ist nicht auszuschließen, dass die verwendeten Algorithmen und Modelle die Komplexität der Sprache nicht abbilden können und dadurch die Qualität der verwendeten Modelle und Algorithmen leidet. Da es sich bei der Erzeugung von qualitativen Trainingsdaten um einen aufwendigen Prozess handelt, ist eine Möglichkeit bereits vorhandene Daten zu nutzen. Einerseits ist zwischen den fachlichen Themenbereichen zu unterscheiden. So gibt es beispielsweise Datensätze aus den Bereichen Twitter [52], technischen Foren [68], Film [12, 2], Lifestyle [13] und Standortbestimmung [9]. Andererseits ist zwischen den Arten der Kommunikation zu unterscheiden. So kann zwischen Mensch-Maschine und Mensch-Mensch Kommunikation unterschieden werden. Diese Unterscheidung ist aufgrund der höheren Komplexität der Unterhaltungen zwischen Menschen dringend notwendig [75]. Es hat sich gezeigt, dass Menschen anders interagieren, wenn sie davon ausgehen mit einem Programm zu kommunizieren [31]. Diesen Ef-

fekt nutzt die Wizard-of-Oz Methode, welche im Rahmen der Mensch-Maschine Kommunikation erläutert wird.

Nicht beachtet werden im Rahmen dieser Arbeit Datensätze, welche auf gesprochener Kommunikation beruhen. Es wird lediglich auf Dialoge aus geschriebener Kommunikation zurückgegriffen.

Mensch-Mensch Kommunikation

Ein Beispiel für einen Korpus mit Mensch-Mensch Kommunikation ist der Ubuntu Korpus [68]. Dieser beinhaltet knapp eine Millionen Dialoge zwischen jeweils zwei Personen. Als Basis dienten die Ubuntu Chat Logs von 2004 bis 2015[2]. Die Dialoge beinhalten mindestens drei und durchschnittlich acht Wechsel zwischen den Nutzern.

Dieser Art Korpora steht Williams und Young [75] kritisch gegenüber. Mensch-Mensch Kommunikation beinhaltet eine größere Anzahl an Wendungen innerhalb des Dialogs und auch das Verständnis von Fehlern stellt Dialogsysteme vor Probleme. Die Kommunikation ist sehr viel komplexer und kann mit aktuellen Systemen nicht korrekt abgebildet werden. Serban u. a. [60] ist dieser Aussage gegenüber kritisch und schließt es nicht aus, diese Art Korpora als Trainingsdaten zu verwenden. Er schränkt die Aussage von [75] auf die Verwendung von gesprochenen Dialogen und für die Verwendung in der Lernmethode Reinforcement Learning ein. Gegen die Aussage von [75] spricht die Etablierung des Twitter Korpus [52] als Standard für die Evaluation von Ergebnissen solcher Systeme [37].

Mensch-Maschine Kommunikation

Da der Mensch sich laut [31] automatisch anders verhält, wenn er davon ausgeht mit einem Programm zu kommunizieren, kann zwischen zwei Arten dieser Kommunikation unterschieden werden. Bei der einen Art handelt es sich um eine echte Mensch-Maschine Kommunikation. So kommuniziert der Mensch mit einem Gesprächsassistenten, Telefonansagesystemen oder anderen denkbaren Systemen. Bei der anderen Art handelt es sich um eine Mensch-Mensch Kommunikation, wobei jedoch einer der Menschen davon ausgeht mit einer Maschine zu kommunizieren. Da der Mensch davon ausgeht mit einer Maschine zu kommunizieren, passt er seine Art zu kommunizieren an [31]. Es handelt sich bei diesem Verfahren um die sogenannte Wizard-of-Oz Methode. Diese wird unten näher beschrieben. Es gibt Korpora wie den »Carnegie Mellon Communicator Corpus«. Dieser basiert auf insgesamt 180.605 Anrufen bei einem Reiseplanungssystem [3]. Basiert ein Korpus lediglich auf Daten aus einem Dialog Systems besteht eine Beschränkung auf die dort implementierten Funktionalitäten.
Eine weite Verbreitung finden die Korpora der Dialog State Tracking Challenge (DSTC) 1 - 3. Im Speziellen sind hierbei die Daten der zweiten Challenge »The Second Dialog State Tracking Challenge« zu nennen, welche ungefähr 500 Dialoge mit durchschnittlich 7.88 Frage-Antwort-Interaktionen bei 184 einzigartigen Anrufern beinhalten. Die Daten wurden mit drei unterschiedlichen Dialog Managern erstellt. Hierdurch ist der Korpus nicht auf die Funktionalität eines einzelnen Dialog Managers beschränkt.

Den von [31] beschriebenen Effekt der Andersartigkeit in der Kommunikation nutzt die bereits erwähnte Wizard-of-Oz Methode. Bei dieser Methode kommunizieren zwei Menschen miteinander. Einer der beiden geht jedoch davon aus, mit einer Maschine zu kommunizieren. Die Person, welche die Maschine inszeniert, wird als Wizard bezeichnet. Ursprünglich wurde diese Methode auch als PNAMBIC (Pay No Attention to the Man Behind the Curtain) bezeichnet und erst später als Wizard of Oz. Diese Bezeichnung wurde durch J. Bernstein eingeführt [17]. Diese Methode

[2] `http://irclogs.ubuntu.com/`

wurde bei der Erstellung mehrerer Korpora genutzt [57, 10]. In der Literatur finden sich auch Aussagen, es handle sich bei der Wizard-of-Oz Methode um Mensch-Mensch Kommunikation [10]. Diese Unstimmigkeit zeigt die Besonderheit der so erzeugten Daten. Eine eindeutige Zuordnung zu einer Kategorie ist nur schwer möglich und diskutierbar.

3.4.2 Daten selbst erstellen

Eine weitere Möglichkeit ist die Erstellung von eigenen Daten. Dies kann dann notwendig werden, wenn es zu dem Themenbereich keine nutzbaren Daten gibt. Der fachliche Kontext oder die verwendete Sprache sind mögliche Gründe hierfür. Eine geringe Komplexität der vorgesehenen Dialoge kann dazu führen, dass keine große Menge an Daten nötig ist. Ein Vorteil dieses Vorgehens besteht darin, dass die Datengrundlage vollständig bekannt ist und eine Analyse bezüglich der Qualität nicht benötigt wird. Die Komplexität der Dialoge und der Sprache hat somit der Ersteller selbst in der Hand. Bei einem Use-Case der von Standards, wie Hotelbuchung oder Film und Fernsehen abweicht, kann der Aufwand der selbstständigen Erstellung der Daten durchaus geringer sein als die Anpassung bereits vorhandener. Jedoch ist der hierfür benötigte Zeitaufwand nicht zu unterschätzen. Auch besteht die Gefahr einen unterbewussten Bias in die Daten mit einzubauen.

In [28] wurde gezeigt, dass bereits eine geringe Anzahl an Trainingsdaten bei einem neuronalen Netz, welches mittels Supervised Learning trainiert wird, ausreichend gute Ergebnisse liefern kann. Dort dienten 120 Dialoge als Grundlage für das Training und 32 für die Evaluation. Eine vergleichbare Anzahl wurde in [71] verwendet. Hier haben die Entwickler auf einen selbst erstellten Datensatz von 108 Dialogen zurückgegriffen. Diese beiden Beispiele zeigen, dass bereits mit einer geringen Anzahl von ungefähr 100 Dialogen mit guten Ergebnissen gerechnet werden kann.

3.4.3 Automatisierte Dialog Generierung

Ein weiterer Ansatz besteht darin, ohne Zugriff auf Daten oder mit einer geringen Anzahl, neue Dialoge zu generieren. Liu und Lane verwendeten einen zweistufigen Aufbau für das Training ihres vorgestellten Systems [35]. Als Basis wurden die Daten der zweiten Dialog State Tracking Challenge (DSTC) verwendet. Mit diesen Daten trainierten sie einen User Agent, welcher im Weiteren den Dialog Agent trainierte. Es handelt sich in diesem Beispiel zwar um eines, welches Daten generierte, jedoch verwendeten die Forscher mit dem Datensatz der DSTC 2 eine sehr umfangreiche Datengrundlage [26] und die daraus resultierenden Ergebnisse sind auf ihre Anwendbarkeit auf kleinere Datensätze zu hinterfragen. Ein Modell, welches mit geringerer Anzahl an Daten auskommt, wurde durch Schatzmann u. a. beschrieben [55, 56, 54]. Es handelt sich um wahrscheinlichkeitstheoretische Methoden zur Simulation des Benutzerverhaltens, das auf kompakten Darstellung des Benutzerziels und einer stack-ähnlichen Benutzeragenda basiert. Das Modell bietet eine Möglichkeit, die relevanten Dialoggeschichten aus Sicht des Benutzers zu koordinieren. Die Anpassung erfolgt anhand einer sehr kleinen Anzahl an Parametern. Dieses Verfahren kann verwendet werden, wenn keine Trainingsdaten vorhanden sind. Die Nutzbarkeit dieses Verfahrens wurde unter anderem in [66, 39, 28] bewiesen.

3.5 Augmentierung der Daten

Das Ziel der Augmentierung ist es die gezielte Manipulation der vorhandenen Daten für das Training durchzuführen [14]. Ziel ist es so die Ergebnisse in Bezug auf die korrekte Interpretation der gewünschten Aktionen anhand der Interaktionen des Benutzers zu verbessern. In diesem Abschnitt wird auf verschiedene Verfahren der Augmentierung für ein verbessertes Dialog Management eingegangen.

Eine sehr einfache Variante der Augmentierung wurde im Framework Rasa umgesetzt. Auf Rasa wird im Abschnitt 3.6.3 auf Seite 15 ausführlicher eingegangen. Das Ziel der umgesetzten Augmentierung besteht darin, dass das Dialog Management den vergangenen Dialog ignoriert, wenn keine Abhängigkeit zu diesem besteht [50]. Es soll lediglich auf zuvor getätigte Interaktionen eingegangen werden, wenn diese von Belang sind. Die Augmentierung erfolgt indem zufällig gewählte Dialoge aneinandergereiht werden. Ein Vergleich zwischen Ergebnissen unter Verwendung dieses Verfahrens und ohne konnte nicht gefunden werden. Lediglich ein Kommentar auf Github, dass eine Erhöhung der Anzahl der augmentierten Dialoge zu besseren Ergebnissen führt [70], weist auf die positiven Effekte hin. Genaue Messungen liegen hierfür nicht vor.

Eine weitere Methode der Anreicherung ist das Hinzufügen von domänenfremden Anfragen. Hierzu können beispielsweise vorhandene Dialoge mit Chat-Daten aus einem Open-Domain Korpus angereichert werden [79]. Die Erweiterung erfolgt, indem an zufällig gewählten Positionen im Dialog zufällige Dialoge aus den Chat-Daten eingefügt werden und anschließend die letzte durch den Chatbot getätigte Interaktion wiederholt wird. Als Gründe für die Nutzung eines solchen Verfahrens spricht laut [79] das größere Vokabular des Modells und das Verständnis über das Rückführen in den Gesprächsfluss. Als Ergebnis konnte festgestellt werden, dass die Resultate bei automatisierten Tests und bei Tests mit Menschen jeweils besser waren im Vergleich zu dem identischen Dialog Manager ohne der Augmentierung innerhalb der Trainingsdaten.

Du und Black konnten nach der Analyse von Dialogen mehrere Eigenschaften feststellen, bei denen sich für Dialogsysteme Probleme ergeben [14]. So werden lange Äußerungen, uneinheitliches Verhalten beim Wechsel des Kontextes und die Unterbrechung der Kontinuität als solche Phänomene genannt. Um diese Probleme anzugehen, erfolgte die Manipulation der Daten mit zwei Methoden. Diese Methoden bezeichnen die Autoren als »Permutation« und »Flipping«. Die Permutation tauscht die Reihenfolge zweier Aussagen aus. Die Permutation zielt auf das Dialog Management. Ein Problem bei dieser Methode könnte darin bestehen, dass Dialoge permutiert werden, welche es korrekter Weise aufgrund ihres Kontextes nicht erlauben und somit die Qualität der Trainingsdaten verringert wird. Dieses Problem wird jedoch nicht diskutiert.
Flipping als zweite Methode kehrt die Reihenfolge eines Satzes um. Es wird der Satz in der Mitte geteilt und die beiden Teile werden vertauscht oder es wird die Reihenfolge von Haupt- und Nebensatz getauscht. Das Flipping erhöht die Variabilität bei den Trainingsdaten für das Natural Language Processing (NLP). Das Ziel ist es die Genauigkeit bei der Erkennung der Intentionen zu erhöhen. Durch die beschriebenen Methoden konnten unter Verwendung verschiedener Netze und vier verschiedener Korpora in zwei Sprachen (Englisch und Chinesisch) 1 bis 3 Prozentpunkte bessere Ergebnisse bei Tests erzielt werden [14].

Ein Verfahren, welches der Datengenerierung nahe kommt, wird von [32] beschrieben. Es ist aufgrund der genutzten Korpora, Weibo und Twitter, davon auszugehen, dass das beschriebene Verfahren nicht in aufgaben-orientierten, sondern offenen Dialogsystemen zum Einsatz kommt. Dieses Modell besteht aus zwei Komponenten. Der »conditional variational autoencoder (CVAE)« erzeugt Frage-Antwort Paare, welche durch den »discriminator« auf die Durchführbarkeit geprüft werden. Hierfür wird auf Methoden des maschinellen Lernens zurückgegriffen. Die Komponenten werden zunächst mittels eines Korpus trainiert und anschließend werden neue Dialoge anhand der Ergebnisse des Trainings erzeugt. Eine nähere Erläuterung der angewendeten Verfahren ist [32] zu entnehmen.

Diese beschriebenen Methoden zeigen die unterschiedlichsten Herangehensweisen an die Augmentierung und die verschiedenen Ziele, die mit der jeweiligen Methode angestrebt werden. So kann eine Verbesserung des Dialog Management durch das Hinzufügen von Geplauder [79] oder

das Ändern der Reihenfolge innerhalb eines Dialogs [14] erfolgen. Auch die Erzeugung neuer Frage-Antwort Paare [32] ist ein Verfahren für die Verbesserung des Dialog Management. Es wurde mit dem Flipping [14] auch eine Methode für die Verbesserung der Ergebnisse des Natural Language Processing dargelegt.

3.6 Frameworks

Im Laufe dieses Abschnitts wird auf Frameworks eingegangen, welche Dialogsysteme umsetzen. Es wird eine Closed-Source Variante, eine Implementierung aus dem wissenschaftlichen Umfeld und auf eine Open-Source Lösung behandelt. DialogFlow repräsentiert die Closed-Source Variante. Es handelt sich um die Cloud Lösung von Google. Mit RavenClaw wird ein hierarchisch strukturiertes, regelbasiertes System der Carnegie Mellon University erläutert. RavenClaw findet neben der Verwendung in einzelnen Systemen [65] ein breites Interesse im Umfeld der Forschung. Bei Rasa handelt es sich um ein »Open source machine learning framework to automate text- and voice-based conversations« [51].

3.6.1 DialogFlow

Bei DialogFlow handelt es sich um das Dialog Management Tool, welches von Google bereitgestellt wird. Weitere konventionelle, Cloud-basierte Tools sind unter anderem IBM Watson Assistant oder Azure Bot Service von Microsoft. All diese Lösungen vereint, dass sie den Entwicklern von Dialogsystemen einen vergleichbaren Funktionsumfang bieten und es nicht ermöglichen Algorithmen und verwendete Modelle einzusehen oder an die jeweiligen Bedürfnisse anzupassen.

DialogFlow ist den regelbasierten Dialog Management Tools zuzuordnen [25]. Die Dokumentation des Tools gibt selbst keine Auskunft darüber wie das Dialog Management umgesetzt wird. Es wird lediglich darauf verwiesen, dass für das Natural Language Understanding auf maschinelles Lernen zurückgegriffen wird [20]. Da DialogFlow vollständig in der Google Cloud Plattform integriert ist, muss eine Anmeldung auf dieser Plattform für die Erstellung eines Assistenten vorgenommen werden.

Der typische Ablauf einer Konversation ist der Abbildung 3.5 zu entnehmen. Die Interaktion mit DialogFlow ist nicht, wie die Abbildung es vermuten lässt, ausschließlich mit Google Assistant möglich. Es sind unterschiedliche Optionen wie beispielsweise Slack, Facebook, Twitter, Amazon Alexa oder verschiedene Telefonieintegrationen möglich [21].

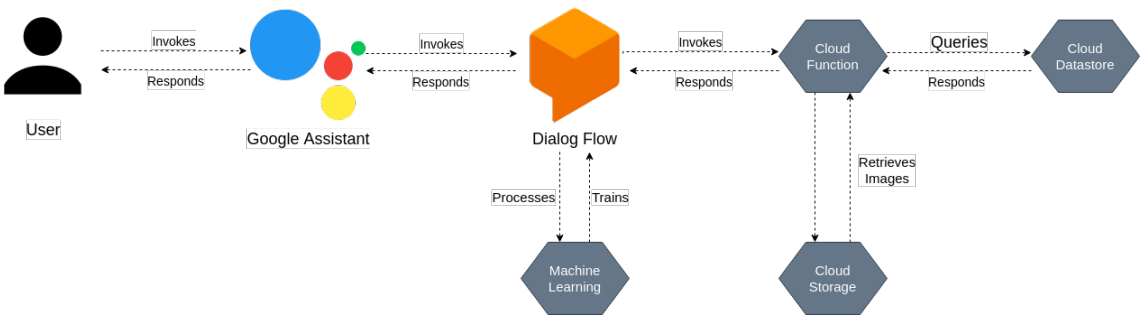


Abbildung 3.5: DialogFlow-basierender Chatbots nach [64]

Es ist möglich, Intents und Entities zu erstellen. Bei einem Intent handelt es sich um die Absicht des Benutzers. Eine Entity stellt einen Parameter dar, welcher vom Benutzer innerhalb des In-

tents übergeben wird. Übergibt der Benutzer dem Assistenten beispielsweise “Ich möchte für Freitag einen Tisch reservieren” so ist die Tischreservierung der Intent und Freitag die Entity dieses Intents. Mithilfe von Eingabe- und Ausgabekontext kann der Ablauf der Unterhaltung gesteuert werden [64]. Um Interaktionen mit dem Agenten zu ermöglichen, welche nicht über statische Antworten umzusetzen sind, bietet DialogFlow die Möglichkeit, dies mit Fulfillments umzusetzen. Bei Fulfillments handelt es sich um Services die durch den Entwickler erstellt werden können und als Cloud Functions in der Plattform entwickelt und ausgeführt werden.

DialogFlow bietet vordefinierte Agenten, welche den Einstieg vereinfachen sollen [23]. Es werden Anwendungsfelder wie Hotelbuchung, Navigation oder Online-Shopping abgebildet. Dies vereinfacht die Erstellung von Entities und Intents. Auch wird die Möglichkeit geboten, das optionale Feature Small Talk einzubinden. Bei Aktivierung dieser Erweiterung »reagiert der Agent auf allgemeine Themen, emotionale Antworten und Fragen zum Agent« selbst [22].

Ein explizites Dialog Management ist nicht möglich. DialogFlow ermöglicht es nicht das Dialog Management selbst zu implementieren oder zu optimieren. Es handelt sich um ein geschlossenes System, welches auch das Betreiben auf einer eigenen Infrastruktur nicht vorsieht.

3.6.2 RavenClaw

Bei RavenClaw handelt es sich um ein Dialog Management Framework für aufgaben-orientierte Systeme von Gesprächsassistenten [8]. Es kommen keine Methoden aus dem maschinellen Lernen zum Einsatz, um das Dialog Management durchzuführen. Es handelt sich um ein zweistufiges System. Des Weiteren handelt es sich um ein hierarchisch strukturiertes Dialog Management System, welches lediglich regelbasiert arbeitet. RavenClaw wurde ursprünglich in der Carnegie Mellon University entwickelt und als Open Source Projekt für weitere Forschungen und Entwicklungen veröffentlicht.
Die domänenspezifischen Aufgaben werden in einer hierarchischen Baumstruktur von Dialog-Agenten umgesetzt. Jeder Agent ist für die Durchführung eines Teils des Dialogs verantwortlich. Ein solcher beispielhafter Baum ist in der Abbildung 3.6 auf der nächsten Seite dargestellt. Die Wurzel verfügt über mehrere Unterknoten, wie im abgebildeten Beispiel der Login. Dieser Knoten teilt sich in weitere Agenten, welche in diesem Beispiel die Blätter des Baums ergeben. Es wird unterschieden zwischen zwei Typen von Agenten. Diese werden bezeichnet als *Fundamental Dialog Agents* und *Dialog Agencies*. Die Ersteren werden in der Abbildung dargestellt durch die grauen Knoten, welche die Blätter des Baums symbolisieren. Diese implementieren die atomaren Aktionen im Dialog. Auch hier gibt es Unterkategorien, jedoch wird auf diese nicht eingegangen. Der Zweck der Dialog Agencies, dargestellt durch die weißen Knoten, ist die Steuerung der untergeordneten Strukturen und die Kapselung dieser von der jeweils höheren Ebene.
Die zweite Stufe des Systems ist die Dialog Engine. Diese führt das Dialog Management aus, indem sie anhand der Spezifikation der domänenspezifischen Aufgaben in der hierarchischen Struktur der Agenten, die Konversation ausführt. Die Dialog Engine koordiniert des Weiteren die Fehlerbehandlung [7, 6], Zeitsteuerung oder Koordinierung von Wendungen im Dialog.

Ein Problem bei dieser Architektur stellt die Tatsache dar, dass ein hohes Maß an Implementierungsaufwand besteht. Das Dialog Management muss in Form der Agenten implementiert werden. Dieser hohe Aufwand stellt bei der möglichen Einführung des Systems ein Hindernis dar. Es ist nicht möglich, auf statistische Methoden zurückzugreifen. Im Gegensatz dazu lässt sich mit Rasa ein Dialog System anhand von Beispieldialogen erstellen, welche als Basis für statistische Modelle zur Durchführung des Dialog Management dienen können.

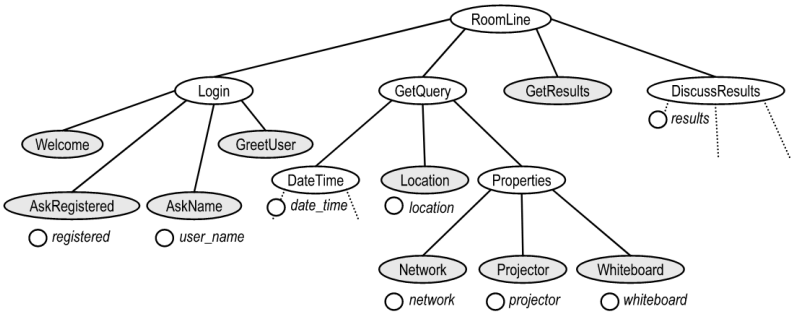


Abbildung 3.6: Beispiel eines Dialog Baums [8]

3.6.3 Rasa

Mit mehr als 2 Millionen Downloads [1] handelt es sich bei Rasa um ein Open Source Python Framework [5] mit zunehmender Relevanz. Rasa ist einerseits ein Open Source Projekt, welches über mehr als 300 Mitwirkende verfügt [1], und andererseits erscheinen auch dazu Publikationen zu Neuerungen innerhalb des Frameworks [5, 71, 72]. Durch diesen Umstand ist der Zugriff auf den Code des Frameworks uneingeschränkt möglich ist , eine ausführliche Dokumentation vorhanden und es existieren wissenschaftliche Betrachtungen. Im Rahmen dieser Arbeit wird nicht auf Rasa X eingegangen. Es handelt sich um ein closed-source Toolset für open-source Rasa[3].

Rasa ist in die zwei Hauptkomponenten Natural Language Understanding (Rasa NLU) und Dialog Management (Rasa Core) aufgeteilt.

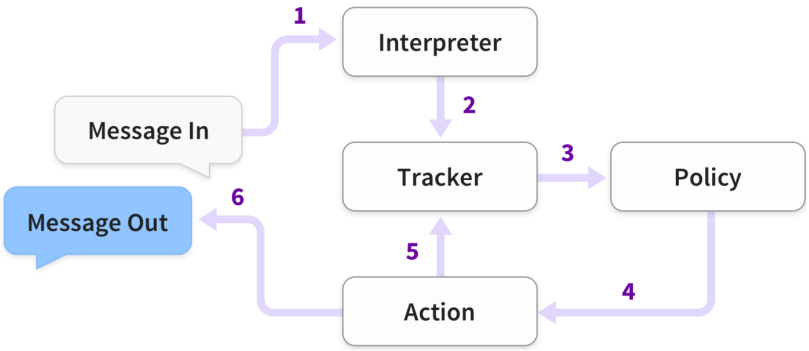


Abbildung 3.7: Architektur von Rasa [5]

Wird durch den Benutzer eine Nachricht eingegeben, werden die in der Abbildung 3.7 dargestellten Schritte durchlaufen [5]. Der Interpreter, der erste Schritt, wird durch Rasa NLU ausgeführt. Die Schritte Tracker, Policy und Action werden in Rasa Core umgesetzt. Die hier beschriebenen Schritte bilden die Komponenten der in der Abbildung 3.1b auf Seite 3 dargestellten Komponenten ab. Rasa hat das Ziel mit einem geringen initialen Aufwand bei Implementierung und einer geringen Anzahl an Trainingsdialogen einen Gesprächsassistenten umzusetzen. So stehen beispielsweise neun verschiedene Policies [50], Komponenten zur Realisierung des Dialog Management, zur Auswahl um verschiedene Anwendungsfälle zu verwirklichen. So ist es mit der *FormPolicy* möglich definierte Formulare auszufüllen. Die Policy fragt so lange, bis alle Felder des Formulars gefüllt wurden. Neben Policies wie dieser, welche algorithmisch interagieren, kann auch mit der *KerasPolicy* und *TEDPolicy* auf Policies auf Basis von maschinellem Lernen zurückgegriffen wer-

[3]Dokumentationen zu Rasa X finden sich unter `https://rasa.com/docs/rasa-x`

den.

Im Weiteren wird auf die Dateien und deren Eigenschaften eingegangen, welche bei Initialisierung eines Rasa Projekts vorhanden sind. Die Initialisierung eines Projekts erfolgt mit dem Rasa Command Line Interface (CLI). Ein initiales Projekt ist der Abbildung 3.8 zu entnehmen.

domain.yml
In dieser Konfiguration wird das Umfeld definiert, in welchem der Chatbot operiert. Dort werden `intents`, `entities`, `slots` und `actions` definiert [49]. Bei Intents handelt es sich um Absichten des Benutzers. Entities definieren Informationen, die den Intents entnommen werden können. Bei Slots handelt es sich um einen Key-Value Store, um Informationen temporär zu speichern. Bei Actions handelt es sich letztlich um Reaktionen die für den Bot in Frage kommen, um auf den Benutzer zu reagieren. Eine etwaige Variante der Actions sind `responses`. Es handelt sich dabei um die Antwortmöglichkeiten des Bots. Optional können auch diese innerhalb der Konfiguration angelegt werden.

config.yml
Hier sind verschiedenste Konfigurationen vorzunehmen. In einem initialen Projekt sind neben der NLU Pipeline auch die verwendeten Policies für das Dialog Management definiert.

data/nlu.md
Als Grundlage für das Training des Natural Language Understanding (NLU) dienen die in diesem Dokument hinterlegten Daten. Es beinhaltet mögliche Intentionen des Benutzers. Dabei ist darauf zu achten, dass die einzelnen Intents im Format `## intent:<intent-name>` anzugeben sind (siehe Listing 3.1 Zeile 1). Die Beispieldaten der einzelnen Intents sind wie in den Zeilen 2-6 im Listing 3.1 vorzunehmen. Um ein korrektes Training zu ermöglichen sind pro Intent mindestens fünf Beispieldaten nötig.

```
1 ## intent:greet
2 - hey
3 - hello
4 - hi
5 - good morning
6 - good evening
```

Listing 3.1: Beispiel für data/nlu.md in Rasa

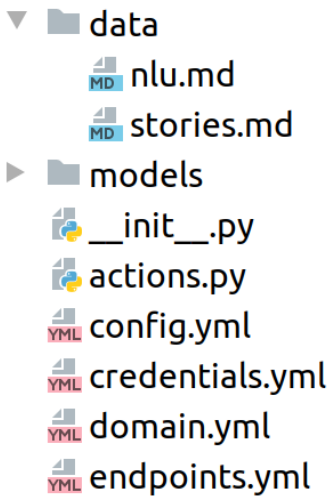


Abbildung 3.8: Datenstruktur eines initialen Rasa Projekts

data/stories.md
Trainingsdialoge für das Dialog Management werden in Form von diesem Markdown erstellt. Der Beginn eines Dialogs, in Rasa auch als Story bezeichnet, wird durch deren Namen in Form einer Überschrift vom Typen H2 gekennzeichnet (siehe Listing 3.2 auf der nächsten Seite Zeile 1). Benutzerintentionen werden gekennzeichnet über einen Stern (siehe Zeile 2 und Zeile 4) und Aktionen des Bots über einen eingerückten Bindestrich (siehe Zeile 3 und Zeile 5).

```
## happy path
* greet
  - utter_greet
* mood_great
  - utter_happy
```

Listing 3.2: Beispiel Story in Rasa

Auf die weiteren Dateien wird nicht eingegangen, da sie für ein grundsätzliches Verständnis der Funktionsweise nicht benötigt werden. Informationen zu den zusätzlichen Dateien und eine ausführlichere Beschreibung der Anpassungsmöglichkeiten der beschriebenen Dateien ist in der offiziellen Dokumentation von Rasa unter `https://rasa.com/docs/rasa/` nachzulesen.

Es ist aber auch möglich eigene Komponenten zu implementieren und zu den bestehenden hinzuzufügen. So ist die Implementierung eigener Interpreter, sogenannter Featurizer, Policies und Actions möglich. Dadurch dass es sich um ein Open Source Projekt handelt, ist es auch möglich die Umsetzung der verwendeten Komponenten zu analysieren und gegebenenfalls an die eigenen Gegebenheiten anzupassen. Aufgrund dieser genannten positiven Eigenschaften dieses Frameworks wird im weiteren Verlauf der Arbeit für Implementierungen Rasa genutzt.

4 Verwendete Korpora

Dieses Kapitel beschäftigt sich mit der Auswahl der Datenquellen für die Trainingsdaten der späteren Experimente, welche innerhalb der Evaluierung diskutiert werden. Die Evaluation der in Kapitel 5 auf Seite 20 beschriebenen Augmentierungen und in Kapitel 6 auf Seite 31 dargestellten Lernvarianten erfolgt unter Verwendung des Rasa Frameworks. Bei Auswahl der genutzten Korpora wird darauf geachtet, dass diese bereits im benötigten Format vorliegen.

Es werden zwei verschiedene Datenquellen verwendet. MultiWOZ 2.1 Korpus [10] liegt bereits im für Rasa geeigneten Format vor [72]. Der zweite Korpus beinhaltet 196 Dialoge aus zwei sich überschneidenden Domänen. Die Dialoge wurden teilweise durch Menschen erstellt und andere wurden synthetisch generiert [71]. Auch dieser Korpus liegt bereits im korrekten Datenformat vor.

4.1 MultiWOZ 2.1

Bei dem MultiWOZ Korpus [10, 16] handelt es sich um einen Korpus, der Mensch-Maschine Kommunikation beinhaltet und mit der Wizard-of-Oz Methode erzeugt wurde. Der ursprüngliche Korpus beinhaltet insgesamt 10.438 Dialoge, wovon 3.406 Dialoge aus einer einzigen Domäne und 7.032 Dialoge aus mehreren Domänen bestehen. Es sind zwei bis fünf Domänen pro Dialog vorzufinden [10]. Er beinhaltet Dialoge zwischen Gästen einer Touristeninformation und einem Wizard. Der Korpus beinhaltet die Domänen Sehenswürdigkeiten, Krankenhaus, Polizei, Hotel, Restaurant, Taxi und Zug. Die Domänen verfügen über Schnittmengen. So verfügen alle Domänen über die Entitäten Adresse, Postleitzahl und Telefonnummer. Es gibt auch Entitäten, die in einer Teilmenge der Domänen vorzufinden sind. Ein Beispiel für eine solche Entität ist das Ziel, welches in den Domänen Taxi und Zug aufzufinden ist. Diese Schnittmengen lassen sich auch bei den Intentionen der Benutzer und Aktionen des Assistenz Systems finden.

Es ist ein bereits für Rasa aufbereiteter Korpus vorhanden, welcher auf dem MultiWOZ Korpus basiert [72]. Dieser beinhaltet 9.061 Dialoge, wobei 7.249 für das Training und 1.812 für Tests vorgesehen sind. Es wurden 1.377 Dialoge aus dem ursprünglichen Korpus entfernt, da sie sich als unvollständig herausstellten. Die Ersteller dieses Korpus nutzen ihn für die Evaluierung ihrer Ergebnisse, jedoch weisen sie darauf hin, dass MultiWOZ 2.1 über Schwächen verfügt. Sie weisen darauf hin, dass MultiWOZ vermutlich nahezu historienunabhängig ist. Das bedeutet, dass eine vom Assistenten gewählte Aktion nicht zwingend von vorherigen Aktionen und Intentionen des Benutzers direkt abhängt, sondern nahezu ausschließlich von der letzten vorhergegangenen Intention des Benutzers. Diese Aussage untermauern sie, indem sie das Training des Dialog Management in zwei Ausprägungen durchführen. Einmal wird das Training mit einer Historie von 2 Interaktionen zwischen Benutzer und Assistenten durchgeführt. Das andere mal mit einer Historie von 10 Interaktionen. Historie bedeutet in diesem Kontext, dass dem Dialog Management zur Ermittlung der nächsten auszuführenden Aktion die letzte ermittelte Intention des Benutzers und die letzten *n* durchgeführten Interaktionen zwischen dem Assistenten und dem Benutzer zur Verfügung stehen. Die Genauigkeit wich in den beiden gewählten Ausprägungen um maximal 4% ab. Es wurden bei einer Historie von 2 Genauigkeiten zwischen $61\% - 69\%$ und bei einer Historie von 10 Genauigkeiten zwischen $61\% - 73\%$ festgestellt. Was unter historienunabhängig zu verstehen ist, lässt sich anhand eines Beispiels darstellen (siehe Abb. 4.1 auf der nächsten Seite).

Die linke Spalte zeigt den Dialog, wie er in den Trainingsdaten hinterlegt ist. Die rechte Spalte stellt den während der Tests durch das Dialog Management prognostizierten Dialog da. Es zeigt sich, dass die Aktionen des Assistenten, dargestellt durch den grauen Text über den Antworten des Assistenten, in ihrer Reihenfolge vertauscht sind. Auch sind die prognostizierten Aktionen an dieser Stelle formal nicht falsch. Sie weichen lediglich von den erwarteten ab. Benutzer könnten dies als korrekt deuten.

Abbildung 4.1: MultiWOZ 2.1 Dialog: Vergleich zwischen Ziel (links) und prognostiziertem Dialog (rechts) (vgl. [72]). Es ist zu sehen, dass durch das Dialog Management prognostizierte Aktionen im Vergleich zum Ziel in ihrer Reihenfolge vertauscht sind.

Trotz bekannter Problematik in diesem Korpus wird dieser für die Evaluation verwendet. Es erfolgt eine Analyse der Ergebnisse, um festzustellen, inwieweit den Resultaten aus [72] zuzustimmen ist und, ob eine Augmentierung zu signifikant besseren Ergebnissen führen kann, beziehungsweise wie aussagekräftig diese ist. Offen bleibt, ob dieser Korpus für eine vollständige Evaluation geeignet ist. Diese Fragestellung wird nach ersten Experimenten evaluiert.

4.2 Generierte Daten mit zwei Domänen

Neben MultiWOZ 2.1 findet in [72] noch ein zweiter Korpus Anwendung. Es handelt sich um den in [71] eingeführten Korpus. Dieser beinhaltet zwei Domänen. Die Domäne Hotel beinhaltet 89 Trainings- und 30 Testdialoge. 58 weitere Trainingsdialoge sind der Restaurant Domäne zuzuordnen. Somit beinhaltet der gesamte Korpus 177 Dialoge. Diese sind in 147 Trainingsdialoge und 30 Testdialoge geteilt.

Die Dialoge wurden in zwei Stufen erstellt. Zunächst erstellten die Autoren händisch 11 kooperative Dialoge in der Hotel-Domäne. Diese Dialoge dienen als Basis für die Generierung. Mit dieser Methode konnten 108 neue unkooperative, eindeutige Dialoge erzeugt werden. Die generierten Dialoge wurden in ein Trainingsset von 78 Dialogen und ein Testset von 30 Dialogen aufgeteilt. Die Restaurant Domäne beinhaltet 8 kooperative und 50 unkooperative generierte Dialoge.

5 Algorithmen der Augmentierung

Im Abschnitt 3 auf Seite 3 Stand der Technik dieser Arbeit wurden unter anderem Methoden der Augmentierung dargestellt, welche bereits implementiert und diskutiert wurden. In diesem Kapitel werden Ideen dieser Verfahren aufzugreifen und teilweise erweitert. In diesem Abschnitt werden drei Algorithmen zur Augmentierung beschrieben und im Laufe der Arbeit evaluiert. Es wird ein Algorithmus zur Augmentierung anhand von Vertauschungen beschrieben. Des Weiteren wird dem Dialog Chitchat hinzugefügt und neue Dialoge werden anhand eines Graphen generiert.

Bevor auf die Algorithmen und die unterschiedlichen Ausprägungen eingegangen werden kann ist zunächst auf die Struktur eines Dialogs und die grundlegenden Elemente einzugehen. Dies erfolgt, um eine formale Beschreibung der Algorithmen zu ermöglichen.
Die Menge D beschreibt die Gesamtmenge aller verfügbaren Dialoge. Ein einzelner Dialog wird als D_i bezeichnet. Somit gilt

$$D = \{D_1, D_2, ..., D_n\}\text{ , wobei gilt } n \in \mathbb{N} \tag{5.1}$$

Ein Dialog D_i im einzelnen besteht aus einer Abfolge von einzelnen Interaktionen zwischen dem Assistenten und dem Benutzer. Eine Interaktion wird als $d_{i,j}$ definiert. Ein Dialog wird einerseits durch die Interaktionen und andererseits durch die eindeutige Reihenfolge der aufeinander folgenden Interaktionen definiert. Somit kann der Dialog D_i wie folgt beschrieben werden:

$$D_i = (d_{i,1}, d_{i,2}, ..., d_{i,m}), \text{ wobei gilt } m \in \mathbb{N} \tag{5.2}$$

Bei einem $d_{i,j}$ kann es sich entweder um eine Intention des Benutzers, oder um eine vom Assistenten ausgeführte Aktion handeln. Die Menge I beinhaltet alle möglichen Intentionen des Benutzers und die Menge A alle vom Assistenten ausführbaren Aktionen. Die disjunkte Vereinigung dieser beiden Mengen A und I bildet alle möglichen Interaktionstypen M. Für eine Interaktion $d_{i,j}$ gilt

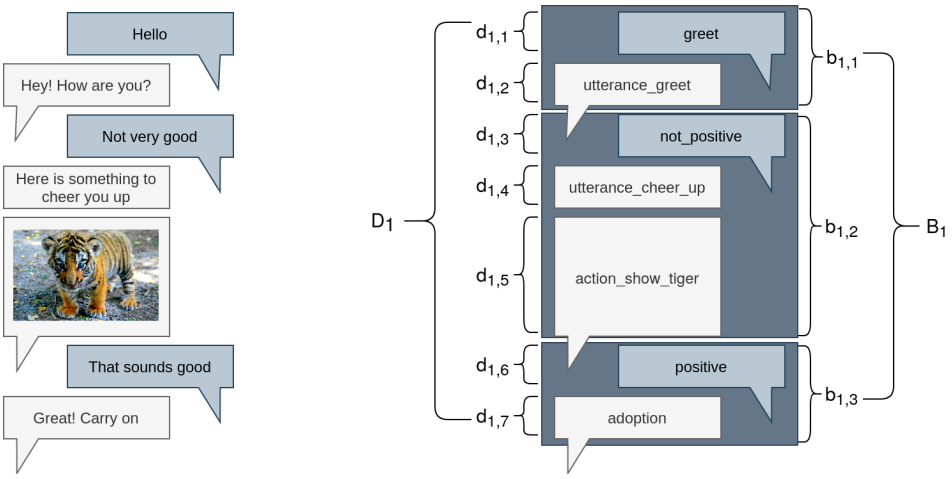
$$\begin{aligned} d_{i,j} &\in M \\ M &= A \dot{\cup} I \end{aligned} \tag{5.3}$$

Diese Formalisierungen lassen sich anhand eines Beispiels anschaulich darstellen. Hierzu wird der Dialog D_1 betrachtet. Dieser Dialog, wie er zwischen Mensch und Assistenten durchgeführt wird, ist der Abbildung 5.1a auf der nächsten Seite (links) zu entnehmen. Die formalisierte Darstellung dieses Dialogs lässt sich der Abbildung 5.1b auf der nächsten Seite (rechts) entnehmen. In dieser Darstellung wurden unter anderem die Eingaben des Benutzers durch die technische Bezeichnung der Intentionen ersetzt. Gleiches erfolgte bei den Aktionen des Assistenten. Der dargestellte Dialog beinhaltet sieben Interaktionen und lässt sich in der oben eingeführten Weise darstellen durch

$$D_1 = (d_{1,1}, d_{1,2}, \ldots, d_{1,7}) \tag{5.4}$$

Weiter können die Interaktionen aufgeteilt werden in Intentionen und Aktionen:

$$\begin{aligned} d_{1,1}, d_{1,3}, d_{1,6} &\in I \\ d_{1,2}, d_{1,4}, d_{1,5}, d_{1,7} &\in A \end{aligned} \tag{5.5}$$



(a) Durchgeführter Dialog (b) Formalisierte Darstellung

Abbildung 5.1: Beispieldialog zur Erläuterung der Bezeichnungen und des Zusammenhangs zwischen Interaktionen und Blöcken. Linksbündig dargestellte Sprechblasen sind Aktionen des Assistenten und rechtsbündig dargestellte Sprechblasen Intentionen des Benutzers.

Es obliegt allen in dieser Arbeit betrachteten Dialogen eine grundlegende Struktur.

1. Ein Dialog hat immer mit einer Intention des Benutzers zu beginnen.
2. Auf eine Intention folgt immer eine Aktion des Assistenten.
3. Auf eine Aktion des Assistenten folgt eine Intention oder eine Aktion oder das Ende des Dialogs.

Formal gilt für alle $D_i \in D$:

$$\begin{aligned} &d_{i,1} \in I \\ &d_{i,j} \in I \wedge d_{i,j\pm1} \in A, \textit{ wenn } (j+1) \geq |D_i| \\ &d_{i,j} \in A \wedge d_{i,j\pm1} \in M, \textit{ wenn } (j+1) \geq |D_i| \end{aligned} \tag{5.6}$$

Durch diese vorgegebene Struktur der Dialoge können Blöcke $b_{i,k}$ definiert werden. Der Beispieldialog aus aus Abbildung 5.1b beinhaltet die drei Blöcke $b_{1,1}$, $b_{1,2}$ und $b_{1,3}$. Diese werden in der Blockmenge B_1 zusammengefasst.Ein Block beginnt mit einer Intention und beinhaltet alle darauf folgenden Aktionen bis zur nächsten Intention, wobei diese den Beginn des nächsten Blocks darstellt. Ein Block $b_{i,k}$ ist wie folgt zu definieren:

$$\begin{aligned} b_{i,k} &= (d_{i,p}, d_{i,p+1}, \ldots, d_{i,q}) \\ \textit{wobei gilt: } &d_{i,p} \in I \\ &\{d_{i,p+1}, \ldots, d_{i,q}\} \in A \\ &q > p \\ &(d_{i,q+1} \in I \vee q+1 \geq |D_i|) \end{aligned} \tag{5.7}$$

Daraus folgernd ergibt sich, dass die Konkatenation aller Blöcke, dargestellt als $\wr$, eines Dialogs wieder den Dialog selbst ergibt. Es ist dabei auf die korrekte Reihenfolge der Blöcke zu achten.

Somit gilt:

$$\begin{aligned} &\bigcirc\!\!\!\!\bigcup_{k=1}^{|B_i|} b_{i,k} \\ &= (d_{i,j} \mid j = 1, \ldots, n \wedge n = |D_i|) \\ &= D_i \end{aligned} \tag{5.8}$$

Es ist dem Benutzer per Definition nicht möglich zwischen zwei Aktionen des Assistenten innerhalb eines Blocks eine Intention einzufügen. Diese Möglichkeit wird per Definition ausgeschlossen. Dies liegt daran, dass Assistenz Systeme das Tätigen einer Intention nur dann ermöglichen, wenn dies explizit vorgesehen ist. Sonderfälle, wie das Abbrechen von Aktionen, wie beispielsweise bei dem Sprachassistenten Alexa durch den Befehl “Alexa Stopp”, werden hierbei explizit ausgeschlossen.

Aufbauend auf diesen Definitionen können Algorithmen beschrieben werden, welche zur Augmentierung der Dialoge dienen.

5.1 Vertauschung

Mit der Permutation [14] wurde bereits im Abschnitt 3.5 auf Seite 11 eine Variante der Augmentierung beschrieben. Dabei werden zwei zufällig gewählte Interaktionen innerhalb des Dialogs miteinander vertauscht. Für die dort augmentierten Dialoge $D_i \in D$ gilt $|D_i| \geq 3$. Es wurden zwei Varianten der Permutation beschrieben. In der ersten Variante wird die letzte mit der vorletzten Interaktion vertauscht. So wird der Dialog

$$D_1 = (d_{1,1}, d_{1,2}, d_{1,3}, d_{1,4}) \tag{5.9}$$

durch die Permutation zu

$$D_2 = (d_{1,1}, d_{1,2}, d_{1,4}, d_{1,3}) \tag{5.10}$$

In diesem Beispiel haben $d_{1,3}$ und $d_{1,4}$ ihre Position innerhalb des Dialogs getauscht. In der zweiten Variante der Permutation tauschen die vorletzte und drittletzte Interaktion ihre Positionen. Es wird also der Dialog

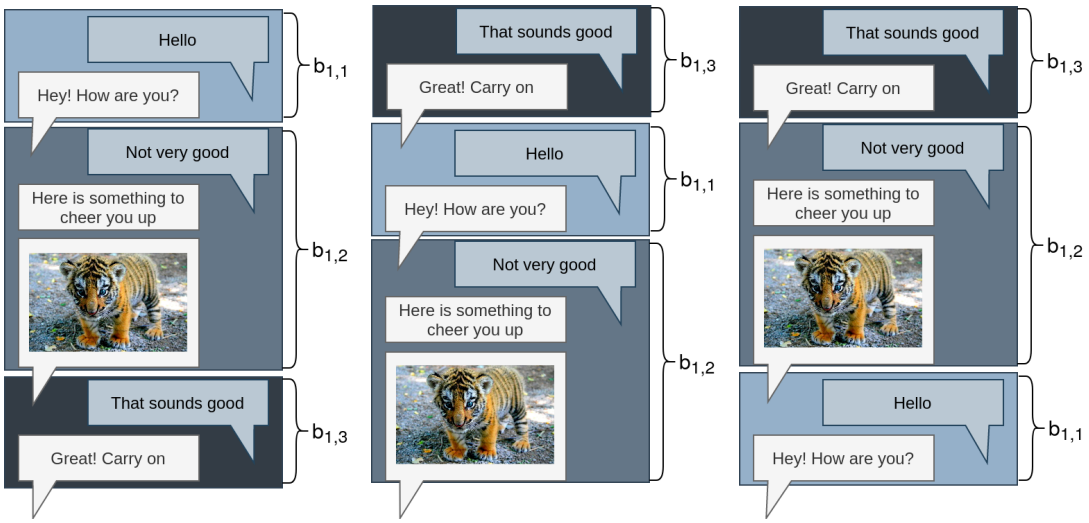
$$D_1 = (d_{1,1}, d_{1,2}, d_{1,3}, d_{1,4}) \tag{5.11}$$

durch diese Variante der Permutation zu

$$D_3 = (d_{1,1}, d_{1,3}, d_{1,2}, d_{1,4}) \tag{5.12}$$

Somit haben $d_{1,2}$ und $d_{1,3}$ ihre Positionen getauscht.

Die Idee der Permutation wird in dieser Variante der Augmentierung aufgegriffen und angepasst. Bei diesem Verfahren handelt es sich um ein sehr einfaches mit geringer Komplexität, um die Anzahl der verfügbaren Trainingsdaten um ein Vielfaches zu erhöhen. Da sich die betrachteten Dialoge im Gegensatz zu denen in [14] in Blöcke unterteilen lassen, wird nicht die Reihenfolge der Interaktionen getauscht, sondern die der Blöcke. Dadurch kann sichergestellt werden, dass auch für die augmentierten Dialoge die in (5.6) auf Seite 21 definierten Bedingungen gelten. Im Weiteren wird zunächst die einfache Vertauschung und darauf aufbauend die mehrfach Vertauschung als Algorithmus beschrieben.



(a) Ursprünglicher Dialog ohne Augmentierung

(b) Einfache Vertauschung mit $k = 2$

(c) Mehrfach Vertauschung mit $k = 3$

Abbildung 5.2: Vertauschungen anhand des Beispieldialogs D_1 aus Abb. 5.1 auf Seite 21

5.1.1 Einfache Vertauschung

Bei der einfachen Vertauschung wird der Dialog D_i in die zwei Teildialoge D_i^1 und D_i^2 unterteilt. Für diese Teildialoge gilt

$$\begin{aligned} D_i^1 &= (b_{i,1}, \ldots, b_{i,m}) \\ D_i^2 &= (b_{i,m+1}, \ldots, b_{i,n}) \\ n &= |B_i|; 1 \leq m < n \end{aligned} \tag{5.13}$$

Die einfach Vertauschung erfolgt, indem D_i^1 und D_i^2 in ihrer ursprünglichen Reihenfolge vertauscht werden und in D_j konjugiert werden.

$$\begin{aligned} D_j &= D_i^2 \circ D_i^1 \\ &:= (b_{i,m+1}, \ldots, b_{i,n}, b_{i,1}, \ldots, b_{i,m}) \end{aligned} \tag{5.14}$$

Durch die einfache Vertauschung kann eine neue Menge an Dialogen D^{neu} erzeugt werden, welche die ursprünglichen Dialoge und die neu erzeugten Dialoge beinhaltet. Im Vergleich zu der ursprünglichen Menge D sind in D^{neu} doppelt so viele Dialoge enthalten, da D^{neu} die ursprünglichen Dialoge und die vertauschten Dialoge beinhaltet. Dabei ist davon auszugehen, dass pro Dialog lediglich eine Vertauschung vorgenommen wird. Es ist auch denkbar, dass die Vertauschung mit unterschiedlichen Werten von m durchgeführt wird.

Eine beispielhafte einfache Vertauschung für den in Abb. 5.1 auf Seite 21 dargestellten Dialog D_1 ist Abb. 5.2b entnehmbar. In diesem Beispiel wurde $m = 2$ gewählt, wodurch sich der neue Dialog beschreiben lässt durch $(b_{1,3}, b_{1,1}, b_{1,2})$.

5.1.2 Mehrfach Vertauschung

Bei der mehrfachen Vertauschung wird der Dialog D_i in k Teildialoge $(D_i^1, \ldots, D_i^k)$ unterteilt und in eine neue Reihenfolge gebracht, wobei $k \leq \mid B_i \mid$ gilt. Durch diese Methode sind bis zu $(!k - 1)$ neue Dialoge pro Dialog D_i generierbar.

Für den Dialog $D_1 = (b_{1,1}, b_{1,2}, b_{1,3})$, wie er in Abbildung 5.1 auf Seite 21 als Beispiel eingeführt wurde, lassen sich folgende neue Reihenfolgen dadurch erzeugen:

$$\begin{aligned} &(b_{1,1}, b_{1,3}, b_{1,2}) \\ &(b_{1,2}, b_{1,1}, b_{1,3}) \\ &(b_{1,2}, b_{1,3}, b_{1,1}) \\ &(b_{1,3}, b_{1,1}, b_{1,2}) \\ &(b_{1,3}, b_{1,2}, b_{1,1}) \end{aligned} \tag{5.15}$$

Beispielhaft wird der Dialog $(b_{1,3}, b_{1,2}, b_{1,1})$ in der Abbildung 5.2c auf der vorherigen Seite dargestellt. Welches k zu guten Ergebnissen führt und ab welchem Wert es zu keinen sinnvollen Ergebnissen mehr führt, ist innerhalb der Evaluation festzustellen. $k \approx \mid B_i \mid$ kann dazu führen, dass es dem Dialog Management nicht mehr möglich ist zu korrekten Ergebnissen zu gelangen, da in den augmentierten Trainingsdaten kein Zusammenhang zwischen den einzelnen Blöcken approximiert werden kann. Es ist festzustellen, welches k zu wählen ist, um verbesserte Ergebnisse im Dialog Management zu erlangen.

5.2 Chitchat einfügen

In [79] wird bereits ein Verfahren beschrieben, welches durch das Einfügen von Chat-Daten in den Trainingsdialogen zu besseren Ergebnissen beim Vergleich mit Trainingsdaten und auch bei der Nutzung durch Menschen führt. Die Idee der Augmentierung durch das Hinzufügen von Chat-Daten verfolgt die Hoffnung die Daten so anzureichern, dass es dem Dialog Management möglich wird unnötige Informationen zu erkennen und diese für die Ermittlung der weiteren Aktionen zu ignorieren.

Der hier beschriebene Algorithmus wird nicht auf einen Chat-Korpus zurückgreifen, sondern sogenanntes Chitchat, zu deutsch Plauderei oder Geschwätz, in den Dialog einfügen. Der Chitchat Korpus D_c ist zu beschreiben als:

$$D_C = ((q_0, r_0), \ldots, (q_p, r_p)) \text{ wobei gilt } p \geq 1; p \in \mathbb{N} \tag{5.16}$$

Bei dem Tupel (q_p, r_p) handelt es sich um den Chitchat q_p des Benutzers und die Reaktion r_p des Assistenten auf diesen. Es gilt $q_p \in I$ und $r_p \in A$. Neben der Variation der Anzahl an Tupeln besteht die Möglichkeit nach dem Chitchat eine Rückführung durchzuführen (siehe Abb. 5.3c auf der nächsten Seite) oder nicht (siehe Abb. 5.3b auf der nächsten Seite). Bei der Rückführung handelt es sich um die Wiederholung der letzten Aktionen des Assistenten bevor der Benutzer mit Chitchat reagierte. Die Idee bei der Rückführung besteht darin, dem Benutzer eine bessere Nutzererfahrung zu ermöglichen. In einer Kommunikation zwischen zwei Menschen ist es auch üblich, nach einer Unterbrechung die letzte gestellte Frage nochmals zu wiederholen. Dieses Verhalten wird durch die Rückführung nachgeahmt.

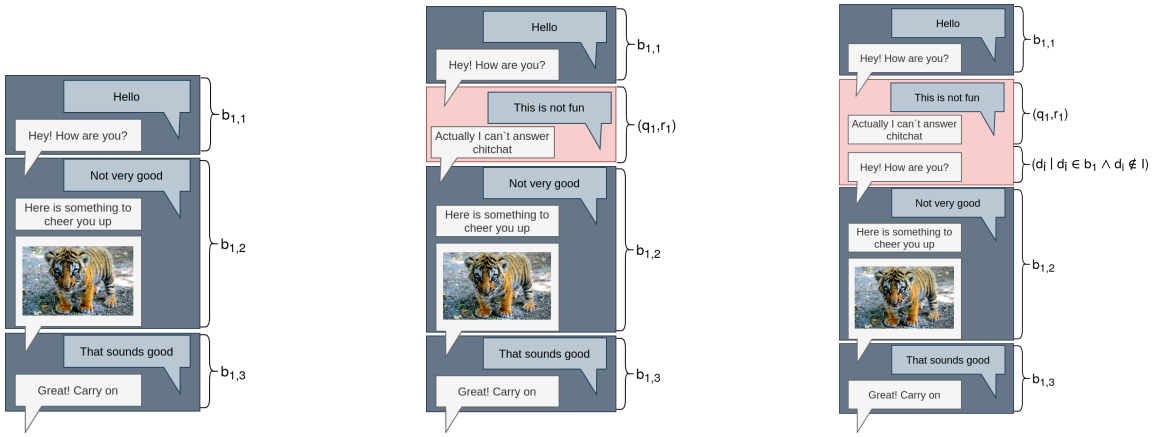
5.2.1 Ohne Rückführung

Zunächst wird der Algorithmus beschrieben, ohne eine Rückführung durchzuführen. Per Definition wird Chitchat nur zwischen zwei Blöcken $b_{i,n}$ und $b_{i,n+1}$ eingefügt. Dies lässt sich begründen, indem es per Definition dem Benutzer nicht möglich ist zwischen zwei Aktionen des Assistenten, welche innerhalb eines Blocks ausgeführt werden, eine Intention einzufügen. Der neu augmentierte Dialog wird als D_i^C bezeichnet. Durch diese Bedingung ergibt sich für das Einfügen von Chitchat

(q_p, r_p) an einer Stelle q in den Dialog D_i folgende Beschreibung.

$$\begin{aligned} D_i^C &= (b_{i,1}, \ldots, b_{i,q}, (q_p, r_p), b_{i,q+1}, \ldots, b_{i,n}) \\ \textit{wobei gilt: } n &= \mid B_i \mid; 1 \leq q < n; (q_p, r_p) \in D_C; q \in \mathbb{N} \end{aligned} \tag{5.17}$$

In Abbildung 5.3b wurde dem Dialog D_1 das Chitchat (q_1, r_1) (q_1 = “This is not fun”, r_1 = “Actually I can’t answer chitchat”) für $q = 1$ eingefügt. Der Vorgang des Einfügens von Chitchat kann pro Dialog D_i mehrfach durchgeführt werden. Maximal kann hinter jeden Block, der letzte Block ausgeschlossen, einmal Chitchat eingefügt werden. Die daraus resultierenden Dialoge werden dargestellt durch $D^{C,n}$. Bei n handelt es sich um die Anzahl des eingefügten Chitchats im Dialog. Bei dem in Abbildung 5.3b dargestellten Dialog handelt es sich um einen Dialog, welcher als $D_1^{C,1}$ bezeichnet wird.



(a) Dialog ohne Augmentierung (b) Ohne Rückführung (c) Mit Rückführung

Abbildung 5.3: Die beiden Varianten von Chitchat

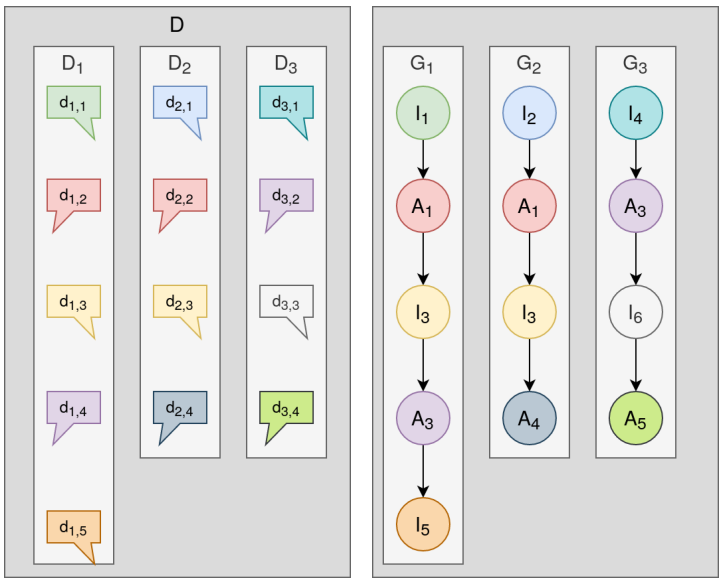
5.2.2 Mit Rückführung

Um dem Benutzer zu symbolisieren, auf welche Aktion der Assistent nach dem Chitchat eine Intention erwartet, ist es sinnvoll eine Rückführung in Form einer Wiederholung der letzten Aktion des vorherigen Blocks durchzuführen. Dialoge, welche mit diesem Algorithmus augmentiert sind, werden als $D^{Cr,n}$ dargestellt. Ein Beispiel einer solchen Rückführung wird in Abbildung 5.3c gezeigt. Dieser Dialog unterscheidet sich von dem in Abbildung 5.3b darin, dass innerhalb des rot markierten Blocks, dem Chitchat, auch die Interaktion $d_{1,2}$=“Hey! How are you?”, die letzte Aktion des letzten Blocks b_1, beinhaltet ist.

5.3 Generierung neuer Dialoge

In diesem Abschnitt wird ein Algorithmus erläutert, welcher es ermöglicht aus einer bestehenden Dialogmenge D neue Dialoge zu generieren. Dies erfolgt indem Zusammenhänge zwischen den einzelnen Dialogen ermittelt werden. Diese Zusammenhänge ermöglichen es Dialoge zu generieren um so die Menge der Trainingsdialoge zu vervielfachen. Zunächst wird ein Graph erstellt, welcher den Zusammenhang zwischen den einzelnen Interaktionstypen der Dialogmenge darstellt. Anhand dieses Graphen sind neue Dialoge zu generieren. Dieser Algorithmus wird in zwei Varianten beschrieben. Zunächst erfolgt lediglich auf Basis des Graphen die Generierung. Darauf aufbauend wird ein Algorithmus beschrieben, welcher mit einer bekannten Wahrscheinlichkeit zufällige Wendungen in den generierten Dialogen einfügt, um so die Variabilität der Dialoge zu

erhöhen.



(a) Dialogmenge D = $\{D_1, D_2, D_3\}$ (b) Abgebildete Graphen G_1, G_2, G_3

Abbildung 5.4: Abbildung der Dialogmenge D auf die Graphen G_i

Um die Erstellung des bereits genannten Graphen zu erläutern ist zunächst die Dialogmenge D zu betrachten. Eine solche Dialogmenge ist in der Abbildung 5.4a dargestellt. Die Dialogmenge D beinhaltet in diesem Beispiel die drei Dialoge D_1, D_2 und D_3, welche jeweils vier beziehungsweise fünf Interaktionen beinhalten. Jede Interaktion $d_{i,j} \in D_i \in D$ ist eindeutig einem Element der Menge der möglichen Interaktionstypen M zuordenbar. Dies wurde in der Abbildung 5.4a durch die Einfärbung der Interaktionen visualisiert. So lassen sich beispielsweise $d_{1,4}$ und $d_{3,2}$ dem Interaktionstypen $A_3 \in A \subset M$ zuordnen.

Da jede Interaktion $d_{i,j}$ einem eindeutigen Interaktionstypen zugeordnet werden kann und eine feste Reihenfolge der Interaktionen in einem Dialog besteht, kann der Dialog D_i auch als gerichteter Graph von Interaktionstypen dargestellt werden. Der gerichtete Graph des Dialogs D_i wird als G_i bezeichnet und ist in der Abbildung 5.4b zu sehen.

Um Zusammenhänge zwischen den einzelnen Dialogen festzustellen, um so neue Dialoge zu erstellen ist der Graph $G = (V, E)$ zu bilden. Bei V handelt es sich um die Knoten und bei E um die Kanten des Graphen. Dies erfolgt, indem die Graphen G_i, welche aus der Dialogmenge D gebildet werden, zum Graphen G aufsummiert werden.

$$G = \sum_{i=1}^{|D|} G_i \tag{5.18}$$

Werden die Graphen G_1, G_2 und G_3 der Abbildung 5.4b zum Graphen G summiert, ergibt sich der Graph G, wie er in der Abbildung 5.5 auf der nächsten Seite dargestellt wird. Dieser Graph G ist im Weiteren für die Generierung neuer Dialoge zu nutzen. Vor der Generierung selbst, werden noch Knoten und deren Bedeutung in den Dialogen erläutert.

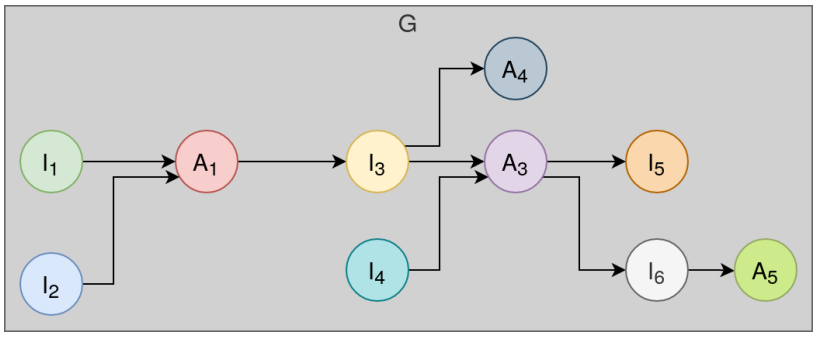


Abbildung 5.5: Graph G abgeleitet von der Dialogmenge D

Es erfolgt die Betrachtung sechs verschiedener Arten von Knoten. Sie unterscheiden sich jeweils in der Anzahl der eingehenden Kanten $d^-(v)$ und ausgehenden Kanten $d^+(v)$. Bei v handelt es sich um den Knoten, wobei $v \in V$ gilt.

Zuerst sind die Knoten zu beachten, welche den eindeutigen Anfang beziehungsweise das Ende von Dialogen darstellen. Diese zeichnen sich dadurch aus, dass gilt $d^-(v) = 0$ beziehungsweise für einen Endknoten gilt $d^+(v) = 0$. In der Graphentheorie werden solche Knoten auch als Quelle, wenn $d^- = 0$ gilt, und Senke, wenn $d^+(v) = 0$ gilt, bezeichnet [67]. Beispiele hierfür sind den Abbildungen 5.6b und 5.6a zu entnehmen. Im in 5.5 abgebildeten Graphen G weist jeder Start- beziehungsweise End-Knoten der ursprünglichen Graphen G_1, G_2 und G_3 auch die Eigenschaft des Start- beziehungsweise End-Knoten auf. Diese Eigenschaft ist jedoch bei komplexeren Graphen nicht grundsätzlich gegeben. So ist es denkbar, dass Interaktionstypen, welche in verschiedenen Dialogen vorzufinden sind, in einem Dialog als Start-Knoten dienen und in anderen Dialogen mitten im Dialog vorzufinden sind. Diese Eigenschaft gilt auch für End-Knoten. Bei der Definition der Algorithmen zur Erzeugung neuer Dialoge ist diese Eigenschaft mit zu bedenken.
Neben diesen Knoten gibt es Knoten, an denen eine eingehende und eine ausgehende Kante vorzufinden ist (siehe 5.6c). Daneben sind Knoten zu finden, welche Entscheidungspunkte in den Dialogen darstellen, für die gilt $d^+(v) = 1$ und $d^-(v) > 1$ (siehe 5.6d). Neben diesen Entscheidungspunkten zeigen sich auch zusammenführende Knoten, für die gilt $d^+(v) > 1$ und $d^-(v) = 1$. Ein solcher Knoten ist mit A_1 in 5.6e abgebildet. Letztlich sind noch Knoten zu nennen, welche zusammenführend und auch als Entscheidungspunkt dienen. Im dargestellten Beispiel ist A_3 ein solcher Knoten.

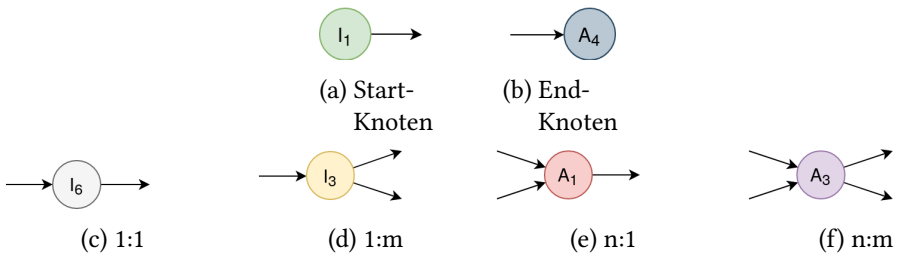


Abbildung 5.6: Varianten der möglichen Knoten im Graphen G

Es lassen sich Schlüsse über die Struktur und Komplexität der Dialoge in der Dialogmenge D ziehen, in welcher Anzahl die einzelnen Varianten von Knoten in G vorkommen. Lassen sich viele Knoten, wie in 5.6d, 5.6e und 5.6f finden, deutet dies auf eine hohe Komplexität und Variabilität innerhalb der Dialoge hin.

Nachdem der Graph G definiert und Varianten der einzelnen Knoten dargelegt wurden, erfolgt mit der *Generierung anhand des Graphen G* und der *Generierung anhand des Graphen G mit dem Zufallsfaktor r* die Erläuterung zweier Algorithmen zur Erstellung neuer Dialoge.

5.3.1 Generierung anhand des Graphen G

Die Generierung neuer Dialoge basiert ausschließlich auf dem Graphen G. Durch dieses Verfahren können neben allen ursprünglichen Dialogen auch weitere Dialoge generiert werden, welche sich aus der Struktur von G ergeben.

Die Hypothese hinter diesem Verfahren besteht darin, dass grundsätzlich alle Dialoge, welche durch den Graphen G abbildbar sind, auch syntaktisch korrekt sind. Dies ist darin begründet, dass G die Summe aller einzelnen Graphen G_i ist und diese per Definition syntaktisch korrekt sind. Neue Dialoge lassen sich generieren, indem eine zufällige Intention als Startpunkt im Graphen gewählt wird und dieser von dort an durchlaufen wird und somit neue, syntaktisch korrekte Dialoge erzeugt werden. Geht man davon aus, dass der Graph alle möglichen Dialoge abbildet, so kann eine Vielzahl generierter Dialoge auf Basis dieses Graphen dazu führen, dass eine Approximation des Graphen durch die Algorithmen des Dialog Management erfolgt und damit bessere Testergebnisse erzielt werden können. Um diese Hypothese auf ihre Korrektheit zu prüfen, wird ein Algorithmus, wie unter 1 beschrieben, umgesetzt. Dieser kann für die Erstellung solcher Dialoge genutzt werden. Dem Algorithmus ist der Graph, die Anzahl der zu generierenden Dialoge und die Länge pro Dialog zu übergeben. Als Rückgabe erhält man die Menge D^{neu}, welche die generierten Dialoge enthält.

Algorithm 1 Generierung Anhand des Graphen G

```
1:  procedure GENERATE(Graph, numberOfDialogs, lengthPerDialog)
2:      D^neu ← ∅
3:      for i ← 1; i <numberOfDialogs; i ← i + 1 do
4:          D_i ← ∅
5:          j ← 1
6:          d_j ← getRandomNode(Graph)
7:          while j < lengthPerDialog do
8:              D_i ← D_i + d_j
9:              nextNodes ← getNextPossibleNodes(Graph, d_j)
10:             if nextNodes ≠ ∅ then
11:                 j ← j + 1
12:                 d_j ← chooseNextNode(nextNodes)
13:             else
14:                 break
15:         D^neu ← D^neu + D_i
16:     return D^neu
```

Ein Dialog mit der Länge 4, welcher mit diesem Algorithmus anhand des Graphen G aus Abbildung 5.5 auf der vorherigen Seite generiert werden kann, ist in der Abbildung 5.7 auf der nächsten Seite dargestellt. Als zufälliger Startknoten wurde I_1 gewählt. Im Dialog selbst wird I_1 auf $d_{4,1}$ abgebildet. Es handelt sich um einen Zufall, dass es sich bei I_1 um einen Startknoten im Graphen handelt. Es hätte auch jeder andere beliebige Knoten gewählt werden können. Um zur Veranschaulichung einen Dialog zu generieren, welcher nicht ein Teildialog eines bereits vorhandenen ist, wurde I_1 gewählt.

Es wurde ein Teilgraph generiert, welcher sich über die Kanten

$$\{(I_1, A_1), (A_1, I_3), (I_3, A_4)\} \tag{5.19}$$

beschreiben lässt. Dieser Teilgraph wird auf den Dialog $D_4 = (d_{4,1}, d_{4,2}, d_{4,3}, d_{4,4})$ abgebildet. Auf den Schritt der Abbildung wurde für eine bessere Lesbarkeit im Algorithmus (siehe Algorithmus 1 auf der vorherigen Seite) verzichtet.

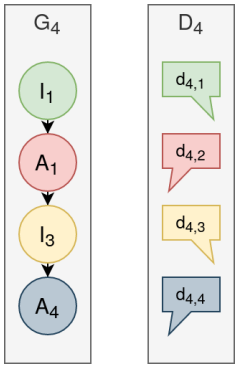


Abbildung 5.7: Generierter Dialog anhand des Graphen G

5.3.2 Generierung anhand des Graphen G mit dem Zufallsfaktor r

Mit dem im vorherigen Abschnitt beschriebenen Algorithmus wird das Ziel verfolgt, dass die Algorithmen des Dialog Managements den Graphen G approximieren. Es besteht die Möglichkeit, das dieses Verfahren dazu führt, dass eine Überanpassung (englisch Overfitting) erfolgt. Um dieser Hypothese entgegenzuwirken wird ein weiterer Algorithmus umgesetzt, welcher mit einer Wahrscheinlichkeit, bezeichnet als r, einen zufälligen nächsten Knoten wählt und nicht einen der Knoten, welcher anhand des Graphen G vorgesehen sind. Durch dieses Verfahren lassen sich Dialoge erzeugen, wie er in 5.8 abgebildet ist. Dieser Dialog beinhaltet eine Kante, welche im ursprünglichen Graphen G nicht beinhaltet ist. Es handelt sich um die Kante (A_3, I_2), hervorgehoben durch die rote Einfärbung in der Abbildung.

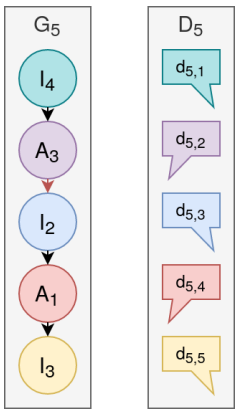


Abbildung 5.8: Generierte Dialoge anhand des Graphen G mit dem Zufallsfaktor r. Der Zufallsfaktor kommt zwischen den Aktionen A_3 und I_2 zu tragen. Hier hervorgehoben durch die rote Markierung.

Der Algorithmus 2 auf der nächsten Seite unterscheidet sich vom Algorithmus 1 auf der vorherigen Seite darin, dass mit dem Parameter `changeProbability` die Wahrscheinlichkeit r übergeben

wird. Für r gilt:

$$r \in \mathbb{R} : 0 < r \leq 1 \tag{5.20}$$

Die in (5.20) beschriebene Definition von r gilt auch für den Rückgabewert der Methode `random`. Diese Methode wird genutzt, um eine zufällige Zahl zu generieren, um so anhand der gegebenen Wahrscheinlichkeit in den Zeilen 9-11 die Auswahl eines zufälligen Knoten umzusetzen.

Algorithm 2 Generierung Anhand des Graphen G mit Zufallsfaktor r

```
1:  procedure GENERATE(Graph, changeProbability, numberOfDialogs, lengthPerDialog)
2:      D^neu ← ∅
3:      for i ← 1; i < numberOfDialogs; i ← i + 1 do
4:          D_i ← ∅
5:          j ← 1
6:          d_j ← getRandomNode(Graph)
7:          while j <lengthPerDialog do
8:              D_i ← D_i + d_j
9:              if random() ≤ changeProbability then        ▷ Random returns value in [0,1)
10:                 d_j ← getRandomNode(Graph)
11:             else
12:                 nextNodes ← getNextPossibleNodes(Graph, d_j)
13:                 if nextNodes ≠ ∅ then
14:                     j ← j + 1
15:                     d_j ← chooseNextNode(nextNodes)
16:                 else
17:                     break
18:         D^neu ← D^neu + D_i
19:     return D^neu
```

6 Lernvarianten

Im vorherigen Abschnitt wurden Augmentierungsalgorithmen dargestellt, um das Dialog Management mit identischer Menge an Trainingsdaten zu einer höheren Genauigkeit bei den Ergebnissen zu führen. Um Aussagen über die Algorithmen der Augmentierung zu treffen, erfolgt auch ein Vergleich zwischen den zwei Lernvarianten Supervised Learning und Reinforcement Learning. Hierzu wird zunächst auf die im Supervised Learning verwendete Policy und deren Konfiguration eingegangen. Im Weiteren erfolgt die Darstellung des implementierten Reinforcement Learning Ansatzes.

Beide Lernvarianten bieten verschiedene Einsatzmöglichkeiten. Supervised Learning eignet sich vor allem dann, wenn bereits Trainingsdaten im ausreichenden Maß vorhanden sind. Reinforcement Learning bietet die Möglichkeit Feedback von Benutzern in Form eines Rewards für zukünftiges Training zu nutzen. Der Schwerpunkt dieser Arbeit liegt nicht darin, die beiden Lernvarianten zu vergleichen, sondern Verbesserungen bei der Genauigkeit in den jeweiligen Lernvarianten durch verschiedene Arten der Augmentierung festzustellen.

Die Implementierung beider Varianten erfolgt, wie auch die Implementierung der Augmentierungsalgorithmen, unter Verwendung des Rasa Frameworks (siehe 3.6.3 auf Seite 15). Für die Nutzung dieses Frameworks spricht unter anderem die leichte Erweiterbarkeit, die Quelloffenheit und eine ausführliche Dokumentation. Für die Implementierung des Supervised Learning wird auf bereits vorhandene Policies zurückgegriffen.

6.1 Supervised Learning

Mit [72] beschreiben die Hauptentwickler des Rasa Frameworks die Transformer Embedding Dialogue Policy (TED-Policy). Es handelt sich um eine Dialog Management Policy, welche auf einer Transformer Architektur basiert. Die Policy basiert auf einem Unidirektional Transformer und Embedding Layers. Eine genaue Erläuterung der Architektur ist [72] zu entnehmen. Eine Erläuterung der Funktionsweise kann der offiziellen Dokumentation[1] des Frameworks entnommen werden.

Um eine Vergleichbarkeit der Experimente in der Evaluation zu erreichen, erfolgt eine einheitliche Konfiguration der Policy. Rasa ermöglicht die Nutzung mehrerer Policies für das Umsetzen des Dialog Management innerhalb eines Assistenten. Um jedoch Ergebnisse zwischen den unterschiedlichen Augmentierungen und Lernvarianten direkt miteinander zu vergleichen wird, auf diese Möglichkeit verzichtet. In allen im Supervised Learning vorgenommenen Experimenten ist die im folgenden Listing dargestellte Konfiguration vorzunehmen.

```
policies:
  - name: TEDPolicy
    max_history: 5
    epochs: 100
```

Listing 6.1: Konfiguration der TED-Policy für das Supervised Learning

Mit dem Parameter `max_history` wird der maximale zeitliche Bezug festgelegt. Mit der vor-

[1] `https://rasa.com/docs/rasa/core/policies/#ted-policy`

genommen Konfiguration ist festgelegt, dass Intentionen und Aktionen, welche mindestens fünf Nachfolger im Dialog aufweisen, irrelevant für die Entscheidung des Dialog Management sind. Mit dieser Konfiguration hätte der in Abbildung 5.4a auf Seite 26 dargestellte Dialog D_1 die maximale Länge, um vollständig in der Historie aufgenommen zu werden. Dialoge mit einer größeren Länge werden nicht vollständig für die Auswahl der aktuellen Aktionen in Betracht gezogen.

6.2 Reinforcement Learning

Neben dem Supervised Learning wird als zweite Lernvariante Reinforcement Learning umgesetzt. Als neuronales Netz wird auch bei dieser Lernvariante auf das Netz der TED-Policy zurückgegriffen. Das Reinforcement Learning wird mittels eines Deep-Q-Learning Algorithmus realisiert. Auch dieser Algorithmus beinhaltet ein Environment und einen Agenten, wie im Abschnitt 3.3.2 auf Seite 7 beschrieben. Der Agent erkundet mit den ihm möglichen Aktionen seine Umgebung, das Environment, und erhält dafür Feedback in Form von einem Reward. Der Agent optimiert seine Aktionen iterativ, um das bestmögliche Feedback zu bekommen. Eine ausführliche Beschreibung von Deep-Q-Learning kann [42, S. 5] entnommen werden. Der dort beschriebene Algorithmus wurde erstmals verwendet, um Atari Spiele zu erlernen, hat aber auch in anderen Domänen sehr gute Ergebnisse gezeigt. So konnten unter anderem positive Ergebnisse beim Einsatz für das autonome Fahren [45] und die Ermittlung von Entwicklungen auf dem Finanzmarkt [30] festgestellt werden. Durch die Unterscheidung in den beiden Umfeldern, Spielen von Atari Spielen und Dialog Management, sind Anpassungen des Algorithmus vorzunehmen. Im Folgenden sind die Anpassungen des generischen Deep-Q-Algorithmus an die Besonderheiten des Trainings an das Dialog Management beschrieben.

Environment reset und γ Faktor
Bei einem Atari Spiel besteht das Ziel darin möglichst viele Punkte zu erzielen, bevor das Spiel durch ein Game Over endet. Das Ziel Dialog Management besteht darin in Bezug auf die Historie und die letzte Intention des Benutzers, die korrekte folgende Aktion zu ermitteln. Ein Game Over bei einem Spiel begründet sich in einer Vielzahl von verschiedenen Fehlentscheidungen. Nicht ausschließlich die letzte Entscheidung führte zu diesem Game Over. Es erfolgten mehrere Entscheidungen, welche schlussendlich zu diesem Game Over führten. Nach einem Game Over erfolgt bei dem ursprünglichen Algorithmus ein Zurücksetzen der Umgebung auf einen Anfangszustand. Im Dialog Management lässt sich eine Fehlinterpretation, also ein Game Over, ausschließlich auf die letzte durchgeführte Entscheidung zurückführen. Es besteht kein direkter Bezug zu zuvor getroffenen Entscheidungen. Jede Entscheidung steht für sich selbst. Aufgrund dieser Gegebenheit erfolgt kein Zurücksetzen des Environments, also kein Neubeginn des Dialogs. Bedingt durch die Gegebenheit der unabhängigen Entscheidungen wird auch auf eine zukünftige Belohnung über den sogenannten Discount Faktor γ verzichtet, das heißt hier gilt $\gamma = 0$.

Exploration und Exploitation
Einen weiteren Hyperparameter stellt ϵ da. Dieser Parameter gibt die Wahrscheinlichkeit der zufälligen Wahl einer Aktion, und somit der Exploration, an. Mit welcher Wahrscheinlichkeit wird eine zufällige Aktion oder eine Aktion anhand der Vorhersage des neuronalen Netzes gewählt? Bei der Wahl dieses Parameters bestehen zwei grundlegende Möglichkeiten. Einerseits kann ϵ ein fester Wert zugewiesen werden oder der Wert wird laufend durch eine Funktion angepasst, welche es ermöglicht zu Beginn des Trainings ein hohes Maß an Exploration zu verfolgen und im späteren Verlauf eine Exploitation, Ausbeutung des Wissens, durchzuführen. Die hierbei vielversprechendere Variante liegt in der adaptiven Ermittlung anhand einer Funktion.

$$\epsilon_t = 0.1 \times \frac{1}{\log(t+2)} \tag{6.1}$$

Hierfür wurde der Annealing Epsilon Algorithmus gewählt [61, S.61]. t beziffert den Index für die Episode. So ist beispielsweise der ϵ Wert für die 10 Episode $\epsilon = 0,092$. In der Episode 100 beträgt ϵ noch $0,0497$. Bei späteren Episoden nimmt somit die Wahrscheinlichkeit der Exploration, der zufälligen Wahl einer Aktion, ab.

Reward
Bei Dialog Management kann eine ermittelte Aktion entweder als korrekt oder als inkorrekt gewertet werden. Somit ergibt sich ein Reward von dem Wert 1 für eine korrekt ermittelte Aktion oder 0 für eine falsch ermittelte Aktion.

Episoden
Bei dem Erlernen von Atari-Spielen handelt es sich bei einer Episode um einen Durchlauf des Spiels bis zum Erreichen vom Ende des Spiels durch erreichen des Ziels oder Sterben des Charakters. Auch bei dieser Umsetzung endet eine Episode bei inkorrekter Wahl einer Aktion oder bei korrektem Durchlauf aller Dialoge.
Die Zahl der durchzuführenden Episoden wird anhand der Anzahl der Interaktionen der Trainingsdialoge ermittelt. Die Zahl der Episoden entspricht $^1/_5$ der Interaktionen. Die dynamische Ermittlung hat das Ziel ein Overfitting oder ein nicht ausreichendes Training zu verhindern.

7 Evaluation

In den vorherigen Abschnitten dieser Arbeit wurden mit der Vertauschung, dem Einfügen von Chitchat und der Generierung neuer Dialoge, drei Algorithmen zur Augmentierung beschrieben. Des Weiteren wurde das Supervised Learning sowie die Implementierung des Reinforcement Learning dargestellt. In diesem Kapitel werden die Algorithmen der Augmentierung unter Verwendung der Lernvarianten evaluiert. Dies erfolgt, um Aussagen bezüglich möglicher positiver oder negativer Aspekte der Augmentierung auf das Dialog Management zu geben.

Trainingsdaten
Um die Evaluierung durchzuführen, werden die zwei Korpora unabhängig voneinander als Trainingssets genutzt. Der MultiWOZ 2.1 Korpus wird aufgrund der Datenmenge von insgesamt 9.061 Dialogen und den damit verbundenen Hardwareanforderungen lediglich in einer Ausprägung mit geringerer Größe für das Training genutzt. Der kleinere Korpus beinhaltet 925 Dialoge, wobei 740 als Training- und 185 als Testdialoge dienen. Die Dialoge wurden hierfür automatisiert zufällig aus dem MultiWOZ 2.1 Korpus ausgewählt und in Trainings- und Testdialoge unterteilt.
Der zweite Korpus, welcher generierte Daten aus zwei Domänen beinhaltet (siehe Kapitel 4.2 auf Seite 19) wird ohne weitere Anpassungen genutzt. Somit verfügt dieser über 147 Trainings- und 30 Testdialoge.
Um Aussagen über Effekte der Augmentierungen bei unterschiedlich großer Menge an Trainingsdialogen treffen zu können, wird das Training auch mit verringerter Anzahl an Trainingsdialogen ausgeführt werden. Hierfür werden Dialoge automatisiert zufällig aus dem jeweiligen Korpus ausgewählt. Wird ein solches Experiment analysiert, so wird die Menge der Trainingsdaten in Prozent zur gesamten Anzahl an Trainingsdialogen angegeben. Ein direkter Vergleich zwischen einzelnen Tests erfolgt immer auf identischen Trainingsdaten. Somit sind Abweichungen aufgrund der gewählten Daten ausgeschlossen. Die Testdialoge bleiben jeweils unverändert.

Kennzahlen
Um eine Vergleichbarkeit zwischen den Ergebnissen der einzelnen Experimente zu gewährleisten, werden zwei Kennzahlen genutzt werden. Es handelt sich dabei um die Genauigkeit (engl. accuracy) und das F-Maß F_1. Diese werden anhand der im Test getätigten Vorhersagen im Vergleich zu den erwarteten Werten berechnet. Die Berechnung erfolgt durch das genutzte Framework. In diesem Abschnitt sind die zugrunde liegenden Formeln beschrieben.
Zur Berechnung der Genauigkeit erfolgt ein Vergleich zwischen allen Vorhersagen y_i und den erwarteten Werten $\hat{y}_i$. Eine Division durch die Anzahl n, die Menge der Elemente ergibt die Genauigkeit.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n-1} 1(\hat{y}_i = y_i) \tag{7.1}$$

Bei dem F-Maß handelt es sich um das »harmonische[s] Mittel aus positivem Vorhersagewert und Richtig-Positiv-Rate« [53, S. 91]. Die Berechnung erfolgt anhand von drei »Basis der Häufigkeit der vier Fehler TP« (true positive)», TN« (true negative)», FP« (false positive) »und FN« (false

negative) [53, S. 91].

$$\begin{aligned} F_1 &= 2 \times \frac{TP}{R+P} \\ R &= TP + FN \\ P &= TP + FP \end{aligned} \tag{7.2}$$

Die Ermittlung der Kennzahlen erfolgt auf Basis der einzelnen Interaktionen, und falls nicht anders angegeben, wird das Ergebnis von einem Experiment für die Berechnung genutzt. Um Signifikanztests durchzuführen und 95%-Quantile zu ermitteln, werden Experimentreihen mit je 5 Experimenten durchgeführt. Hierfür sind zu Beginn jeweils 5 Trainingssets pro Datenmenge automatisiert zufällig erstellt worden. Mögliche Signifikanz wird anhand des Student t-Tests ermittelt.

Implementierung
Die Implementierung erfolgt im Rasa Framework in Version 1.8.2 mit Python 3.7.6. Die Ermittlung der genannten Kennzahlen erfolgt durch das Framework. Die Implementierung und die genutzten Trainingsdaten sind öffentlich einzusehen[1].

Zunächst ist die Evaluierung der Augmentierung unter Verwendung von Supervised Learning als Lernvariante dargestellt. Daraufhin die Evaluation dieser Algorithmen unter Nutzung des Reinforcement Learnings.

7.1 Supervised Learning

Das Training ohne Anpassungen ist mit einer Historie von 5 bei 100 Epochen konfiguriert. Diese Standardkonfiguration wird in den folgenden Diagrammen als *no augmentation* in blauer Farbe dargestellt. Die Evaluierung der Algorithmen ist in identischer Reihenfolge wie diese im Kapitel 5 auf Seite 20 eingeführt worden. So erfolgt zunächst die Evaluierung der Vertauschung, gefolgt vom Einfügen des Chitchats und abschließend die Darstellung der Ergebnisse der Generierung neuer Dialoge anhand des Graphen *G*.

7.1.1 Vertauschung

Die Analyse der Augmentierung anhand Vertauschung erfolgt unter Verwendung des MultiWOZ-Korpus und des generierten Korpus. Die Augmentierung durch Vertauschung wurde beschrieben im Abschnitt 5.1 auf Seite 22.

Korpus: MultiWOZ

Es erfolgen Trainings mit 5% (37 Dialoge), 25% (186 Dialoge) und 100% (740 Dialoge) der vorhandenen Trainingsdaten. Die Durchführung von Experimenten geschieht einerseits ohne Augmentierung (Abb. 7.1 auf der nächsten Seite blau) und andererseits unter Verwendung der Vertauschung (Abb. 7.1 auf der nächsten Seite grün). Als Konfiguration für die Vertauschung wird eine mehrfache Vertauschung vorgenommen, wobei jeder einzelne Dialog in bis zu 3 Teildialoge unterteilt wird. Dies wird durch die Kennzahl $k = 3$ dargestellt. Ist eine Unterteilung in drei Teildialoge nicht möglich, da weniger Blöcke in diesem Dialog vorhanden sind, wird die maximale Zahl der Teildialoge genutzt. Es gilt somit:

$$k = \min\{3, |B_i|\} \tag{7.3}$$

[1] `https://github.com/karnehm/Chatbot_data_augmentation_and_learningtypes`

Jeder Dialog wird zwei Mal kopiert und mit diesem Verfahren manipuliert. Im Weiteren wird dies durch die Kennzahl c angegeben werden. Dadurch wird die Anzahl der Trainingsdialoge im Vergleich zu der ursprünglichen Menge verdreifacht. Die Trainingsdialoge beinhalten somit die originalen Dialoge und zwei Kopien, welche mit der Vertauschung manipuliert wurden.

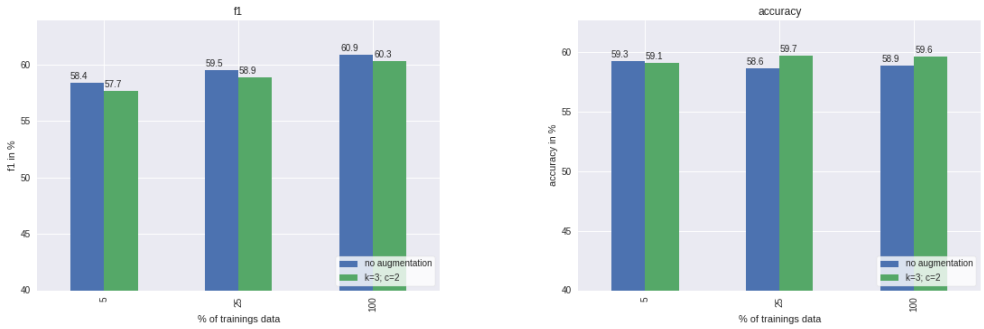


Abbildung 7.1: MultiWOZ 2.1 - ohne Augmentierung und Vertauschung mit $c = 2$ und $k = 3$ im Vergleich. Bei zunehmender Anzahl an Trainingsdialogen ist keine Verbesserung der Genauigkeit ersichtlich. Augmentierung macht hierbei keinen Unterschied.

Den in 7.1 dargestellten Diagrammen ist zu entnehmen, dass mit mehr Trainingsdaten die Genauigkeit und das F_1 Maß nicht wie erwartet in einem sehr klaren Teil zunimmt. Hierbei ist es unabhängig, ob die Augmentierung vorgenommen wurde oder nicht. So beträgt die Accuracy mit 37 Dialogen ohne Augmentierung 59,3%. Mit 740 Dialogen konnte ein Wert von 58,9% festgestellt werden. Für das F_1 Maß ist eine geringfügige Verbesserung bei mehr Dialogen festzustellen. Jedoch liegt diese mit 2,5% auch in einem unerwartet geringen Bereich. Diese Ergebnisse legen eine genauere Analyse nahe. Diese erfolgt anhand der Konfusionsmatrix (siehe Abb. 7.2 auf der nächsten Seite), welche durch das Rasa Framework erstellt wird. Die Matrix beinhaltet die Menge A der möglichen Aktionen des Assistenten.

Die Abbildung 7.2 auf der nächsten Seite stellt die Konfusionsmatrix für 100% der verfügbaren Daten ohne Augmentierung da. Die weiteren Experimente weisen eine vergleichbare Konfusionsmatrix auf. Insgesamt wurden 1.584 der 3.655 Interaktionen fehlerlos ermittelt. Auffallend ist hierbei jedoch, dass 780 der korrekt vorhergesagten Interaktionen der Aktion `action_listen` zuzurechnen sind. Die restlichen 804 korrekt ermittelten Interaktionen verteilen sich auf 34 unterschiedliche Aktionen. Für 8 Aktionen konnte keine richtige Vorhersage festgestellt werden. Bei `action_listen`, in der Abbildung durch einen roten Rahmen hervorgehoben, handelt es sich um eine Standard-Aktion von dem Rasa Framework. Die Dokumentation beschreibt diese Aktion wie folgt: »Stop predicting more actions and wait for user input« [48]. Diese Aktion könnte somit als Ende eines Blocks bezeichnet werden. Bereits in [72] wurde diagnostiziert, dass dieser Korpus aufgrund einer möglichen Historienunabhängigkeit in seinen Dialogen für die Evaluation von Dialog Management Systemen nur schlecht geeignet ist. Es lässt sich hier feststellen, dass die korrekte Ermittlung vom Ende eines Blocks deutlich genauer erfolgt als die Vorhersage der restlichen Aktionen. Auch zeigt sich, dass Aktionen häufig miteinander vertauscht werden. So wird fälschlicherweise die Aktion `bye_general` 12 mal mit der Aktion `welcome_general` vertauscht. Daneben wird die Aktion `welcome_general` 32 mal vorhergesagt, anstelle der korrekten Aktion `bye_general`. Die Vertauschung zweier Aktionen miteinander zeigt sich obendrein bei den Aktionen `inform_booking_restaurant` und `inform_restaurant`, `request_train` und `inform_train` sowie `action_book_train` und `inform_train`. Diese Beobachtung in den Testergebnissen spricht auch für die bereits in [72] festgestellte historische Unabhängigkeit in den

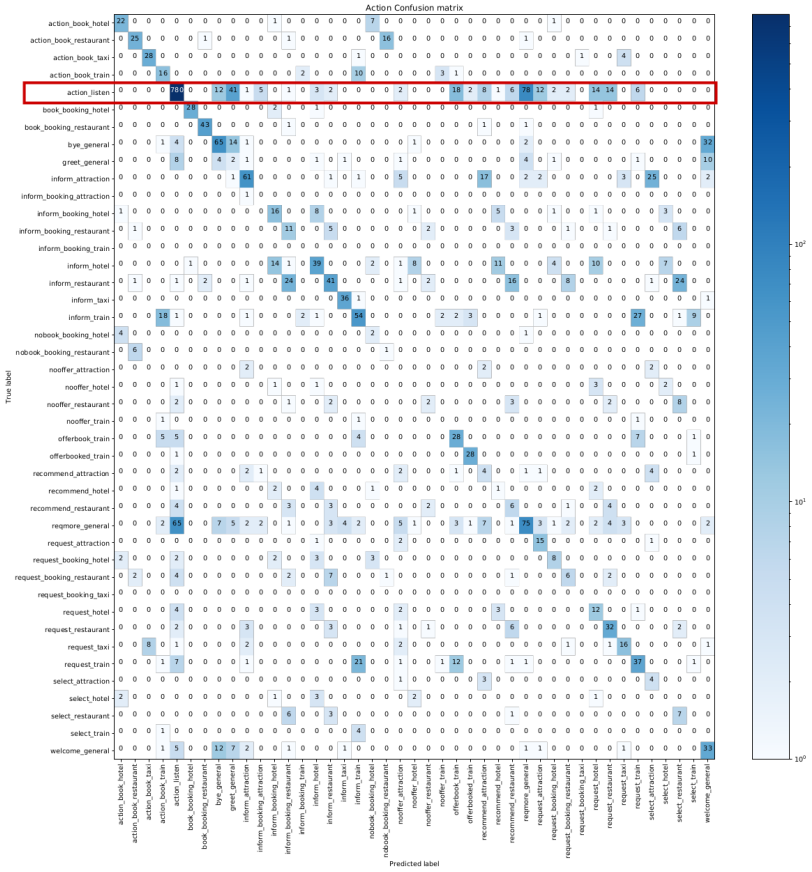


Abbildung 7.2: Konfusionsmatrix des MultiWOZ Korpus ohne Augmentierung bei 100% der Trainingsdaten. Matrizen anderer Experimente weisen vergleichbare Ergebnisse aus.

Dialogen. Die oben dargestellten Paare sind jeweils der identischen Domäne zuzuordnen. Es kann nicht ausgeschlossen werden, dass ein menschlicher Benutzer die ermittelten Aktionen des Dialog Managements als korrekt definieren würde und somit eine höhere Genauigkeit festgestellt werden könnte.

Ebenso die geringe Verbesserung bei zunehmender Menge an Trainingsdaten spricht für die historische Unabhängigkeit und eine hohe Varianz in den Dialogen. Diese Varianz lässt sich unter anderem auf die Art und Weise der Erstellung zurückführen. Die Dialoge wurden von mehreren Menschen als Wizards erstellt, welche keinen festen Regeln in der Kommunikation folgen mussten. Es wird somit verständlich, weshalb diese Methode der Erstellung auch der Mensch-Mensch-Kommunikation zugeordnet werden kann, wie es beispielsweise in [10] getan wird und weshalb dieser Art der Korpora kritisch zu betrachten ist.

Die Aussage von [72], dass sich dieser Korpus aufgrund seiner Struktur nicht oder nur sehr begrenzt für die Evaluierung von Dialog Management Algorithmen eignet, kann bei Betrachtung dieser Ergebnisse bestätigt werden. Die Ergebnisse der Konfusionsmatrix lassen eine Bestätigung dieser Beurteilung zu. Auch wenn Verbesserungen durch eine Augmentierung festgestellt werden könnten, ist die Aussagekraft solcher Ergebnisse stark zu bezweifeln. Aufgrund dessen wird der MultiWOZ 2.1 Korpus für weitere Evaluierungen nicht genutzt.

Korpus: Generierte Daten

Im Weiteren erfolgt die Analyse des Verfahrens anhand des zweiten Korpus. Trainings mit 5% (8 Dialoge), 25% (37 Dialoge) und 100% (147 Dialoge) der verfügbaren Daten werden durchgeführt. Die Augmentierung erfolgt in vier Konfigurationen der Vertauschung. Die Augmentierungen werden mit den Ergebnissen ohne Augmentierung verglichen. Die Anzahl der Teildialoge k wird in der ersten Analyse von 2 bis 5 variiert. Es erfolgt jeweils eine konstante Vervielfachung von $c = 2$ in diesen Experimenten.

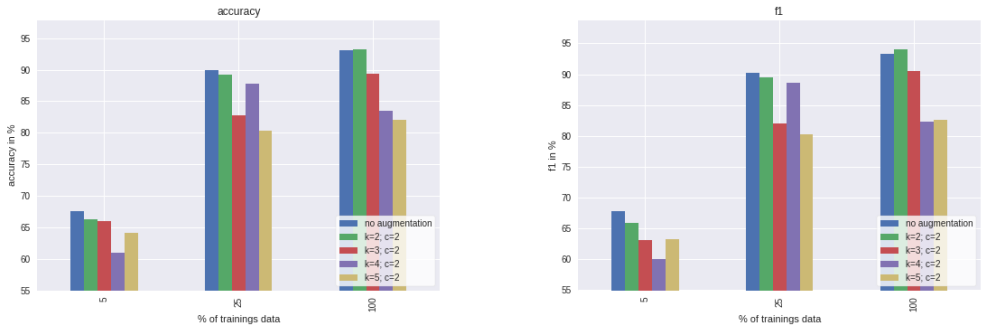


Abbildung 7.3: Vertauschung mit $c = 2$ und $k = [2; 5]; k \in \mathbb{N}$; Bei 100% ist mit mehr Teildialogen eine Abnahme der Genauigkeit zu sehen; Bei geringerer Anzahl Dialoge ist dies nicht so deutlich sichtbar.

In den Resultaten des MultiWOZ 2.1 Korpus konnte keine klare Verbesserung bei zunehmender Datenmenge festgestellt werden. Die in der Abbildung 7.3 dargestellten Ergebnisse zeigen von 5% der Daten hin zu 25% der Daten sichtbar höhere Werte für das F_1 Maß und die Accuracy. Eine weitere Verbesserung ist auch zwischen 25% und 100% festzustellen. Die Graphiken weisen darauf hin, dass durch die Augmentierung lediglich in einem Experiment eine Verbesserung gegenüber dem Vergleichswert erzielt werden konnte. So weißt das F_1 Maß für $k = 2 \wedge c = 2$ bei vollständigen Daten ein leicht besseres Ergebnis aus. Es handelt sich um eine Steigerung von 0,7% und liegt im Rahmen der zu erwartenden Schwankungen. Vor allem zeigen die Ergebnisse, dass dieses Verfahren mit mehr Teildialogen bei einer größeren Anzahl an Trainingsdialogen zu geringeren Genauigkeiten im Vergleich zum Vergleichswert führt. So weißt das F_1 Maß bei $k = 4 \wedge c = 2$ und 100% der Trainingsdialoge ein Wert von 82,3% auf. Dies ist 11,1% unter der Marke ohne Augmentierung. Es kann davon ausgegangen werden, dass ein größerer Wert von k zu verschlechterten Ergebnissen in dieser Lernvariante führt. Diese Tendenz zeigt sich bei geringer Zahl an Trainingsdialogen nicht so deutlich wie bei einer höheren Anzahl.

Im Folgenden ist ein möglicher Zusammenhang zwischen der Vervielfachung der Dialoge und der Genauigkeit des Dialog Management festzustellen. Hierfür werden Trainings mit einer konstanten Anzahl von 2 Teildialogen und Vervielfachungen von 2 bis 5 durchgeführt. Wie der Abbildung 7.4 auf der nächsten Seite entnommen werden kann, lässt sich bei geändertem Vervielfachungsfaktor c keine klare Änderung bei der Accuracy und dem F_1 Maß bemerken. Für beide Werte sind nur sehr geringe Schwankungen sichtbar.

Um eine abschließende Aussage über das Verfahren zu tätigen, werden weitere Experimente mit $k = 2 \wedge c = 2$ zur Ermittlung der Schwankungen zwischen 5% und 25% der Daten durchgeführt. Das Experiment wird hierfür jeweils fünfmal mit zuvor automatisiert zufällig gewählten Dialogen wiederholt. Die Ergebnisse dieser Experimente werden daraufhin mit den Kennzahlen ohne Augmentierung bei identischen Daten verglichen. Die dabei ergebenden Graphen der Accuracy und des F_1 Maßes sind in der Abbildung 7.5 auf der nächsten Seite dargestellt. Es lässt sich feststel-

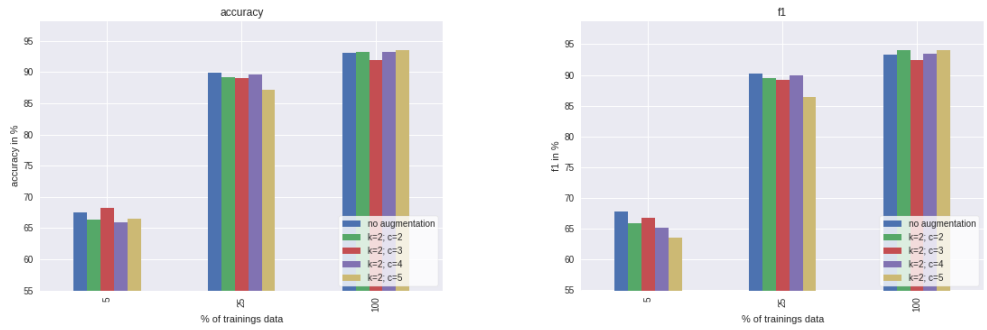


Abbildung 7.4: Vertauschung mit $c = [2; 5], c \in \mathbb{N}$ und $k = 2$; Mehr Kopien, ein höherer Wert von c, führt nicht zu ersichtlichen Änderungen bei den Kennzahlen accuracy und F_1-Wert Kennzahlen.

len, dass der durchschnittliche Wert ohne Augmentierung stetig höher liegt, jedoch befinden sich die Ergebnisse innerhalb des 95%-Quantils der Ergebnisse ohne Augmentierung. Es ist somit davon auszugehen, dass die Anwendung der Vertauschung zu einer durchschnittlichen Verschlechterung der beiden Kennzahlen führt, welche nicht als signifikant zu deuten ist.

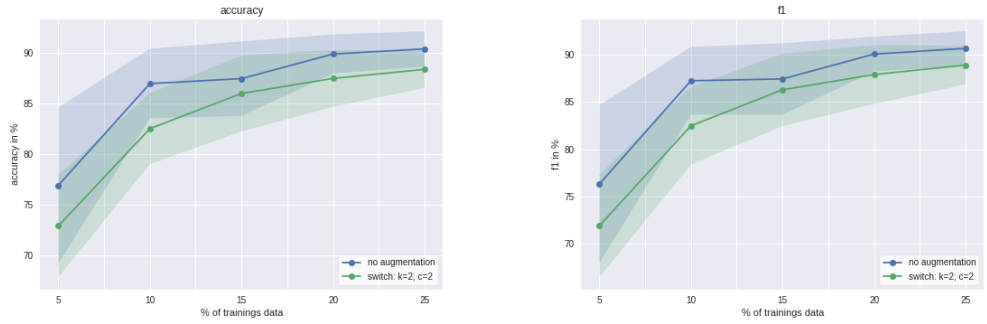


Abbildung 7.5: Konfigenzintervalle von 5% bis 25% der Daten ohne Augmentierung und mit Vertauschung $k = 2; c = 2$; Ergebnisse ohne Augmentierung sind durchschnittlich besser. Mit Vertauschung ist eine geringere Schwankung festzustellen.

7.1.2 Chitchat einfügen

In diesem Abschnitt der Evaluation erfolgt die Analyse des Chitchat Verfahrens mit Supervised Learning als Lernvariante. Die Überlegung in diesem Verfahren besteht darin, ein verbessertes Ergebnis zu erhalten, indem in einen bestehenden Dialog zwischen zwei zufällig gewählten Blöcken ein Chitchat, eine unerwartete unbedeutende Frage, eingefügt wird und somit das Dialog Management die Aufmerksamkeit auf relevante Interaktionen verbessert. In den Experimenten werden zwei Ausprägungen des Chitchat Korpus betrachtet. Einerseits wird ein Korpus mit der Mächtigkeit von 1 genutzt. Es gilt somit $|D_c| = 1$. Dies hat zur Folge, dass in jedem Fall das identische Chitchat eingefügt wird. Diese Variante wird als `single` bezeichnet. Im Gegensatz dazu gilt für den im Weiteren als `multi` bezeichneten Chitchat Korpus $|D_c| = 100$.

Die erste Reihe an Experimenten fügt einen oder zwei Chitchats pro Dialog ein. Dabei wird unterschieden zwischen den beiden Chitchat Korpora `single` und `multi`. Dadurch ergeben sich vier Konfigurationen. Die daraus resultierenden Ergebnisse sind der Abbildung 7.6 entnehmbar.

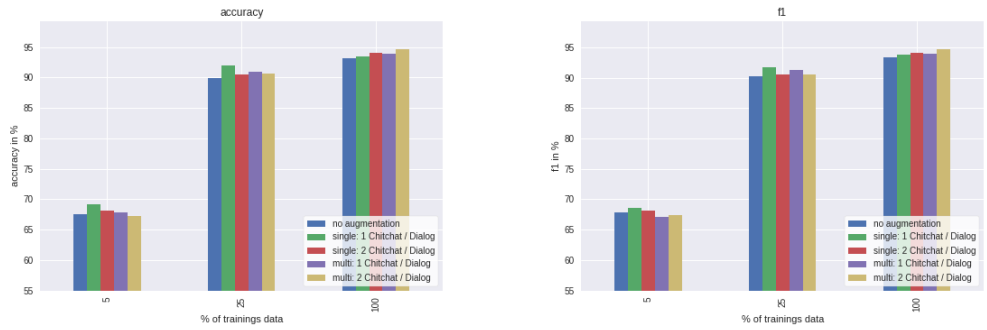


Abbildung 7.6: Einfügen von 1 und 2 Chitchats pro Dialog in zwei Ausprägungen des Chitchat Korpus. *single* beinhaltet einen eindeutigen Chitchat und *multi* beinhaltet 100 verschiedene. Deutlichste Verbesserung gegenüber Vergleichswert ist bei *single* mit einem Chitchat bei 25% zu bemerken. Dieser ist jedoch nicht signifikant.

Dabei auffallend sind die Ergebnisse des `single` Korpus bei 25% der Daten mit einem Chitchat pro Dialog (grün). Für die Feststellung einer möglichen Signifikanz ist das Experiment fünfmal zu wiederholen. Die Ergebnisse zeigen eine Verbesserung für beide Kennzahlen. Gleichwohl ist hierbei nicht von einer Signifikanz zu sprechen, wie der unterhalb abgebildeten Tabelle zu entnehmen ist.

Tabelle 7.1: Ergebnisse des Student-t-Test für Accuracy und F_1 Maß bei einfügen von einem Chitchat pro Dialog mit $|D_c| = 1$ gegenüber Vergleichswert ohne Augmentierung.

Anzahl Dialoge in %	Kennzahl	*single: 1 Chitchat/Dialog*	*ohne Augmentierung*	*p-Wert*
25%	Accuracy	91,6% ± 1,7%	90,4% ± 1,7%	0,222
	F_1 Maß	91,8% ± 1,8%	90,6% ± 1,8%	0,229

Auch die weiteren Konfigurationen weisen in der Abbildung 7.6 durchweg bessere Ergebnisse im Vergleich zu deren ohne Augmentierung (blau) auf. Auffallend sind vor allem Verbesserungen bei vollem Zugriff auf die Trainingsdialoge.

Eine nächste Reihe an Experimenten augmentiert Trainingsdialoge wie folgend beschrieben.

1. Unveränderte Dialoge zu Trainingsset hinzufügen
2. n Kopien der Dialoge durchführen
3. m Chitchats in kopierte Dialoge einfügen
4. Augmentierte Dialoge zu Trainingsset hinzufügen

Dieser Vorgang wird für $m = [1, 2]$ und $n = [1, 2]$ in beiden Chitchat Korpora durchgeführt. Damit ergeben sich 8 Ausprägungen, deren Ergebnisse in der Abbildung 7.7 auf der nächsten Seite dargestellt sind.

Es zeigt sich für den Chitchat Korpus `single` eine Verbesserung beider Kennzahlen bei $n = 2$, was zwei Kopien entspricht. So ergeben sich für diesen Korpus die besten Ergebnisse bei 2 Kopien der Dialoge (lila und gelb oben). Bei 25% der Daten, zwei Kopien und einem Chitchat pro Dialog, weißt sich eine 2,4% höhere Accuracy und ein 2,6% besseres F_1 Maß im Vergleich zu der blau dargestellten Kennlinie aus.

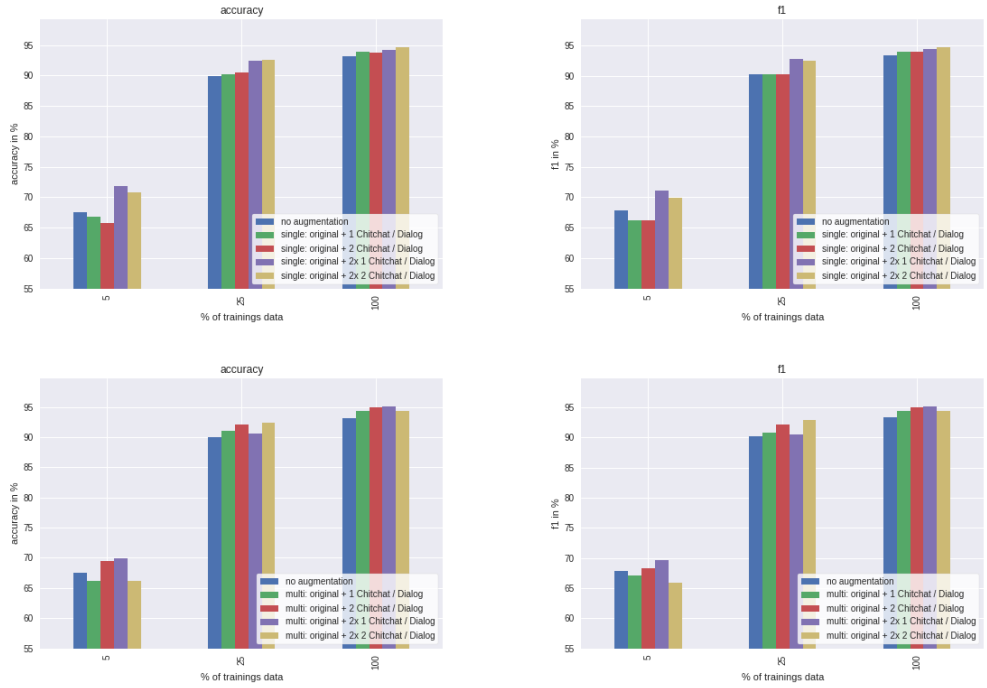


Abbildung 7.7: Original Dialoge mit 1-2 Kopien und 1-2 Dialogen pro Chitchat in den zwei Ausprägungen des Chitchat Korpus mit einem eindeutigen Chitchat (oben) und des Chitchat Korpus mit 100 verschiedenen (unten). Eine Verbesserung ist in allen Fällen festzustellen. Bei 100% der Dialoge, *multi: original + 2x 1 Chitchat / Dialog* lässt sich für die Accuracy $p = 0,069$ bei 5 Experimenten feststellen.

Für den Korpus mit $|D_c| = 100$ lässt sich anhand der Ergebnisse kein vergleichbares Muster erkennen. Die deutlichste Verbesserung ist mit 100% der Dialoge, zweifacher Kopie und einem Chitchat pro Dialog (lila unten) bei diesem Korpus zu verzeichnen. Ein Signifikanztest mit fünf Experimenten führt mit dem Student-t Test für die Accuracy zu $p = 0,069$. Für das F_1 Maß ergibt sich ein Wert von $p = 0,151$. Es lässt sich somit für $p > 0,1$ bei Accuracy eine Signifikanz feststellen. Für das F_1 Maß lässt sich ein leicht besserer, wenn auch signifikanter Effekt bemerken.

Bei beiden Varianten der Chitchat-Korpora lässt sich feststellen, dass die größten Verbesserungen bei geringerer Anzahl an Trainingsdialogen festzustellen ist. Die Augmentierung führt bei einer hohen Anzahl an Dialogen zu einer geringeren Auswirkung.

Abschließend werden die 95%-Quantile für die Bereiche zwischen 5% bis 25% mit jeweils 5%-Punkten Steigerung mit je 5 Experimenten für die Konfiguration *single: original + 2x 1 Chitchat / Dialog* mit Ergebnissen ohne Augmentierung vergleichend dargestellt. Des Weiteren werden diese Experimente bei identischer Konfiguration zusätzlich mit Rückführung vorgenommen. Die Rückführung wurde im Abschnitt 5.2.2 auf Seite 25 beschrieben.
Der Abbildung 7.8 auf der nächsten Seite kann entnommen werden, dass unter Verwendung der Augmentierungen die Schwankung der Genauigkeit und des F_1 Maßes in beiden Ausprägungen (grün und rot) im Vergleich zu der Schwankung ohne Augmentierung (blau) abnimmt. Deutlich erkennbar ist die Verbesserung beider Kennzahlen bei Einfügung der Rückführung. Es kann anhand dieser Ergebnisse davon ausgegangen werden, dass die Rückführung positive Effekte auf die Genauigkeit des Dialog Management hat. Dies zeigt sich auch bei Betrachtung der p-Werte. Diese sind für die Datenmengen 15%, 20% und 25% für beide Kennzahlen folgend dargestellt.

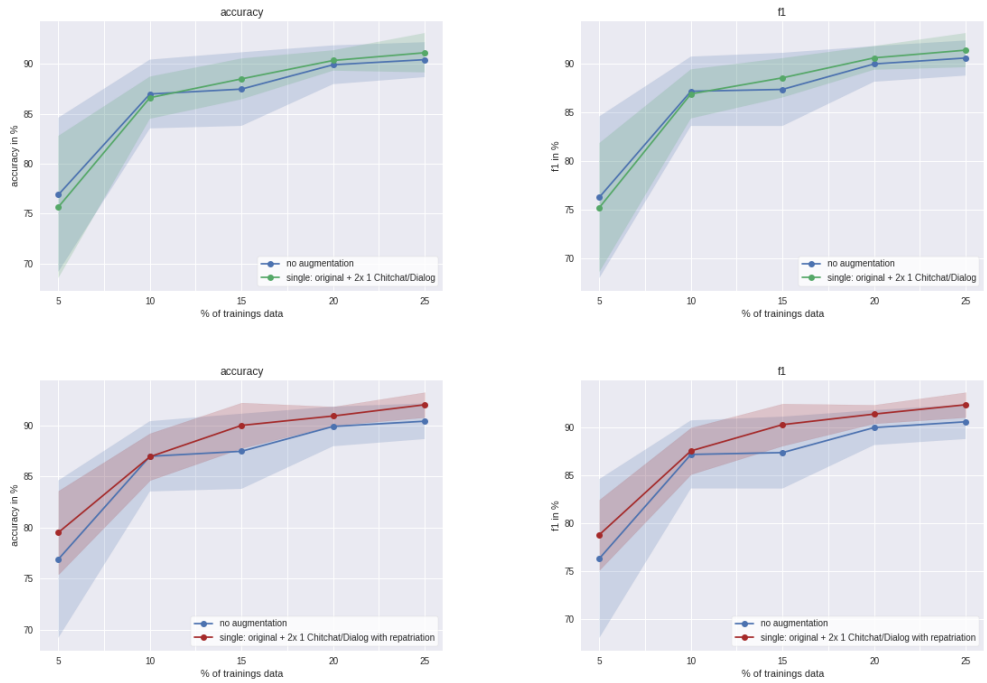


Abbildung 7.8: Konfidenzintervalle für `single` Chitchat Korpus bei 5%-25% der Daten. Konfiguration: *original + 2x 1 Chitchat / Dialog* ohne Rückführung (oben) und mit Rückführung (unten). Rückführung zeigt positiven Effekt auf Genauigkeit des Dialog Management bei Tests. Ab 15% der Daten ist durchgehend eine durchschnittlich höhere Genauigkeit erkennbar.

In den Zahlen zeigt sich nochmals, dass durch das hinzufügen von Chitchat eine verringerte Schwankung in den Ergebnissen festzustellen ist. Des Weiteren konnte mit 25% der vorhandenen Trainingsdialoge ein p-Wert von 0,068 für die Accuracy und 0,057 für das F_1 Maß erzieht werden. Die Verbesserung beträgt 1,6% und 1,8%.

Tabelle 7.2: Vergleich der Kennzahlen mit Einfügen von Chitchat mit Rückführung und ohne Augmentierung bei 15% bis 25% der Daten. Signifikanz für p<0,1 bei 25% der Dialoge.

Anzahl Dialoge in %	Kennzahl	*single: original + 2x 1 Chitchat pro Dialog mit Rückführung*	*ohne Augmentierung*	*p-Wert*
15%	Accuracy	90,0% ± 2,2%	87,5% ± 3,7%	0,140
	F_1 Maß	90,3% ± 2,2%	87,4% ± 3,8%	0,099
20%	Accuracy	90,9% ± 1,0%	89,9% ± 1,9%	0,222
	F_1 Maß	91,4% ± 1,0%	90,0% ± 1,8%	0,094
25%	Accuracy	**92,0% ± 1,2%**	**90,4% ± 1,7%**	**0,068**
	F_1 Maß	**92,4% ± 1,3%**	**90,6% ± 1,8%**	**0,057**

Ablation

Nachdem Verbesserungen in beiden Kennzahlen festgestellt werden konnten, ist in Erfahrung zu bringen, ob diese auf die Augmentierung oder auf die reine Vervielfachung der Trainingsdaten zurückführen ist. Um dies festzustellen, ist ein weiteres Experiment durchzuführen, bei dem keine

Augmentierung erfolgt und das Training mit dreifacher Anzahl an Dialogen statt findet. Die Erhöhung der Anzahl erfolgt durch Kopieren der Dialoge. Die Erhöhung der Datenmenge um den Faktor 3 ergibt sich, da durch die durchgeführte Augmentierung die ursprünglichen Daten und zwei augmentierte Kopien in den Trainingsdaten enthalten. Die Abbildung 7.9 zeigt die Kennlinie ohne Augmentierung mit 3 Kopien im Vergleich zu Augmentierung mit $n = 2 \wedge m = 1$ und Rückführung. In der Abbildung nicht dargestellt ist ein Training mit drei Kopien der Dialoge ohne weitere Anpassungen.

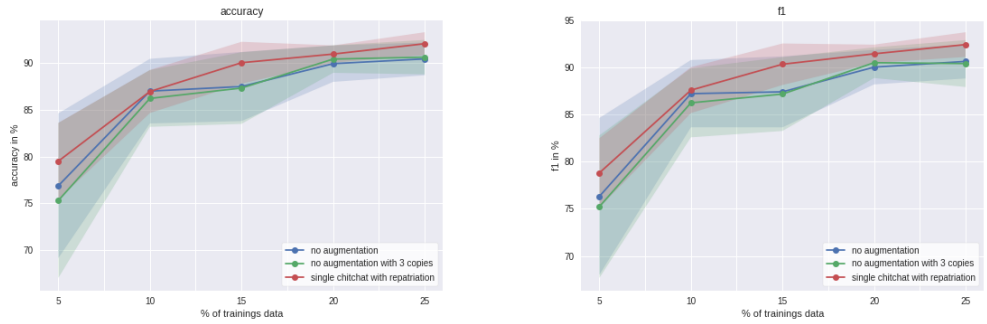


Abbildung 7.9: Die Augmentierung (rot) führt zu geringeren Schwankungen und zu geringfügig besseren durchschnittlichen Kennzahlen. Ein t-Test zeigt keinen signifikanten Unterschied für die Verbesserung bei erhöhter Anzahl an Epochen im Vergleich zu Kennlinie.

Es zeigt sich zwischen der Kennlinie (blau) und der dreifachen Kopie ohne Augmentierung (grün) keine deutliche Unterscheidung in den Linien [2]. Erfolgt ein Vergleich zwischen den Kennzahlen mit 3 Kopien und der Kennlinie ist kein Unterschied zwischen beiden Experimentreihen festzustellen. Dies lässt darauf rückschließen, dass die Verbesserungen in den Kennzahlen auf die Augmentierung und nicht auf die erhöhte Anzahl an Trainingsdaten zurückzuschließen ist. Somit führt die Augmentierung mittels Einfügen von Chitchat mit Rückführung zu positiven Effekten auf das Dialog Management.

7.1.3 Generierung neuer Dialoge

Als dritte und letzte Variante der Augmentierung erfolgt die Evaluation der Generierung neuer Dialoge anhand des Graphen G, wie in Abschnitt 5.3 auf Seite 25 beschrieben, unter Verwendung von Supervised Learning als Lernvariante. Dieses Verfahren ist anhand von drei Parametern konfigurierbar. Bereits in der Einführung wurde der Parameter r beschrieben, welcher die Wahrscheinlichkeit eines zufälligen Wechsels angibt. Des Weiteren werden `mul` und `length` für die Konfiguration eingeführt. Bei `mul` handelt es sich um den Multiplikationsfaktor. Dieser gibt die Anzahl der zu generierenden Dialoge abhängig von der Anzahl der ursprünglich vorhandenen an. Sind 10 Trainingsdialoge vorhanden und gilt `mul=2`, so werden 20 neue Dialoge generiert werden. Bei `length` handelt es sich um die Anzahl an Interaktionen pro zu generierendem Dialog. Die Kombination dieser beiden Parameter ermöglicht eine feine Parametrisierung. So ist durch die Variation des Multiplikationsfaktors `mul` ein optimaler Rahmen festzustellen. Die feinere Parametrisierung ist durch die Länge der Dialoge `length` festzustellen. Für das Training und die damit verbundenen Ergebnisse in den Tests ist es nicht relevant, ob beispielsweise `mul=2` $\wedge$ `length=20` oder `mul=4` $\wedge$ `length=10` gilt. Beide Konfigurationen führen zu deckungsgleichen Ergebnissen, da die Gesamtzahl der Interaktionen übereinstimmend ist und diese die Grundlage des maschinel-

[2] Eine Erhöhung von 100 Epochen auf 300 Epochen, anstatt der Durchführung von Kopien, führt zu vergleichbaren Ergebnissen.

len Lernens bilden.

Generierung anhand des Graphen G

Zunächst erfolgen Experimente mit `length`=30 und unterschiedlichen Werten für `mul`. Für den Multiplikationsfaktor werden die Werte 2, 3 und 4 gewählt um darauf aufbauend die optimale Länge der Dialoge festzustellen.

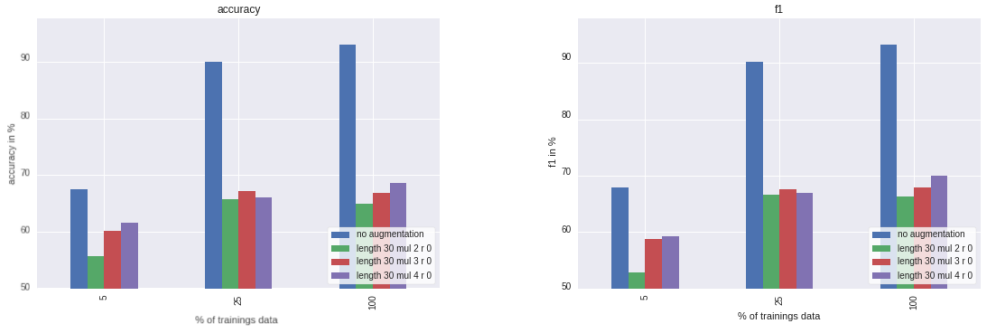


Abbildung 7.10: Generierung neuer Graphen mit konstanter Länge 30 und variablen Multiplikationsfaktor von 2 bis 4. Es zeigt sich eine deutliche Verschlechterung der Genauigkeiten im Vergleich zu Ergebnissen ohne Augmentierung. Auch ist keine Verbesserung der Genauigkeit bei zunehmender Anzahl an Dialogen feststellbar.

Das Diagramm der Abbildung 7.10 zeigt eine sehr deutliche Verschlechterung beider Kennzahlen in Gegenüberstellung zum Vergleichswert ohne Augmentierung. Es kann keine Accuracy von über 69% festgestellt werden. Dies ist unabhängig von der Menge an zurückgegriffenen Trainingsdialogen. Ähnliche Kennzahlen sind auch bei höherem Multiplikationsfaktor und einer größeren Länge als den dargestellten festzustellen. Verbesserungen sind dabei nicht zu beobachten. Auch zeigt sich bei zunehmender Anzahl an Dialogen nur eine geringe Verbesserung in den Kennzahlen. So lässt sich bei einem Multiplikationsfaktor von 4 und der Länge von 30 (lila) für die Accuracy bei 5% der Daten nach 5 Experimenten ein Wert von 62,7% $\pm$ 4,6% feststellen. Hingegen bei 25% der Daten 65,3% $\pm$ 2,9% ermittelt wurden. Es ergibt sich daraus ein p-Wert von p=0,224. Es lässt sich somit keine signifikante Steigerung der Genauigkeit bei zunehmender Menge an Dialogen erkennen. Ein vergleichbares Bild zeichnet sich auch bei den Kennzahlen mit 25% der Daten und 100% der Daten. Der Tabelle 7.3 sind die Kennzahlen und p-Werte der in Abbildung 7.10 lila dargestellten Konfiguration bei 5 Experimenten dargestellt.

Tabelle 7.3: Accuracy, F_1 Maß und p-Werte zwischen den einzelnen Datenmengen für *length 30 mul 4 r 0*. Nur zwischen 5% und 100% der Daten ist Signifikanz für p<0,05 festzustellen.

Anzahl Dialoge in %	Kennzahl	Wert	p-Wert zu 5%	p-Wert zu 25%	p-Wert zu 100%
5%	Accuracy	62,7% $\pm$ 4,6%	-	0,224	0,030
	F_1 Maß	62,0% $\pm$ 4,7%	-	0,098	0,007
25%	Accuracy	65,3% $\pm$ 2,9%	0,224	-	0,124
	F_1 Maß	65,9% $\pm$ 3,3%	0,098	-	0,095
100%	Accuracy	67,7% $\pm$ 2,4%	0,030	0,124	-
	F_1 Maß	68,6% $\pm$ 1,9%	0,007	0,095	-

Durch eine Kombination der originalen Dialoge mit generierten Dialogen lässt sich jedoch ein abweichendes Bild zeichnen. Hierbei zeigt sich bei einer geringen Anzahl an generierten Dialogen eine Verbesserung in den Kennzahlen. Die Abbildung 7.11 zeigt dies. Somit lässt sich eine Verbesserung mit einer Länge von 30 pro Dialog und dem Multiplikationsfaktor 2 feststellen.

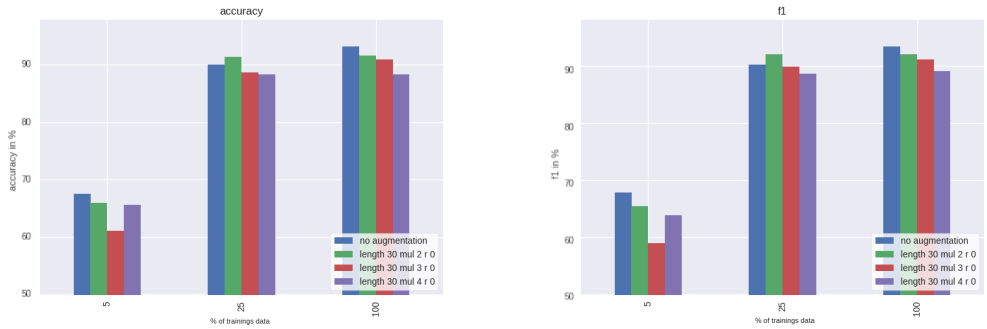


Abbildung 7.11: Original Dialoge kombiniert mit Generierung neuer Graphen mit variablen Multiplikationsfaktor 2 bis 4 und konstanter Länge von 30. Bei geringem Multiplikationsfaktor kann eine Verbesserung bemerkt werden. Bei zunehmender Anzahl der generierten Dialoge sinkt die Genauigkeit.

Im Weiteren erfolgen Trainings mit dieser Variante der Augmentierung mit einem konstanten Wert für `mul`, um eine optimale Anzahl an Interaktionen festzustellen. Die in der Abbildung 7.11 dargestellten Ergebnisse legen nahe, dass mit einer zunehmenden Gesamtzahl an Interaktionen eine Verringerung der Genauigkeit bemerkbar ist. Somit erfolgen die Experimente mit `mul`=2 und einer Länge der Dialoge von 10 bis 40 Interaktionen pro Dialog.

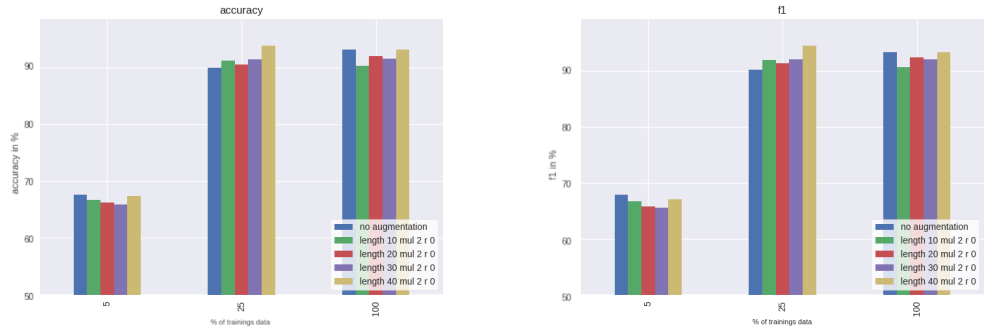


Abbildung 7.12: Generierung neuer Graphen mit konstantem Multiplikationsfaktor 2 und variablen Multiplikationsfaktor von 10 bis 40. Deutlichste Verbesserung ist bei 25% der Daten und einer Länge von 40 erzielt. So können bei 25% der Daten vergleichbare Ergebnisse zu denen ohne Augmentierung festgestellt werden.

Es zeigt sich bei einer Erhöhung der Länge eine Verbesserung beider Kennzahlen, wenn die Genauigkeiten der Augmentierungen untereinander verglichen werden. Sind 25% der Trainingsdialoge verfügbar, liegen die Ergebnisse jeweils über den Vergleichswerten ohne Augmentierung. Bei 100% der Dialoge ist eine generelle Verschlechterung gegenüber der Kennlinie festzustellen. Ein deutlich verbessertes Ergebnis für beide Kennzahlen kann bei den Parametern `length`=40 und `mul`=2 festgestellt werden. So kann bei dieser Konfiguration mit 25% der Trainingsdaten eine Accuracy von 93,9% erreicht werden. Dieser Wert ist mit 0,8-Prozent-Punkten minimal höher als

ohne Augmentierung bei 100% der Trainingsdialoge. Aufgrund dieses Ergebnisses wird das Experiment mit identischer Konfiguration für diese Anzahl an Trainingsdialoge 5-mal wiederholt, um eine mögliche Signifikanz festzustellen. Es ergibt sich dabei eine Accuracy von $91,3\% \pm 1,6\%$. Somit liegt das in der Abbildung 7.12 auf der vorherigen Seite dargestellte Ergebnis deutlich über dem statistischen Mittelwert. Ein möglicher Grund für diese Abweichung liegt in den genutzten Trainingsdialogen. Die durchschnittliche Länge der Dialoge im abgebildeten Experiment liegt bei 28,6 Interaktionen pro Dialog. Die mittlere Länge der Dialoge der weiteren Trainings beträgt hingegen 26,8 Interaktionen. Es lässt sich feststellen, dass nicht nur die Länge der Trainingsdialoge um 8% höher ist, sondern ebenfalls die Anzahl der Interaktionstypen und somit auch die Zahl der Knoten im erzeugten Graphen. Es kann davon ausgegangen werden, dass die erhöhte Menge an Interaktionen zu diesem verbesserten Ergebnis führt. Stichhaltig lässt sich dies nicht Verifizieren. Diese Analyse veranschaulicht, dass schon geringfügige Abweichungen in den Trainingsdialogen in diesem Verfahren zu deutlichen Schwankungen in den Genauigkeiten führen können. Es konnte gezeigt werden, in einzelnen Fällen sind bereits mit 25% der Trainingsdialoge nur gering abweichende Ergebnis zu den Ergebnissen bei 100% der Dialoge ohne Augmentierung möglich. Eine Signifikanz konnte jedoch nicht festgestellt werden.

Generierung anhand des Graphen G mit dem Zufallsfaktor r

Die Erkenntnisse aus den vorherigen Analysen dienen im Weiteren dazu, die Augmentierung anhand des Graphen mit dem Zufallsfaktor r zu evaluieren. Eingeführt wurde diese Variante der Augmentierung im Abschnitt 5.3.2 auf Seite 29. Experimente, ausschließlich basierend auf generierten Dialogen werden nicht ausgeführt. Es wird die zuvor als vielversprechendste Parametrisierung befundene Konfiguration übernommen und eine Variation des Zufallsfaktors vorgenommen. Es erfolgt die Augmentierung mit den Zufallsfaktoren 5, 10 und 15, welche mit den Werten ohne Augmentierung und mit der Augmentierung ohne Zufallsfaktor verglichen werden. Der Zufallsfaktor gibt die Wahrscheinlichkeit an, mit welcher Interaktionen hintereinander gefügt werden, welche in dieser Variante in den Trainingsdaten nicht vorhanden sind.

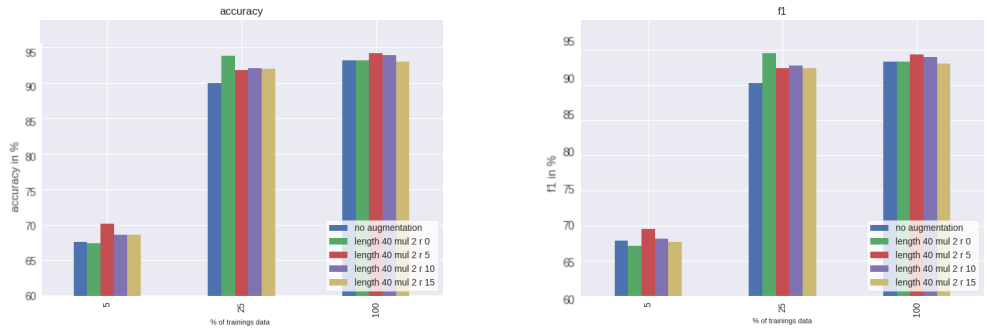


Abbildung 7.13: Generierung der Dialoge anhand des Graphen G und des Zufallsfaktors r. Bei geringer Anzahl an Trainingsdialogen (siehe 25%) verringert der Zufallsfaktor die Genauigkeit der Vorhersage. Bei 100% der Dialoge und $r = 5$ kann aufgrund dieser Daten von einer Verbesserung der Ergebnisse ausgegangen werden, jedoch lässt sich diese nach p-Test nicht bestätigen.

Die in der Abbildung 7.13 abgebildeten Ergebnisse weisen darauf hin, dass bei einer geringeren Anzahl an Trainingsdaten ein Zufallsfaktor $r > 0$ zur Verringerung der Genauigkeit im Vergleich zu $r = 0$ führt. Bereits in vorherigen Experimenten und deren Analyse konnte festgestellt werden,

dass der Durchschnittswert der Accuracy bei 91,3% und das durchschnittliche F_1 Maß 91,4% bei der Konfiguration für $r = 0$ beträgt. Somit liegen auch die Ergebnisse von $r > 0$ innerhalb der erwarteten Schwankung von $r = 0$. Es ist somit nicht davon auszugehen, dass ein Zufallsfaktor $r \in [0; 15)$ zu signifikanten Schwankungen in positive oder negative Richtung führt. Die Abbildung 7.13 auf der vorherigen Seite legt eine Verbesserung für $r = 5$ (rot) bei vollständigem Zugriff auf die Dialoge nahe. Jedoch konnte diese bei Ermittlung der p-Werte anhand von fünf Experimenten nicht bestätigt werden (Accuracy: $p = 0,53$; F_1 Maß: $p = 0,63$).

7.2 Reinforcement Learning

Nach Evaluation der Augmentierungen unter Verwendung von Supervised Learning werden in diesem Abschnitt die Auswirkungen der Augmentierung bei Reinforcement Learning betrachtet. Die Reihenfolge ist identisch zur zuvor durchgeführten Evaluation. Die Bedeutung der einzelnen Parameter ist dem jeweiligen Unterabschnitt des Kapitels Supervised Learning (siehe 7.1 auf Seite 35) zu entnehmen. Die Reihenfolge der einzelnen Algorithmen stimmt mit der Einführung der Augmentierungen überein. Zuvor sind die Kennzahlen des Reinforcement Learnings zur Lernvariante Supervised Learning zu vergleichen.
Es lässt sich, wie in Abbildung 7.14 dargestellt, eine deutlich geringere Genauigkeit bei vielfach längerer Trainingszeit erkennen. Auch zeigt sich in den 95-% Quantilen eine höhere Schwankung in beiden Kennzahlen in Gegenüberstellung zu denen unter Verwendung von Supervised Learning als Lernvariante. Im Rahmen dieses Abschnitts erfolgt ausschließlich ein Vergleich der Kennzahlen der Augmentierung mit den Kennzahlen der Lernvariante Reinforcement Learning.

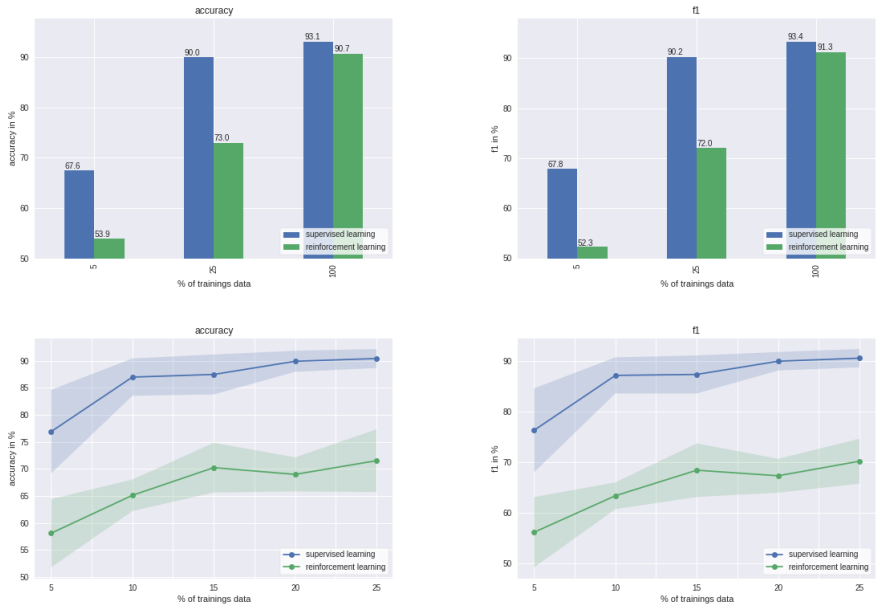


Abbildung 7.14: Vergleich der beiden Lernvarianten ohne Augmentierung. Reinforcement Learning weißt deutlich geringere Genauigkeiten auf.

7.2.1 Vertauschung

Auch in diesem Teil der Evaluation erfolgt zu Beginn die Betrachtung der Ergebnisse der Vertauschung. Zunächst erfolgen Experimente bei konstanter Anzahl an Kopien und variabler Anzahl an Vertauschungen. Der Abbildung 7.15 sind die Ergebnisse der Parameter $c = 2 \wedge k = (2; 5)$ im Vergleich zu den Kennzahlen ohne Augmentierung gegenübergestellt.

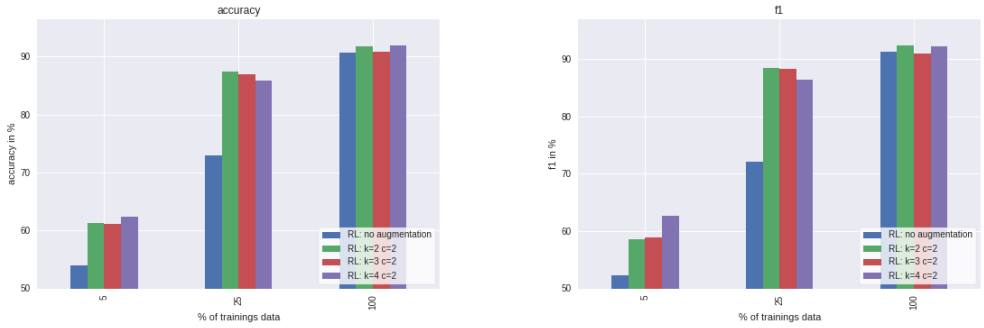


Abbildung 7.15: Vertauschung mit 2 Kopien und 2-4 Vertauschungen. Bei kleiner (5% der Daten) bis mittlerer (25% der Daten) Größe des Trainingskorpus ist eine deutliche Verbesserung in allen drei Konfigurationen zu beobachten. Mit Zunahme der Datenmenge ist mit einer verringerten positiven Auswirkung zu rechnen.

Sehr deutlich zeigt sich dabei eine starke Erhöhung der Ergebnisse bei 25% der Daten. Dabei sind Verbesserungen von 14,4% für die Accuracy und 16,4 % bei dem F_1 Maß festzustellen. Vergleichbare Unterschiede in den Kennzahlen können auch bei 8 Dialogen - entsprechend 5% der Daten - beobachtet werden. Bei großer Anzahl an Trainingsdaten ist bei Betrachtung des Diagramms nicht von einer Verbesserung der Genauigkeit auszugehen. Deutliche Abweichungen zwischen den einzelnen Konfigurationen können nicht festgestellt werden, so befinden sich die Ergebnisse der unterschiedlichen Konfigurationen jeweils im Bereich der zu erwartenden Schwankung. Bei 100% der Trainingsdialoge liegt die Accuracy der Vertauschungen bei 91,8% (grün), 90,8% (rot) und 91,9% (lila). Es lässt sich somit keine Aussage darüber treffen, inwieweit eine erhöhte Anzahl an Vertauschungen in dieser Lernvariante Auswirkungen auf die Genauigkeit verursacht.

Nachdem bereits deutliche Verbesserungen bei dieser Variante der Augmentierung unter Reinforcement Learning festgestellt werden konnten, ist zu ermitteln, ob weitere Anhebungen bei zunehmender Anzahl an Kopien zu bemerken sind. Dies erfolgt, indem die Anzahl der Blöcke mit zwei festgelegt wird und die Anzahl der Kopien zwischen zwei und vier variiert wird. Parametrisiert ist dies somit als $k = 2 \wedge c = (2; 4)$ darstellbar.

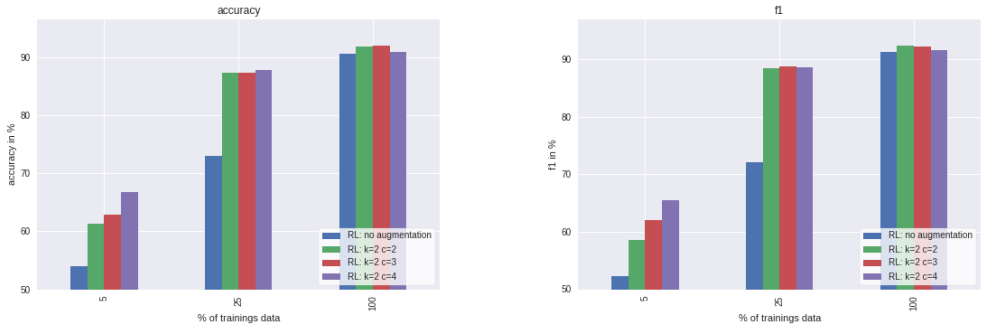


Abbildung 7.16: Augmentierungen mit 2 Vertauschungen bei 2 bis 4 Kopien. Lediglich bei 5% sind Abweichungen bei zunehmender Anzahl an Kopien festzustellen. Weitere deutliche Abweichungen zwischen den Augmentierungen sind nicht zu bemerken.

Bei den in Abbildung 7.16 dargestellten Experimenten können nur sehr geringe Abweichungen zwischen den Ergebnissen der verschiedenen Augmentierungen bei identischer Datenmenge iden-

tifiziert werden. Dies lässt sich bei 25% und 100% der Daten feststellen. Die Abweichungen in den Kennzahlen liegen hierbei jeweils im Bereich der erwarteten Schwankungen. Bei einer geringen Anzahl an Dialogen ist jedoch eine zunehmende Genauigkeit bei höherer Zahl an Kopien zu bemerken. Um festzustellen, ab welcher Anzahl an Dialogen diese Annäherung der Genauigkeit in den Kennzahlen feststellbar werden, werden Experimente zur Identifizierung der zu erwartenden Mittelwerte und der 95%-Quantile durchgeführt.

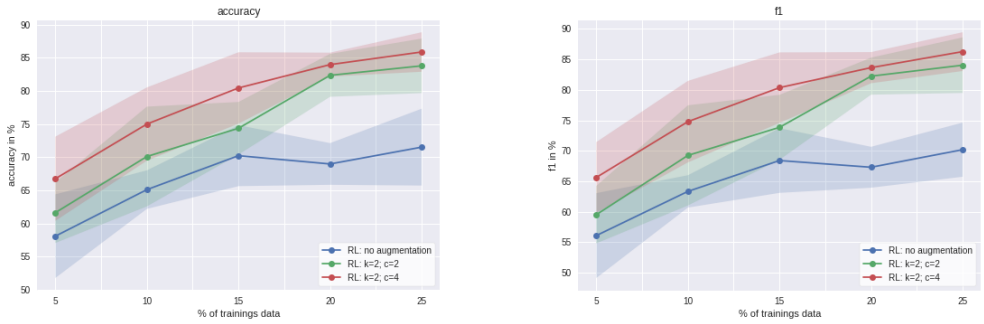


Abbildung 7.17: Die durchschnittliche Genauigkeit bei 4 Kopien (rot) ist durchweg höher als mit 2 Kopien (grün). Ab 15 Dialogen (10% der Daten) ist eine deutliche Signifikanz gegenüber der Kennlinie ohne Augmentierung (blau) ersichtlich. Deutliche Signifikanz ist bei 2 Kopien erst ab 30 Dialogen (20% der Daten) festzustellen.

Die Ergebnisse (siehe Abb. 7.17) zeigen bei zwei Vertauschungen und zwei Kopien (grüne) eine sehr deutliche Verbesserung in beiden Kennzahlen zwischen 23 und 30 Dialogen, was 20% der gesamten Daten entspricht. Wie bereits in Abbildung 7.16 auf der vorherigen Seite ersichtlich, zeigt sich bei 4 Kopien (rot) bei geringer Datenmenge die höchste Genauigkeit, welche bei mittlerer Datenmenge vergleichbar zu den Genauigkeiten der anderen Augmentierungen ist. Bei zunehmender Menge an Trainingsdialogen nähern sich die beiden in der Abbildung 7.17 abgebildeten Varianten der Augmentierung in ihrer jeweiligen Genauigkeit deutlich an. So steigt die Genauigkeit der Variante mit 2 Kopien überproportional im Vergleich zu jener mit 4 Kopien. Auch dies konnte bereits in den Ergebnissen der Abbildung 7.16 auf der vorherigen Seite beobachtet und nun bestätigt werden.

Somit kann davon ausgegangen werden, dass bei zunehmender Menge an Trainingsdialogen eine Anzahl an Kopien von mehr als zwei nicht zu einer weiteren Anhebung der Accuracy und dem F_1 Maß führt. Bei geringer Anzahl an Trainingsdialogen lässt sich, wie die Ergebnisse zeigen, eine deutliche Verbesserung mit mehr Kopien feststellen.

Ablation

Es ist festzustellen, ob sich die Verbesserung der Kennzahlen auch auf die Augmentierung oder ausschließlich auf die Vervielfachung der Daten durch Kopien zurückführen lässt. Bei dem umgesetzten Reinforcement Learning Algorithmus führt eine größere Menge an Trainingsdaten zu einer Zunahme an Episoden während des Trainings. Um dies festzustellen, werden jeweils fünf Experimente mit 5% bis 25% der Dialoge ohne Augmentierung bei dreifacher Datenmenge durchgeführt. Die Ergebnisse werden daraufhin mit denen der Augmentierung mit $k = 2 \wedge c = 2$ verglichen, da diese Augmentierung auch die Anzahl der Dialoge durch zwei Kopien verdreifacht.

Die Ergebnisse der Abbildung 7.18 auf der nächsten Seite zeigen, die zuvor bemerkten Verbesserungen lassen sich vollständig auf die erhöhte Zahl an Episoden zurückführen. Bis einschließlich

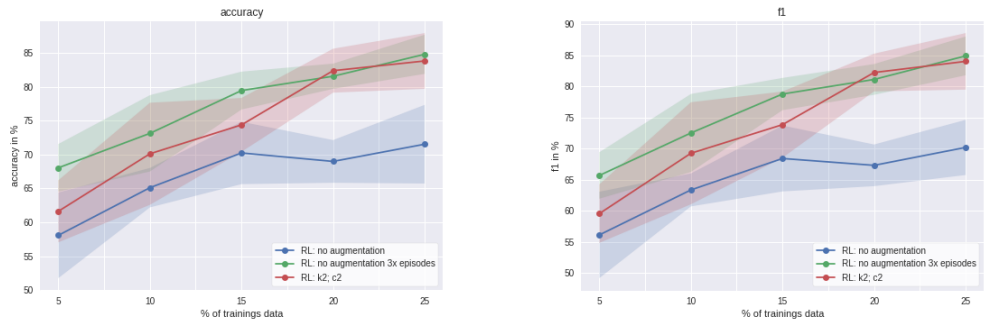


Abbildung 7.18: Verbesserung der Kennzahlen ist auf die erhöhte Anzahl an Episoden und nicht auf die Augmentierung zurückzuführen. Ohne Augmentierung und mit dreifacher Anzahl an Episoden (grün) lässt sich bis 15% der Daten ein durchschnittlich besseres Ergebnis, als mit Augmentierung (rot) feststellen. Ab 20% der Daten sind durchschnittliche Ergebnisse nahezu identisch bei geringerer Schwankung ohne Augmentierung.

15% der Daten lässt sich ohne Augmentierung bei identischer Anzahl an Episoden ein besseres Ergebnis feststellen. Eine Signifikanz kann jedoch nicht beobachtet werden. Wird lediglich der durchschnittliche Wert betrachtet lässt sich ab 30 Dialogen, 20% der Daten, keine Abweichung zwischen erhöhter Anzahl an Episoden und der Augmentierung identifizieren. Bei Betrachtung der 95% Quantile fällt bei Augmentierung eine größere Schwankung der einzelnen Experimente auf. Aufgrund dessen ist die erhöhte Genauigkeit und das verbesserte F_1 Maß vollständig auf die Erhöhung der Anzahl an Episoden zurückführen.

7.2.2 Chitchat

Aufgrund der Erkenntnisse der vorherigen Evaluation dieser Art der Augmentierung wird darauf verzichtet, ausschließlich originale Dialoge ohne Kopien zu trainieren. Zu Beginn dieser Evaluation erfolgen Experimente mit den zwei Ausprägungen der Chitchat Korpora *single*, wobei gilt $|D_C| = 1$, und *multi* mit 100 eindeutigen Chitchat Tupeln. Die Dialoge werden ein bis zweimal kopiert und Chitchat aus dem jeweiligen Korpus wird pro Dialog ein bis zweimal eingefügt. Die Ergebnisse dieser Experimente sind in 7.19 auf der nächsten Seite abgebildet.

Bei Betrachtung der in Abbildung 7.19 auf der nächsten Seite dargestellten Diagramme wird vor allem eine deutliche Steigerung der Accuracy und des F_1 Maßes bei Augmentierungen gegenüber den Ergebnissen ohne Augmentierung bei geringer bis mittlerer Datenmenge. Im Vergleich zur Augmentierung durch Vertauschung ist eine noch markantere Verbesserung in den Kennzahlen festzustellen. Mit einer Erhöhung der Accuracy von 16,3% und 17,9% bei dem F_1 Maß gegenüber den Werten ohne Augmentierung bei 25% der Daten. Somit liegt die Verbesserung nochmals bei Accuracy und F_1 Maß jeweils 1,4% höher als bei dem besten Ergebnis der Vertauschungen (siehe Abbildung 7.16 auf Seite 48).
In den Experimenten, welche mit dem *single* Korpus durchgeführt wurden (siehe Abb. 7.19 auf der nächsten Seite oben), lässt sich keine Abhängigkeit zwischen der Genauigkeit und der Anzahl der Kopien oder der Einfügungen ablesen. Bei vollem Datenzugriff deutet sich bei mehrfacher Einfügung eine Verbesserung an, diese liegt jeweils im Bereich der Schwankungen. In allen Fällen ist eine Verbesserung gegenüber den Kennzahlen ohne Augmentierung festzumachen.
Bei den in Abbildung 7.19 auf der nächsten Seite unten abgebildeten Experimenten mit 100 verschiedenen Chitchats lässt sich eine geringere Schwankung zwischen den einzelnen Ausprägungen feststellen. So liegen bei 25% der Daten zwischen den schlechtesten (rot) und besten (gelb)

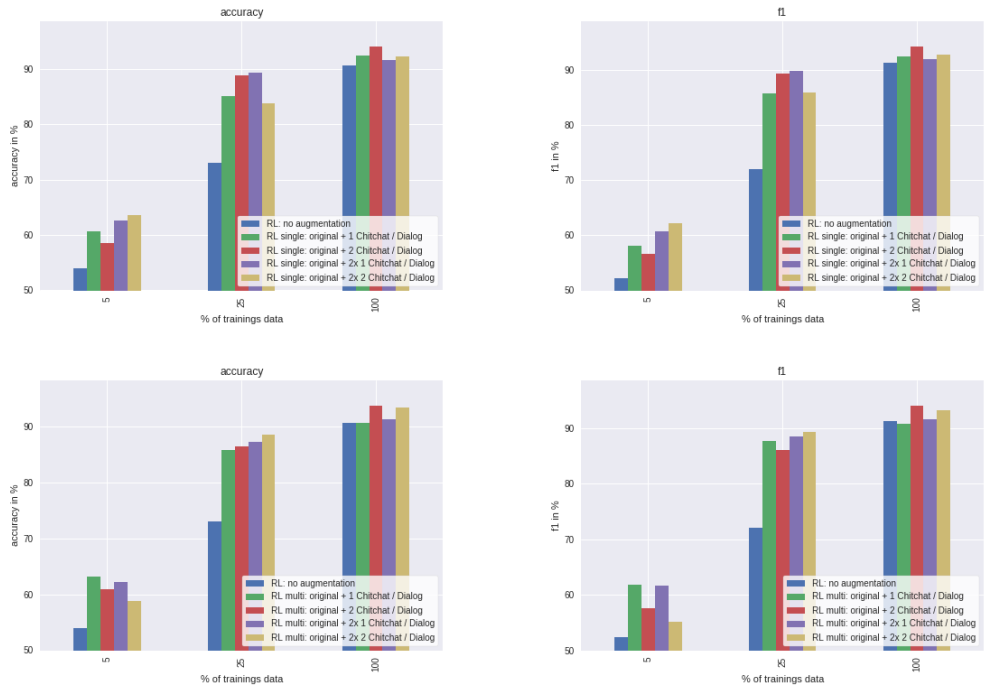


Abbildung 7.19: Experimente mit den zwei Chitchat Korpora *single* (oben) und *multi* (unten) mit den Konfigurationen *original + 1-2 mal 1-2 Chitchat/Dialog*. Alle Experimente weisen deutliche Verbesserung gegenüber Vergleichswert auf. Deutlichste Verbesserungen bei 25% der Daten ersichtlich. Auffallend ein hoher Unterschied zwischen Kennzahlen der Experimente bei *single* Korpus und 25% im Vergleich zu geringen Unterschied bei *multi* Korpus.

lediglich 2,7% (Accuracy) und 3,2% (F_1 Maß). Mit dem *single* Chitchat Korpus, wie in der Abbildung oben dargestellt, beträgt dieser Unterschied 5,5% und 4,0%. Damit ist die Konfiguration zwischen den einzelnen Konfigurationen in dem kleineren Korpus deutlich höher. Der geringe Unterschied zwischen den einzelnen Experimenten im *multi* Korpus ermöglicht es nicht, Aussagen über Auswirkungen der einzelnen Konfigurationen zu treffen. Es ist lediglich festzustellen, dass mehrere Chitchats pro Dialog scheinbar zu einer erhöhten Genauigkeit bei identischer Anzahl an Kopien führt.

Eine deutliche Verbesserung der Ergebnisse ist durch die Diagramme der Abbildung 7.19 gegenüber der Ergebnisse ohne Augmentierung gezeigt worden. Nicht betrachtet ist das Ausmaß der Verbesserung unter Einbeziehung der zu erwartenden Schwankungen. Dafür erfolgt ein Vergleich zwischen den Ergebnissen ohne Augmentierung und Ergebnissen mit der Augmentierung der Konfiguration *single: original + 2 Chitchat/Dialog*. Diese Konfiguration wurde gewählt, da sie die höchsten Ergebnisse bei 5% und 25% aufweist, bei geringerem Aufwand an Trainingszeit im Vergleich zu *single: original + 2x 1 Chitchat/Dialog* Letzteres weist eine 0,6% höhere Accuracy und ein 0,5% höheres F_1 Maß auf, jedoch die Laufzeit der Trainings um ein Vielfaches höher.
In dieser Arbeit nicht abgebildete Experimente ergaben eine geringe durchschnittliche Verbesserung der Ergebnisse, erfolgt wie im Abschnitt 5.2.2 auf Seite 25 eine Rückführung. Aufgrund dessen sind die folgenden Experimente mit einer solchen Rückführung durchgeführt.

Die in Abbildung 7.20 auf der nächsten Seite gezeigten Graphen zeigen Accuracy und F_1 Maß ohne Augmentierung (blau), ohne Augmentierung mit doppelter Anzahl an Episoden (grün) und mit

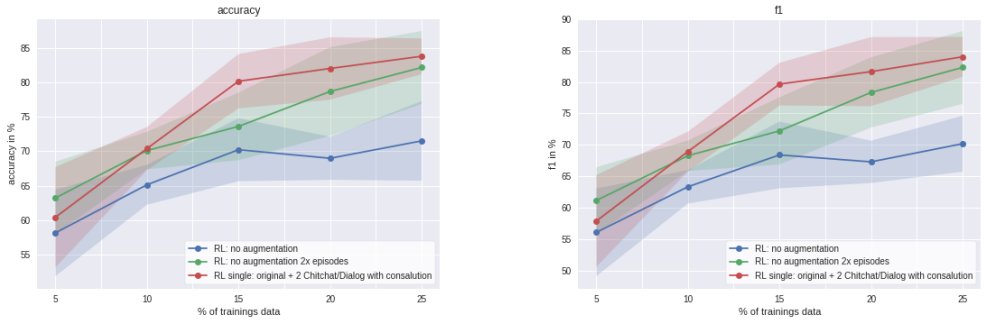


Abbildung 7.20: Konfidenzintervalle für *rl: single original + 2 Chitchat/Dialog mit Rückführung* (rot) im Vergleich zu Kennlinie ohne Augmentierung (blau) und doppelter Anzahl an Episoden (grün) ohne Augmentierung bei 5%-25% der Dialoge. Ab 10% der Daten ist eine durchschnittliche Verbesserung festzustellen und bei 15% der Daten zeigt sich signifikante Verbesserung um 6,6% (Accuracy) und 7,4%.

der bereits beschriebenen Augmentierung *single: original + 2 Chitchat/Dialog with consalution*. Die Gegenüberstellung dieser drei Varianten ermöglicht es, die Effekte der Vervielfachung der Trainingsdaten auf die Kennzahlen und die Auswirkungen der Augmentierung zu betrachten. So lässt sich bei erhöhter Anzahl an Episoden eine klare, teilweise signifikante Verbesserung gegenüber der Standardkonfiguration feststellen. Eine noch deutlichere Verbesserung ist mit der Augmentierung zu bemerken.

So zeigt sich mit Augmentierung eine geringere Schwankung und ab 10% der Dialoge liegen die durchschnittliche Accuracy und das F_1 Maß über Werten ohne Augmentierung bei identischer Anzahl an Episoden. Bei 15% der Dialoge zeigt sich eine signifikante Verbesserung in den Ergebnissen, welche der unten dargestellten Tabelle entnommen werden kann. Diese Verbesserung nimmt jedoch bei zunehmender Anzahl an Dialogen wieder ab. Mit Augmentierung ist jedoch eine geringere statistische Schwankung zwischen den einzelnen Experimenten zu beobachten. So liegt die zu erwartende Schwankung mit Augmentierung bei 25% der Daten bei $\pm 2,6\%$. Ohne Augmentierung liegt dieser Wert bei $\pm 5,3\%$ und ist somit nahezu doppelt so hoch. Eine größere Schwankung, welche sich durch das 95% Quantil zeigt, ist ohne Augmentierung ab 15% der Daten in allen Fällen festzustellen. Die Verbesserung beträgt bei dieser Datenmenge 6,6% und 7,4% bei identischer Anzahl an Episoden.

7.2.3 Generierung neuer Dialoge

Eine geringe Genauigkeit bei der Generierung neuer Dialoge anhand des Graphen ohne Kombination mit ursprünglichen Dialogen ist bereits in einem umfangreichen Maß in der Lernvariante Supervised Learning erkannt worden (siehe 7.1.3). Dies lässt sich mit Reinforcement Learning als genutzte Lernvariante nochmals feststellen. Unter 7.21 auf der nächsten Seite sind Experimente abgebildet, bei denen neue Dialoge mit 30 Interaktionen pro Dialog generiert wurden und die Anzahl der Dialoge um die Faktoren 2 (grün), 3 (rot) und 4 (lila) vervielfacht wurde. Ein zufälliger Wechsel über den Zufallsfaktor *r* erfolgte nicht. In den dort abgebildeten Ergebnissen lässt sich mit zunehmender Menge an Trainingsdaten als Basis zur Generierung keine deutliche Verbesserung der Werte feststellen. Ohne Augmentierung ist eine solche deutlich erkennbar. So steigt das F_1 Maß von 72,0% bei 25% der Daten auf 91,3% bei vollständiger Datengrundlage. Vergleicht man dies mit der Änderung von 65,1% zu 65,9% bei dem Multiplikationsfaktor 2 wird die deutliche Steigerung ohne Augmentierung sehr deutlich. Die Experimente mit den Multiplikationsfaktoren 3 (rot) und 4 (lila) weisen jeweils einen verringertes F_1 Maß bei größerer Menge an Dialogen auf. Bereits im

Tabelle 7.4: Vergleich der Kennzahlen mit Einfügen von Chitchat mit Rückführung und ohne Augmentierung bei 10% bis 25% der Daten. Signifikanz für p<0,05 bei 15% der Daten ist festzustellen.

Anzahl Dialoge in %	Kennzahl	*Original + 2 Chitchat/Dialog mit Rückführung*	*ohne Augmentierung mit 2x Episoden*	*p-Wert*
10%	Accuracy	70,3% ± 3,1%	70,1% ± 2,7%	0,853
	F_1 Maß	68,9% ± 3,1%	68,2% ± 2,4%	0,655
15%	Accuracy	**80,2% ± 3,9%**	**73,6% ± 4,9%**	**0,020**
	F_1 Maß	**79,6% ± 3,3%**	**72,2% ± 5,3%**	**0,011**
20%	Accuracy	82,0% ± 4,5%	78,7% ± 6,4%	0,272
	F_1 Maß	81,6% ± 5,4%	78,3% ± 5,5%	0,274
25%	Accuracy	83,8% ± 2,6%	82,2% ± 5,3%	0,464
	F_1 Maß	83,9% ± 3,1%	82,3% ± 5,7%	0,495

Supervised Learning wurde mit der fehlenden fachlichen Korrektheit der generierten Dialoge ein möglicher Grund genannt. Diese Tatsache konnte bereits in Abbildung 7.10 auf Seite 44 für das Supervised Learning festgestellt werden. Dies ist auch in dieser Lernvariante im selben Maß zu erkennen.

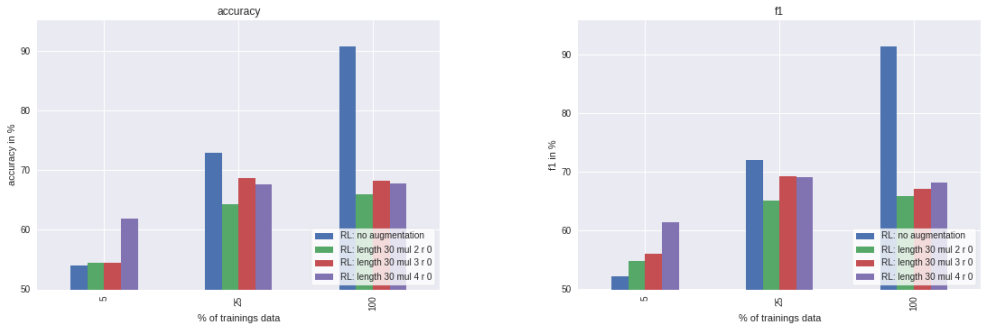


Abbildung 7.21: Generierte Dialoge weisen eine geringere Accuracy und einen geringeren F_1 Maß bei mittlerer und hoher Datenmenge auf. Es ist keine eindeutige Änderung in den Ergebnissen bei steigender Datenmenge von 25% auf 100% zu beobachten.

Eine bereits im Supervised Learning betrachtete Variante kombiniert die generierten Dialoge mit den unveränderten Trainingsdialogen. Die Abbildung 7.22 auf der nächsten Seite zeigt die Genauigkeit und das F_1 Maß der Augmentierungen wie in Abbildung 7.21 abgebildet angereicherten um die originalen Trainingsdaten. Es zeigt sich im Vergleich zu den Ergebnissen in 7.21 eine klare Verbesserung. Bei 5% der Daten lässt sich eine Steigerung von bis zu 11,1% Accuracy und 12,8% F_1 Maß feststellen. Diese kann auf das mehr an Episoden zurückführt werden. Bei identischer Anzahl an Episoden ohne Augmentierung, wie in 7.18 auf Seite 50 durch die grüne Kennlinie dargestellt, lässt sich bei 5% der Daten eine Accuracy von $68,0\% \pm 3,5\%$ feststellen. Mit 65% liegt das beste Ergebnis (grün) mit 5% der Daten unter dem Durchschnittswert innerhalb des 95% Quantils.
Noch deutlicher ist dies bei 25% der Daten zu beobachten. Hier liegt bei identischer Anzahl der Episoden das F_1 Maß bei $84,9\% \pm 3,1\%$. Mit 76,8% liegt das Ergebnis von *original + length 30 mul 2 r 0* deutlich unterhalb der Kennlinie ohne Augmentierung.

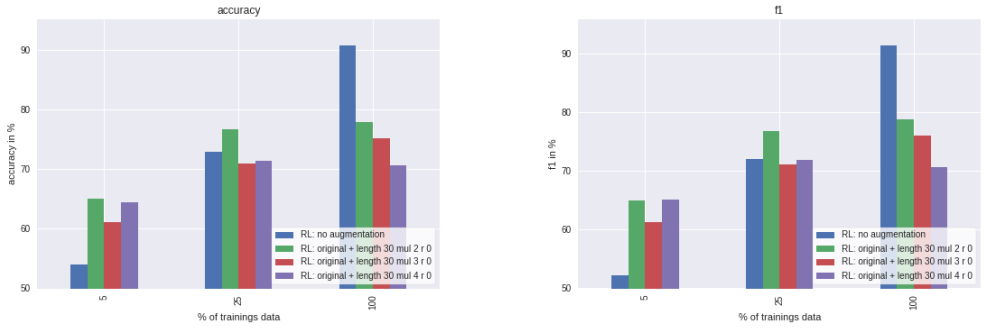


Abbildung 7.22: Generierung neuer Dialoge mit konstanter Länge und unterschiedlichen Multiplikationsfaktoren von 2 bis 4 ohne Zufallsfaktor in Kombination mit originalen, unveränderten Trainingsdaten. Verbesserung bei 5% und 25% der Daten im Gegensatz zu Ergebnissen in Abb. 7.21 auf der vorherigen Seite, lässt sich auf erhöhte Anzahl an Episoden durch dynamische Ermittlung anhand der Anzahl an Trainingsdaten zurückführen.

Abschließend ist die Auswirkung des Zufallsfaktors r auf das Dialog Management zu evaluieren. Die Abbildung 7.23 zeigt Accuracy und F_1 Maß bei originalen Trainingsdaten, kombiniert mit generierten Dialogen mit einer Länge von 30 Interaktionen, einem Multiplikationsfaktor von 2 und einem Zufallsfaktor r von 0% (grün), 5% (rot), 10% (lila), 15% (gelb) und 20% (türkis). Bei geringer bis mittlerer Datenmenge ist mit einem Zufallsfaktor $r > 0\%$ eine generelle Verschlechterung beider Kennzahlen beobachtbar. Es zeigt sich jedoch eine Verbesserung zwischen $r > 0\%$ und $r \leq 15\%$, wobei die Steigerung bei geringer und mittlerer Datenmenge nicht das Niveau ohne Zufallsfaktor oder ohne Augmentierung erreicht. Eine Begründung für diesen Anstieg der Genauigkeit bis zu einem Zufallsfaktor $r = 15\%$ ließ sich nicht finden. Da eine Verbesserung der Kennzahlen im Vergleich zu den Ergebnissen ohne Augmentierung in keinem Experiment zu verzeichnen ist, erfolgt keine weitere Betrachtung dieser Methode.

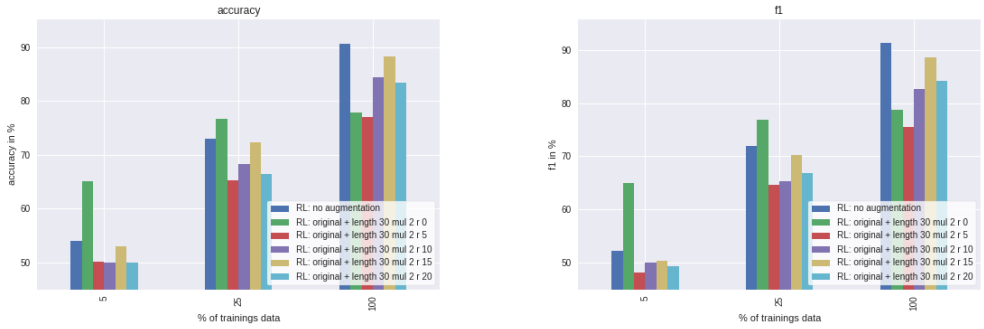


Abbildung 7.23: Es lässt sich mit dem Zufallsfaktor $r > 0$ keine Verbesserung der Kennzahlen im Vergleich zu der Kennlinie (blau) feststellen. Bei zunehmendem r Wert, ausgenommen $r = 0$, ist zunächst eine Verbesserung zu bemerken, jedoch erreicht diese ihren Höhepunkt zwischen $r = 15$ und $r = 20$. Verbesserung bis auf Vergleichswert oder höher konnte nicht beobachtet werden.

8 Diskussion

Dieses Kapitel wird die in der Evaluation festgestellten Ergebnisse diskutieren, um so Aussagen über die einzelnen Augmentierungen und die Lernvarianten zu treffen. Hierfür wird zunächst ein genereller Vergleich zwischen den beiden Lernvarianten ohne Anwendung der Augmentierungen gezogen. Daraufhin werden die einzelnen Augmentierungen unter Verwendung der beiden Lernvarianten und in den unterschiedlichen Konfigurationen der Augmentierungen diskutiert werden. Eine Zusammenfassung wird abschließend die Möglichkeiten der betrachteten Augmentierungen aufgrund der Ergebnisse der Evaluation darlegen.
Die Diskussion wird nicht die Ergebnisse der unter 7.1.1 dargelegten Experimente mit dem MultiWOZ 2.1 Korpus behandeln. Es erfolgt lediglich die Betrachtung der Ergebnisse des im Abschnitt 4.2 beschriebenen Korpus.

8.1 Lernvarianten

Dieser Abschnitt dient dazu, um einen grundlegenden Vergleich zwischen beiden Lernvarianten Supervised Learning und Reinforcement Learning im Dialog Management zu ziehen. Auf die verschiedenen Augmentierungen wird nicht eingegangen. Neben den bereits in Abbildung 7.14 auf Seite 47 dargestellten deutlichen Unterschieden zwischen den Ergebnissen in der Accuracy und dem F_1 Maß ist auch der höhere Zeitaufwand zu nennen, so beträgt das Training bei 25% der Daten im Supervised Learning 00:04:19h $\pm$ 00:21 Minuten, wohingegen das Training im Reinforcement Learning bei identischen Daten und identischer Hardware 01:03:21 h $\pm$ 03:54 Minuten beträgt. Als Grundlage dieser Berechnung dienten jeweils fünf Experimente. Ein Reinforcement Learning Training benötigt somit über 15 mal mehr Zeit als Supervised Learning und die daraus resultierenden Genauigkeiten sind geringer. Des Weiteren konnte festgestellt werden, dass eine genauere Anpassung der Hyperparameter im Reinforcement Learning notwendig ist, um Ergebnisse zu erzielen, welche vergleichbar zu denen des Supervised Learning sind. Um im Reinforcement Learning vergleichbare oder bessere Ergebnisse als durch Supervised Learning zu erzielen sind weitere Anpassungen der Hyperparameter nötig. So konnte bereits durch die Experimente, dargestellt in der Abbildung 7.18 auf Seite 50, gezeigt werden, dass eine Erhöhung der Zahl der Episoden bei geringer bis mittlerer Datenmenge zu einer deutlichen Verbesserung der Genauigkeit führt. Weitere Hyperparameter sind beispielsweise der Discount oder der Reward, bei welchen eine Anpassung zu positiven Effekten führen kann.
Ist nicht davon auszugehen, dass Feedback von Benutzern für zukünftige Trainings genutzt wird, ist jedoch von Reinforcement Learning in der hier umgesetzten Implementierung abzuraten. Wird Feedback für weitere Trainings genutzt, so lassen sich positive Aspekte des Reinforcement Learning nutzen. Ist diese Möglichkeit jedoch nicht vorgesehen, so ist aufgrund der geringeren Genauigkeit und des erhöhten Zeitaufwands während des Trainings von Reinforcement Learning im Dialog Management abzuraten und es ist auf Supervised Learning als Lernvariante zu setzen.

8.2 Augmentierungen

In dieser Arbeit wurden die drei Augmentierungen Vertauschung, Chitchat einfügen und Generierung anhand eines Graphen beschrieben und unter den verschiedenen Lernvarianten evaluiert. Im Weiteren werden die Ergebnisse dieser drei Augmentierungen diskutiert und ein Vergleich der

drei Varianten erfolgt.

8.2.1 Vertauschung

Die Vertauschung basiert in Grundlagen an der Permutation, beschrieben in [14]. Dort konnten mit dieser Augmentierung Verbesserungen von 1% bis 3% bei verschiedenen Korpora und neuronalen Netzen festgestellt werden. Unter Reinforcement Learning als Lernvariante konnte auch hier zunächst eine signifikante Verbesserung der Accuracy und des F_1 Maßes bei geringer und mittlerer Datenmenge festgestellt werden.
Im Supervised Learning zeigte sich, dass eine Zunahme an Vertauschungen zu einer geringeren Genauigkeit führen kann. Diese Erkenntnis unterscheidet sich von den Ergebnissen, wie sie in [14] dargestellt sind. Dort lassen die abgebildeten Ergebnisse den Schluss zu, dass eine mehrfache Vertauschung zu einer verbesserten Genauigkeit führt. Nicht angegeben sind dabei die Schwankungen und die Anzahl der Experimente. Es kann somit nicht sicher festgestellt werden, ob es sich um eine signifikante Steigerung handelt oder ob diese innerhalb der statistischen Schwankungen liegt. In dieser Arbeit konnte festgestellt werden, dass im Supervised Learning eine erhöhte Anzahl an Vertauschungen nach dem im Abschnitt 5.1 auf Seite 22 beschriebenen Verfahren zu einer Verringerung der Genauigkeit führt. Innerhalb der jeweiligen Ablationen, abgebildet in den Abbildungen 7.8 auf Seite 42 und 7.18 auf Seite 50, konnte keine Verbesserung der Genauigkeit oder des F_1 Maßes festgestellt werden, welche auf die Augmentierung zurückzuführen ist. Somit ist von einer Verwendung dieser Art der Augmentierung abzuraten. Eine Verbesserung konnte in keiner der beiden Lernvarianten festgestellt werden. Verbesserungen im Reinforcement Learning ließen sich vollständig auf die erhöhte Anzahl an Episoden zurückführen.

8.2.2 Chitchat einfügen

In beiden Lernvarianten konnte eine Verbesserung durch das Einfügen von Chitchat nachgewiesen werden. Bereits durch das Einfügen von einem Chitchat pro Dialog bei einer Mächtigkeit des Chitchat Korpus von $|D_c| = 1$ konnte im Supervised Learning eine Verbesserung von 1,2% Accuracy (p=0,222) und 1,2% F_1 Maß (p=0,229) festgestellt werden. Verschlechterungen konnten durch das Einfügen von Chitchat nicht festgestellt werden. Unabhängig der verwendeten Lernvariante oder der Anzahl der Einfügungen.

Es konnte keine signifikante Unterscheidung zwischen den beiden Varianten des Chitchat Korpus D_c festgestellt werden. So liegt der höchste Wert der Accuracy bei komplettem Datenbestand für $|D_c| = 1$ bei 94,1% und für $|D_c| = 100$ bei 94,6% (siehe Abb. 7.6 auf Seite 40). Der festgestellte Unterschied liegt innerhalb einer Standardabweichung. Werden lediglich die Ergebnisse bei ausschließlichem Einfügen des Chitchats ohne der Durchführung von Kopien betrachtet, ist eine geringe Verbesserung bei größerem Chitchat Korpus festzustellen. Erfolgt jedoch die Betrachtung der Ergebnisse, unter Hinzufügen der original unveränderten Dialoge (siehe Abb. 7.7 auf Seite 41 und Abb. 7.19 auf Seite 51), so lässt sich diese Beobachtung nicht verfestigen. Dies lässt den Schluss zu, dass die Anzahl der unterschiedlichen eingefügten Chitchats nicht von Relevanz ist. Somit führt lediglich das Einfügen von Chitchat selbst zu einer Verbesserung und es ist nicht notwendig einen mächtigen Korpus an Chitchat bereitzustellen. Diese Ergebnisse lassen sich in dieser Form lediglich auf die Lernvariante Supervised Learning anwenden. Im Reinforcement Learning konnte unter Verwendung des Korpus mit ausschließlich einem Chitchat eine größere Schwankung zwischen den einzelnen Konfigurationen festgestellt werden. Eine Begründung für diese Feststellung lässt sich zum jetzigen Zeitpunkt nicht finden. Die in 7.20 auf Seite 52 abgebildeten Ergebnisse weisen jedoch nicht darauf hin, dass innerhalb der einzelnen Konfigurationen das hohe Maß an Schwankungen auftritt. Mögliche Gründe hierfür können die Wahl der Parameter, die Verteilung der Intents innerhalb des Korpus oder die Struktur der Dialoge darstellen. Um eine genauere Aus-

sage hierüber zu treffen, wären weitere Experimente notwendig.

Im Abschnitt 5.2.2 auf Seite 25 wurde eine Variante des Einfügens von Chitchat mit der Rückführung beschrieben. Die Überlegung dabei bestand darin, dass nach Reaktion auf das Chitchat dem Benutzer die letzte Frage nochmals gestellt wird, um so die erwartete nächste Reaktion des Benutzers hervorzuheben. Die Überlegung dieser Variante bestand lediglich in einer erhöhten Benutzerfreundlichkeit. Die Ergebnisse der Abbildung 7.20 auf Seite 52 zeigten positive Effekte auf die Accuracy und den F_1 Maß in den durchgeführten automatisierten Tests. Diese Erkenntnis konnte auch im Reinforcement Learning bestätigt werden. Das Einfügen von Rückführungen nach dem Chitchat konnte zu einer weitere Verbesserung der durchschnittlichen Kennzahlen führen.

Die Kombination aus originalen Dialogen und zwei Kopien der Dialoge, welche mit einem Chitchat pro Dialog mit Rückführung augmentiert wurden, führten somit zu einer Verbesserung von bis zu 1,6% Accuracy und 1,8% F_1 Maß bei 25% der gesamten Dialoge für das Training. Der p-Wert beträgt hierbei $p = 0,068$ (Accuracy) und $p = 0,057$ (F_1 Maß). Die Ablation zeigte, dass die Verbesserungen nicht ausschließlich auf die erhöhte Anzahl an Kopien zurückzuführen ist. Hierfür wurde das Training ohne Augmentierung mit der dreifachen Menge an Epochen durchgeführt. Als p-Werte ergaben sich hier $p = 0,126$ und $p = 0,194$. Somit ist die Verbesserung der Kennzahlen auch auf die Augmentierung und nicht ausschließlich auf die erhöhte Anzahl an Dialogen zurückzuführen. Im Reinforcement Learning zeichnet sich ein vergleichbares Bild. Hier konnten mit den ursprünglichen Dialogen, einer Kopie und zwei Chitchats pro Dialogen in den augmentierten Dialogen, eine deutliche Verbesserung von 13,7% bei 25% der Daten erzielt werden. Die Ablation weißt darauf hin, dass sich die Verbesserung zu einem großen Teil auf die erhöhte Anzahl an Dialogen zurückführen lässt. Es konnte jedoch bei 15% der Daten eine signifikante Verbesserung von 6,6% Accuracy und 7,4% F_1 Maß festgestellt werden ($p = 0,020$ und $p = 0,011$).

Die in dieser Arbeit gesammelten Erkenntnisse weisen darauf hin, dass Einfügen von Chitchats innerhalb von Dialogen zu verbesserten Genauigkeiten im Dialog Management führt. Eine Rückführung führt zu einer weiteren Verbesserung. Eine klare Unterscheidung der Auswirkungen in den beiden Lernvarianten konnte nicht festgestellt werden. Es konnte des Weiteren in den vorgenommenen Konfigurationen keine Verschlechterung gegenüber der Kennlinie ohne Augmentierung festgestellt werden. Somit führten alle Augmentierungen nach diesem Verfahren zu keiner Verschlechterung und bei korrekter Parametrisierung zu einer signifikanten Verbesserung bei $p < 0,1$.

8.2.3 Generierung neuer Dialoge

Das Verfahren anhand graphenbasierter Generierung von Dialogen erwies sich als die Augmentierung mit der geringsten Genauigkeit, wenn ausschließlich die generierten Dialoge für das Training genutzt werden. Dies zeigte sich in beiden Lernvarianten deutlich. Die Generierung der Daten zeigte keine Verbesserung der Ergebnisse, bei zunehmender Anzahl an Dialogen, welche als Basis zur Generierung dienen. So sind keine deutlichen Unterschiede in den Kennzahlen zwischen 5% der Dialoge und 25% der Dialoge beziehungsweise zwischen 25% und 100% festzustellen (siehe Abb. 7.10 auf Seite 44 und Abb. 7.21 auf Seite 53). Die Tabelle 7.3 auf Seite 44 zeigt die Steigerung der Accuracy bei geringster Datenmenge von $62,7\% \pm 4,6\%$ hin zu 67,7% ± 2,4% bei vollem Datenzugriff. Die Steigerung liegt somit lediglich bei 5% bei einer Vervielfachung der Menge der Dialoge im Trainingsset um den Faktor 20. Diese geringe Steigerung ist gegenüber $+18,3\%$ ohne Augmentierung deutlich geringer. Hierbei stellt die genutzte Lernvariante keinen Unterschied dar. Die

[1] 1. Dialog beginnt mit Intention des Benutzers. 2. Auf Intention folgt Aktion des Assistenten. 3. Auf Aktion folgt Intention oder weitere Aktion.

Hypothese, dass durch einen Dialoggraphen erzeugte zusätzliche Dialoge zu einer Verbesserung führen, konnte somit deutlich widerlegt werden, wenn ausschließlich generierte Dialoge für das Training des Dialog Management genutzt werden. Die generierten Dialoge weisen zwar syntaktische Korrektheit, nach im Kapitel 5 auf Seite 20 beschriebener Definition[1], auf, jedoch lässt sich mit dieser Methodik keine fachliche Korrektheit prüfen. Diese ist jedoch von Relevanz, sollen neue Dialoge generiert werden. Trainiert der Lernalgorithmus fachlich inkorrekte Dialoge, so können auch keine korrekten Vorhersagen getroffen werden.

Bei Verwendung der generierten Dialoge als Anreicherung der Dialoge können Verbesserungen festgestellt werden. So war es bereits mit 25% der Daten möglich, eine identische Genauigkeit, wie bei 100% der Daten ohne Augmentierung (siehe Abb. 7.11 auf Seite 45) zu erzielen. Diese Verbesserung ließ sich jedoch auf die Trainingsdaten zurückführen. In diesem speziellen Fall wiesen sie eine überdurchschnittliche Länge im Vergleich zu den restlichen Trainingsdaten mit 25% der Dialoge auf. Es konnte somit dieses Maß an Verbesserung nur in einem Fall ermittelt werden. Das führt zur Erkenntnis, dass diese Art der Augmentierung sehr von der Qualität der Trainingsdialoge abhängt. Es konnten Verbesserungen festgestellt werden, doch hielten diese einem statistischen Test nicht stand. Es konnte keine Verschlechterung, aber auch kein signifikante Verbesserung festgestellt werden. Auch ist die fehlende Betrachtung der inneren Logik der Dialoge innerhalb der Generierung als Problem in dieser Variante anzusehen. So wird während der Generierung nicht darauf geachtet, dass fachlich korrekte Dialoge erzeugt werden. Die Auswirkung dieser fehlenden Implementierung zeigt sich beispielsweise darin, dass bei zunehmender Anzahl an generierten Dialogen die Genauigkeit abnimmt (siehe Abb. 7.11 auf Seite 45). Zunächst zeigt sich durch die generierten Dialoge ein positiver Effekt, jedoch bei steigender Anzahl an generierten Dialogen, im Vergleich zu ursprünglichen Dialogen, sinkt die Genauigkeit der Vorhersagen.
Im Reinforcement Learning zeigte sich hingegen bei geringer Anzahl an Trainingsdialogen eine deutliche Verbesserung. Diese ist jedoch vollumfänglich auf die erhöhte Anzahl an Episoden zurückzuführen. Bei vollem Zugriff auf die Trainingsdialoge konnte eine deutliche Verschlechterung der Kennzahlen beobachtet werden (siehe Abb. 7.22 auf Seite 54). Dies lässt sich möglicherweise damit begründen, dass während des Trainings durch die fachlich inkorrekten Dialoge ein inkorrekter Reward ermittelt wird. Dies führt in diesem Fall zu einer stärkeren Verschlechterung der Kennzahlen, als dies im Supervised Learning erkennbar ist.

Mit dem Zufallsfaktor r wurde versucht neben der Generierung anhand des Graphen eine höhere Variabilität in die generierten Dialoge einfließen zu lassen. Es bestand die Hypothese, dadurch besser auf unerwartete Interaktionen einzugehen und ein Overfitting zu vermeiden. Im Reinforcement Learning konnte durch die in Abbildung 7.23 auf Seite 54 dargestellten Experimente diese Hypothese teilweise bestätigt werden. So lässt sich bei vollem Datenzugriff eine deutliche Verbesserung mit einem Zufallsfaktor von 10% bis 15% im Vergleich zu den Ergebnissen ohne Zufallsfaktor erkennen. Durch die Verbesserung gegenüber der Augmentierung ohne Zufallsfaktor konnte jedoch nicht die Genauigkeit ohne Augmentierung erreicht werden. Im Supervised Learning konnte auch bei vollem Zugriff auf die Trainingsdaten bei geringem Zufallsfaktor von 5% eine Verbesserung ermittelt werden. Dadurch konnte auch die Accuracy und der F_1 Maß der Kennlinie im Durchschnitt übertroffen werden (siehe Abb. 7.13 auf Seite 46). Durch einen p-Test konnte jedoch keine Signifikanz festgestellt werden (Accuracy: $p = 0,53$; F_1 Maß: $p = 0,63$). Im Supervised Learning konnte auch eine geringfügige Verbesserung bei geringerer Datenmenge gegenüber der Augmentierung beobachtet werden. Bei 25% der Daten lag diese jedoch unterhalb der Ergebnisse ohne Zufallsfaktor und bei 5% der Daten innerhalb der zu erwartenden Schwankungen. Im Reinforcement Learning zeigte sich bei geringer bis mittlerer Datenmenge eine generelle Verschlechterung gegenüber den Kennzahlen ohne Zufallsfaktor.
Es lässt sich somit anhand der Ergebnisse der Evaluation, dargestellt in den Abbildungen 7.13 auf

Seite 46 und 7.23 auf Seite 54, keine eindeutige Aussage über die im Abschnitt 5.3.2 getroffene Vorhersage über die bessere Genauigkeit durch den Zufallsfaktor *r* getroffen werden. Es lässt sich anhand der durchgeführten Experimente kein Muster erkennen, ab welcher Anzahl an Dialogen im zu argumentierenden Trainingsset ein Zufallsfaktor einen positiven Effekt erbringen kann und ab welcher dieser nicht geeignet ist und zu einer Verschlechterung führt.

Im Reinforcement Learning wird von dieser Variante abgeraten. Es konnten deutliche Verschlechterungen im Vergleich zu der Kennlinie festgestellt werde. Im Supervised Learning konnte eine Verbesserung festgestellt werden, jedoch liegt diese nicht in einem Bereich in dem von einer Signifikanz auszugehen ist. Ist kein umfangreiches Testset vorhanden und somit die Möglichkeit der automatisierten Testung nur begrenzt, so ist von dieser Art der Augmentierung abzuraten.
Eine Verwendungsmöglichkeit des Algorithmus besteht darin, ihn im Preprocessing zu nutzen und nicht als Augmentierung. Dadurch besteht die Möglichkeit nach Generierung die Dialoge auf ihre fachliche Korrektheit zu überprüfen und somit eine hohe Qualität der Trainingsdaten zu gewährleisten. Dabei würde es sich jedoch nicht um eine Variante der Augmentierung handeln, sondern um eine Art der Datenbeschaffung über Generierung, wie in 3.4.3 auf Seite 11 beschrieben.

8.3 Zusammenfassung

In dieser Arbeit konnte gezeigt werden, dass Augmentierung einen positiven Effekt auf die Genauigkeit des Dialog Managements von Gesprächsassistenten haben kann. Die besten Ergebnisse ließen sich durch das Einfügen von Chitchats innerhalb der vorhandenen Dialoge mit einer anschließenden Rückführung erreichen. Diese Verbesserung ließ sich sowohl mit der Lernvariante Supervised Learning als auch mit Reinforcement Learning feststellen. Die Augmentierungen *Vertauschung* und *Generierung neuer Dialoge* führten hingegen nicht zu signifikanten Verbesserungen der Kennzahlen. Ein möglicher Grund hierfür ist, dass durch das Einfügen des Chitchats lediglich die Varianz der Dialoge erhöht wird, jedoch nicht der Inhalt an sich geändert wurde. Durch die Vertauschung und die Generierung kann der fachliche Inhalt der Dialoge in einem Maß geändert werden, dass dieser für ein Training nicht geeignet ist. Es zeigte sich bei Chitchat und Vertauschung keine generelle Unterscheidung zwischen den Ergebnissen die abhängig von der Lernvariante sind. Bei Generierung der Dialoge zeigte sich ein deutlicher Unterschied, was dazu führt, dass diese Art der Augmentierung im Supervised Learning zu einer geringen Verbesserung führen kann und im Reinforcement Learning zu einer deutlichen Verschlechterung der Kennzahlen führte.
Besteht nicht die Möglichkeit für zukünftige Trainings das Feedback von Benutzern zu nutzen, ist von Reinforcement Learning in der hier beschriebenen Variante abzuraten. Die gemessenen Ergebnisse zeigen jeweils eine deutlich geringere Genauigkeit im Gegensatz zu den mit Supervised Learning erreichten. Des Weiteren ist ein deutlich höherer Zeitaufwand im Training und eine größere Anzahl an zu konfigurierenden Hyperparametern mit dieser Lernvariante verbunden. Aufgrund der hier gesammelten Erkenntnisse ist festzustellen, dass bei identischer Menge an Trainingsdaten Supervised Learning für Dialog Management zu einer höheren Genauigkeit in automatisierten Tests führt.
Da durch Chitchat in keinem der Experimente eine Verschlechterungen der Kennzahlen festgestellt wurde, kann anhand dieser Erkenntnisse zu einer Verwendung dieser Art der Augmentierung geraten werden. Die Vertauschung und die Generierung führen in dieser Arbeit zu keinen Verbesserungen und somit ist hiervon abzuraten. Jedoch kann eine Generierung im Preprocessing mit anschließender Qualitätskontrolle, automatisiert oder durch einen Menschen, zu einer Verbesserung der Genauigkeit führen.

9 Ausblick in die Zukunft

Die Ergebnisse der Evaluation und die anschließende Diskussion konnten zeigen, dass Augmentierungen bei automatisierten Tests zu Verbesserungen führen können. Im nächsten Schritt ist eine Bewertung der Augmentierung durch einfügen von Chitchat mithilfe von Testpersonen durchzuführen. So lassen sich Auswirkungen der Augmentierung auf das Dialog Management unter realen Bedingungen prüfen.
Des Weiteren ist die Hypothese der Zunahme der Genauigkeit des Dialog Managements durch Feedback von Usern und nachträgliches Training durch Reinforcement Learning zu überprüfen. So ist festzustellen, ob durch Reinforcement Learning mit Feedback von Benutzern als Reward und Augmentierung eine Verbesserung gegenüber dem Supervised Learning erzielt werden kann. Auch hier sind automatisierte Tests und Testpersonen für die Überprüfung der Hypothese möglich.
Das in dieser Arbeit vorgestellte Verfahren zur Generierung von Dialogen stellte sich in seiner beschriebenen Ausprägung als unzuverlässig heraus. Eine Verbesserung von dem Algorithmus durch Einbeziehung des fachlichen Kontextes in die Generierung kann zu einer praktischen Nutzbarkeit des Algorithmus führen. Es ist sicherzustellen, dass die generierten Dialoge nicht nur syntaktisch korrekt sind, sondern auch eine fachliche Korrektheit aufweisen. Die Erarbeitung eines solchen Algorithmus kann in weiteren Arbeiten in Betracht gezogen werden. Hierbei sind die Ergebnisse aus [56, 54, 55] mit in die Überlegungen zukünftiger Methoden mit einzubeziehen.

In der hier vorliegenden Arbeit wurde für die Evaluation jeweils ausschließlich eine Policy für das Dialog Management genutzt. Das Rasa Framework ermöglicht die Kombination verschiedener Policies. So sind neben der reinen Kommunikation weitere Funktionalitäten, wie das füllen von Formularen durch den Gesprächsassistenten umsetzbar. Die Auswirkungen einer Augmentierung in einem solchen System sind festzustellen, um auch hier künftig Aussagen über die Verwendungsmöglichkeiten der beschriebenen Methoden zu treffen.

Durch eine weitere Verbreitung von Sprachassistenten und Chatbots wird sich auch in Zukunft vielmals Fragen zur Datenbeschaffung für das Dialog Management stellen. Diese Arbeit konnte zeigen, dass Datenaugmentierung in Form des in dieser Arbeit beschriebenen Chitchat einfügen einen positiven Effekt auf die Genauigkeit haben kann.

Danksagung

Ich möchte mich zunächst bei Herr Prof. Dr. David Spieler für die Betreuung während dieser Arbeit bedanken. Im nächsten möchte ich mich bei meinen Betreuern und Kollegen Stephan Schiffner, Jonathan Boidol und Simon Schmitz für die Unterstützung bei dieser Arbeit zu tiefst bedanken. Trotz Pandemie und dem damit verbundenen Home-Office fühlte ich mich immer unterstützt. Vielen herzlichen Dank dafür.
Ein tiefer Dank gilt auch den Korrekturleserinnen Helena Kargl, Katha Pechowski und Pia Geffert.

Ich möchte mich zuletzt bei meiner Familie und meiner Freundin bedanken, die mich während meines Studiums und vor allem während der Masterarbeit sehr unterstützt haben.

Literatur

Quellen

[1] Alan Nichol. *Research Updates from Rasa: Transformers in NLU and Dialogue*. Nov. 2020. URL: https://www.youtube.com/watch?v=7h3vi3UrcSU (besucht am 03. 04. 2020).

[2] Rafael E Banchs. »Movie-DiC: a movie dialogue corpus for research and development«. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2012, S. 203–207.

[3] Christina Bennett und Alexander I Rudnicky. »The carnegie mellon communicator corpus«. en. In: *Seventh International Conference on Spoken Language Processing*. 2002.

[4] Die Bibel. *Einheitsübersetzung, hrsg. im Auftrag der Deutschen Bischofskonferenz, der Österreichischen Bischofskonferenz, der Schweizer Bischofskonferenz, des Erzbischofs von Luxemburg, des Erzbischofs von Vaduz, des Erzbischofs von Straßburg, des Bischofs von Bozen-Brixen, des Bischofs von Lüttich*. Stuttgart, 2016. (Besucht am 25. 03. 2020).

[5] Tom Bocklisch u. a. »Rasa: Open Source Language Understanding and Dialogue Management«. en. In: *31st Conference on Neural Information Processing Systems (NIPS 2017)*. arXiv: 1712.05181. Dez. 2017.

[6] D. Bohus und A.I. Rudnicky. »Constructing accurate beliefs in spoken dialog systems«. en. In: *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005*. San Juan, Puerto Rico: IEEE, 2005, S. 272–277. ISBN: 978-0-7803-9479-7. DOI: 10.1109/ASRU.2005.1566504.

[7] Dan Bohus und Alex Rudnicky. »A "K Hypotheses + Other" Belief Updating Model«. en. In: *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*. Boston, 2006, S. 13–18. ISBN: 978-1-57735-296-9.

[8] Dan Bohus und Alexander I. Rudnicky. »The RavenClaw dialog management framework: Architecture and systems«. en. In: *Computer Speech & Language* 23.3 (Juli 2009), S. 332–361. DOI: 10.1016/j.csl.2008.10.001.

[9] Susan Brennan, Katharina Schuhmann und Karla Batres. »Entrainment on the move and in the lab: The walking around corpus«. In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Bd. 35. 35. 2013.

[10] Paweł Budzianowski u. a. »MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling«. In: *Proceedings of the 2018 Conference on Empirical Methods in NaturalLanguage Processing* (2018). arXiv: 1810.00278, S. 5016–5026.

[11] Cambridge University Press. *Meaning of augmentation in English*. en. Definition of augmentation from the Cambridge Advanced Learner's Dictionary & Thesaurus. URL: https://dictionary.cambridge.org/dictionary/english/augmentation (besucht am 19. 08. 2020).

[12] Cristian Danescu-Niculescu-Mizil und Lillian Lee. »Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs«. In: *Procee-*

dings of the 2nd workshop on cognitive modeling and computational linguistics. Association for Computational Linguistics. 2011, S. 76–87.

[13] Ellen Douglas-Cowie u. a. »The HUMAINE database: Addressing the collection and annotation of naturalistic and induced emotional data«. In: *International conference on affective computing and intelligent interaction*. Springer. 2007, S. 488–500.

[14] Wenchao Du und Alan W. Black. »Data Augmentation for Neural Online Chat Response Selection«. en. In: *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI* (Sep. 2018). arXiv: 1809.00428, S. 52–58. (Besucht am 06. 04. 2020).

[15] Alain Dutech und Bruno Scherrer. »Partially Observable Markov Decision Processes«. en. In: *Markov Decision Processes in Artificial Intelligence*. John Wiley & Sons, Ltd, 2013, S. 185–228. DOI: 10.1002/9781118557426.ch7. (Besucht am 08. 04. 2020).

[16] Mihail Eric u. a. »MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines«. In: *Proceedings of The 12th Language Resources and Evaluation Conference* (Dez. 2019). arXiv: 1907.01669. (Besucht am 28. 04. 2020).

[17] Norman M Fraser und G Nigel Gilbert. »Simulating speech systems«. In: *Computer Speech & Language* 5.1 (1991), S. 81–99.

[18] Jianfeng Gao, Michel Galley und Lihong Li. »Neural Approaches to Conversational AI«. English. In: *Foundations and Trends® in Information Retrieval* 13.2-3 (Feb. 2019). Publisher: Now Publishers, Inc., S. 127–298. DOI: 10.1561/1500000074. (Besucht am 02. 04. 2020).

[19] Felix A. Gers, Jürgen Schmidhuber und Fred Cummins. »Learning to Forget: Continual Prediction with LSTM«. en. In: *Neural Computation* 12.10 (Okt. 2000), S. 2451–2471. DOI: 10.1162/089976600300015015. (Besucht am 10. 04. 2020).

[20] Google Inc. *Dialogflow*. de. Library Catalog: cloud.google.com. Feb. 2020. URL: https://cloud.google.com/dialogflow?hl=de (besucht am 03. 04. 2020).

[21] Google Inc. *Einbindungen | Dokumentation zu Dialogflow*. de. Library Catalog: cloud.google.com. Feb. 2020. URL: https://cloud.google.com/dialogflow/docs/integrations?hl=de (besucht am 03. 04. 2020).

[22] Google Inc. *Small Talk | Dokumentation zu Dialogflow*. de. Library Catalog: cloud.google.com. Feb. 2020. URL: https://cloud.google.com/dialogflow/docs/agents-small-talk?hl=de (besucht am 03. 04. 2020).

[23] Google Inc. *Vordefinierte Agents | Dokumentation zu Dialogflow*. de. Library Catalog: cloud.google.com. Feb. 2020. URL: https://cloud.google.com/dialogflow/docs/agents-prebuilt?hl=de (besucht am 03. 04. 2020).

[24] Braden Hancock u. a. »Learning from Dialogue after Deployment: Feed Yourself, Chatbot!« In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Juni 2019). arXiv: 1901.05415. (Besucht am 11. 03. 2020).

[25] Jan-Gerrit Harms u. a. »Approaches for Dialog Management in Conversational Agents«. en. In: *IEEE Internet Computing* 23.2 (2018), S. 13–22. DOI: 10.1109/MIC.2018.2881519. (Besucht am 31. 03. 2020).

[26] Matthew Henderson, Blaise Thomson und Jason D Williams. »The Second Dialog State Tracking Challenge«. en. In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Philadelphia, PA, U.S.A.: Association for Com-

putational Linguistics, 2014, S. 263–272. DOI: 10.3115/v1/W14-4337. (Besucht am 03. 03. 2020).

[27] Sepp Hochreiter und Jürgen Schmidhuber. »Long Short-Term Memory«. In: *Neural Computation* 9.8 (Nov. 1997). Publisher: MIT Press, S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (besucht am 08. 04. 2020).

[28] Vladimir Ilievski u. a. »Goal-oriented chatbot dialog management bootstrapping with transfer learning«. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. Juli 2018, S. 4115–4121. URL: https://dl.acm.org/doi/abs/10.5555/3304222.3304342 (besucht am 03. 03. 2020).

[29] Jakob Uszkoreit. *Transformer: A Novel Neural Network Architecture for Language Understanding*. en. Library Catalog: ai.googleblog.com. Aug. 2017. URL: http://ai.googleblog.com/2017/08/transformer-novel-neural-network.html (besucht am 23. 07. 2020).

[30] Gyeeun Jeong und Ha Young Kim. »Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning«. en. In: *Expert Systems with Applications* 117 (März 2019), S. 125–138. DOI: 10.1016/j.eswa.2018.09.036. (Besucht am 23. 07. 2020).

[31] Arne Jönsson und Nils Dahlbäck. *Talking to a computer is not like talking to your best friend*. English. OCLC: 18821080. Linköping, Sweden: Linköping University, Dept. of Computer und Information Science, 1988.

[32] Juntao Li u. a. »Insufficient Data Can Also Rock! Learning to Converse Using Smaller Data with Augmentation«. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (Juli 2019). Number: 01, S. 6698–6705. DOI: 10.1609/aaai.v33i01.33016698. (Besucht am 06. 04. 2020).

[33] Xiujun Li u. a. »End-to-End Task-Completion Neural Dialogue Systems«. en. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. arXiv: 1703.01008. Feb. 2018, S. 733–743. (Besucht am 05. 03. 2020).

[34] Jakob von Lindern. »Chatbots: Der Freund in meinem Handy«. de-DE. In: *Die Zeit* (Aug. 2020). ISSN: 0044-2070. URL: https://www.zeit.de/digital/2020-07/chatbots-kuenstliche-intelligenz-smartphone-einsamkeit-corona-kontaktbeschraenkung (besucht am 26. 08. 2020).

[35] Bing Liu und Ian Lane. »Iterative Policy Learning in End-to-End Trainable Task-Oriented Neural Dialog Models«. en. In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. arXiv: 1709.06136. IEEE. 2017, S. 482–489. (Besucht am 03. 03. 2020).

[36] Bing Liu u. a. »Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems«. In: *Proceedingsof the 2018 Conference of the North American Chapter of the Association for ComputationalLinguistics: Human Language Technologies (NAACL-HLT)*. arXiv: 1804.06512. Apr. 2018, S. 2060–2069. (Besucht am 12. 03. 2020).

[37] Chia-Wei Liu u. a. »How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation«. en. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Jan. 2017). arXiv: 1603.08023. (Besucht am 26. 03. 2020).

[38] Ryan Thomas Lowe u. a. »Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus«. en. In: *Dialogue & Discourse* 8.1 (Jan. 2017), S. 31–65. ISSN: 2152-9620. URL: http:

//dad.uni-bielefeld.de/index.php/dad/article/view/3698 (besucht am 05. 03. 2020).

[39] Francois Mairesse u. a. »Phrase-Based Statistical Language Generation Using Graphical Models and Active Learning«. en. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*. 2010, S. 1552–1561.

[40] David Meintrup und Stefan Schäffler. »Markov-Ketten«. de. In: *Stochastik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 227–265. ISBN: 978-3-540-21676-6. DOI: 10.1007/3-540-26707-7_9. (Besucht am 02. 06. 2020).

[41] Volodymyr Mnih u. a. »Human-level control through deep reinforcement learning«. en. In: *Nature* 518.7540 (Feb. 2015), S. 529–533. DOI: 10.1038/nature14236. (Besucht am 05. 03. 2020).

[42] Volodymyr Mnih u. a. »Playing Atari with Deep Reinforcement Learning«. en. In: (Dez. 2013). arXiv: 1312.5602. (Besucht am 05. 03. 2020).

[43] Mehryar Mohri, Afshin Rostamizadeh und Ameet Talwalkar. *Foundations of Machine Learning*. 2nd. The MIT Press, 2018. ISBN: 978-0-262-03940-6.

[44] George E. Monahan. »State of the Art—A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms«. In: *Management Science* 28.1 (Jan. 1982), S. 1–16. DOI: 10.1287/mnsc.28.1.1. (Besucht am 05. 03. 2020).

[45] Takafumi Okuyama, Tad Gonsalves und Jaychand Upadhay. »Autonomous Driving System based on Deep Q Learnig«. In: *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*. März 2018, S. 201–205. DOI: 10.1109/ICoIAS.2018.8494053.

[46] Sinno Jialin Pan und Qiang Yang. »A Survey on Transfer Learning«. en. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Okt. 2010), S. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. (Besucht am 31. 03. 2020).

[47] Pascal Poupart. *CS885 Reinforcement Learning Lecture 11b: CS885 Reinforcement LearningLecture 11b:* English. Juni 2018. URL: https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring18/slides/cs885-lecture11b.pdf (besucht am 08. 04. 2020).

[48] Rasa. *Actions*. Library Catalog: rasa.com. März 2020. URL: https://rasa.com/docs/rasa/core/actions (besucht am 22. 06. 2020).

[49] Rasa. *Domains*. Library Catalog: rasa.com. Feb. 2020. URL: https://rasa.com/docs/rasa/core/domains (besucht am 03. 04. 2020).

[50] Rasa. *Policies*. Library Catalog: rasa.com. März 2020. URL: https://rasa.com/docs/rasa/core/policies (besucht am 03. 04. 2020).

[51] Rasa. *Rasa*. Git-Repository. Apr. 2020. URL: http://github.com/RasaHQ/rasa (besucht am 17. 04. 2020).

[52] Alan Ritter, Colin Cherry und William B Dolan. »Data-Driven Response Generation in Social Media«. en. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, S. 583–593.

[53] Thomas A. Runkler. *Data Mining*. de. Wiesbaden: Springer Fachmedien Wiesbaden, 2015. ISBN: 978-3-8348-1694-8. DOI: 10.1007/978-3-8348-2171-3. (Besucht am 24. 07. 2020).

[54] Jost Schatzmann und Steve Young. »The Hidden Agenda User Simulation Model«. en. In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.4 (Mai 2009), S. 733–747. DOI: 10.1109/TASL.2008.2012071. (Besucht am 04. 03. 2020).

[55] Jost Schatzmann u. a. »A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies«. en. In: *The Knowledge Engineering Review* 21.2 (Juni 2006), S. 97–126. DOI: 10.1017/S0269888906000944. (Besucht am 04. 03. 2020).

[56] Jost Schatzmann u. a. »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*. NAACL-Short '07. Rochester, New York: Association for Computational Linguistics, Apr. 2007, S. 149–152. (Besucht am 03. 03. 2020).

[57] Hannes Schulz u. a. »A Frame Tracking Model for Memory-Enhanced Dialogue Systems«. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 2017, S. 219–227.

[58] M. Schuster und K.K. Paliwal. »Bidirectional recurrent neural networks«. In: *IEEE Transactions on Signal Processing* 45.11 (Nov. 1997). Conference Name: IEEE Transactions on Signal Processing, S. 2673–2681. ISSN: 1941-0476. DOI: 10.1109/78.650093.

[59] Iulian V. Serban u. a. »Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models«. en. In: *Thirtieth AAAI Conference on Artificial Intelligence*. arXiv: 1507.04808. 2016. (Besucht am 05. 03. 2020).

[60] Iulian Vlad Serban u. a. »A Survey of Available Corpora for Building Data-Driven Dialogue Systems«. In: (März 2017). arXiv: 1512.05742. URL: http://arxiv.org/abs/1512.05742 (besucht am 03. 03. 2020).

[61] Mohit Sewak. *Deep Reinforcement Learning: Frontiers of Artificial Intelligence*. en. Singapore: Springer Singapore, 2019. ISBN: 978-981-13-8285-7. DOI: 10.1007/978-981-13-8285-7. (Besucht am 24. 07. 2020).

[62] David Silver u. a. »Mastering the game of Go with deep neural networks and tree search«. en. In: *Nature* 529.7587 (Jan. 2016). Number: 7587 Publisher: Nature Publishing Group, S. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://www.nature.com/articles/nature16961 (besucht am 31. 03. 2020).

[63] David Silver u. a. »Mastering the game of Go without human knowledge«. en. In: *Nature* 550.7676 (Okt. 2017). Number: 7676 Publisher: Nature Publishing Group, S. 354–359. DOI: 10.1038/nature24270. (Besucht am 31. 03. 2020).

[64] Abhishek Singh, Karthik Ramasubramanian und Shrey Shivam. »Introduction to Microsoft Bot, RASA, and Google Dialogflow«. en. In: *Building an Enterprise Chatbot: Work with Protected Enterprise Data Using Open Source Frameworks*. Hrsg. von Abhishek Singh, Karthik Ramasubramanian und Shrey Shivam. Berkeley, CA: Apress, 2019, S. 281–302. ISBN: 978-1-4842-5034-1. DOI: 10.1007/978-1-4842-5034-1_7. (Besucht am 09. 03. 2020).

[65] *The RavenClaw Dialog Management Architecture*. URL: https://www.cs.cmu.edu/~dbohus/ravenclaw-olympus/systems_overview.html (besucht am 03. 04. 2020).

[66] Blaise Thomson und Steve Young. »Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems«. en. In: *Computer Speech & Language* 24.4 (Okt. 2010), S. 562–588. DOI: 10.1016/j.csl.2009.07.003. (Besucht am 04. 03. 2020).

[67] Peter Tittmann. *Graphentheorie: Eine anwendungsorientierte Einführung*. de. 3. Aufl. München: Carl Hanser Verlag GmbH & Co. KG, März 2019. ISBN: 978-3-446-46052-2. DOI: 10.3139/9783446465039. (Besucht am 14. 05. 2020).

[68] David C Uthus und David W Aha. »The ubuntu chat corpus for multiparticipant chat analysis«. In: *2013 AAAI Spring Symposium Series*. 2013.

[69] Ashish Vaswani u. a. »Attention is All you Need«. en. In: *Advances in neural information processing systems*. Long Beach, CA, USA, 2017, S. 5998–6008.

[70] Vladimir Vlasov. *Fix bug in docs - set augmentation default to 20 as it is actually set to 20 in all other places by RolandJAAI - Pull Request #4222 - RasaHQ/rasa - GitHub*. Aug. 2019. URL: https://github.com/RasaHQ/rasa/pull/4222#issuecomment-523108795 (besucht am 07. 04. 2020).

[71] Vladimir Vlasov, Akela Drissner-Schmid und Alan Nichol. »Few-Shot Generalization Across Dialogue Tasks«. en. In: (Nov. 2018). arXiv: 1811.11707. (Besucht am 03. 03. 2020).

[72] Vladimir Vlasov, Johannes E. M. Mosig und Alan Nichol. »Dialogue Transformers«. In: (Nov. 2019). arXiv: 1910.00486. (Besucht am 03. 03. 2020).

[73] Tsung-Hsien Wen u. a. »Recurrent Neural Network Based Language Model Personalization by Social Network Crowdsourcing«. en. In: *INTERSPEECH*. 2013, S. 2703–2707.

[74] Jason D. Williams, Kavosh Asadi und Geoffrey Zweig. »Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning«. en. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. arXiv: 1702.03274. Apr. 2017, S. 665–677. (Besucht am 12. 03. 2020).

[75] Jason D. Williams und Steve Young. »Partially observable Markov decision processes for spoken dialog systems«. en. In: *Computer Speech & Language* 21.2 (Apr. 2007), S. 393–422. DOI: 10.1016/j.csl.2006.06.008. (Besucht am 05. 03. 2020).

[76] Steve Young. »Using POMDPS for dialog management«. In: *2006 IEEE Spoken Language Technology Workshop*. ISSN: null. Dez. 2006, S. 8–13. DOI: 10.1109/SLT.2006.326785.

[77] Steve Young u. a. »POMDP-Based Statistical Spoken Dialog Systems: A Review«. en. In: *Proceedings of the IEEE* 101.5 (Mai 2013), S. 1160–1179. DOI: 10.1109/JPROC.2012.2225812. (Besucht am 08. 04. 2020).

[78] Tiancheng Zhao und Maxine Eskenazi. »Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning«. en. In: *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (Sep. 2016). arXiv: 1606.02560. URL: http://arxiv.org/abs/1606.02560 (besucht am 05. 03. 2020).

[79] Tiancheng Zhao u. a. »Generative Encoder-Decoder Models for Task-Oriented Spoken Dialog Systems with Chatting Capability«. en. In: *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue* (Juni 2017). arXiv: 1706.08476, S. 27–36. URL: http://arxiv.org/abs/1706.08476 (besucht am 06. 04. 2020).

Abbildungsverzeichnis

Listings

Tabellenverzeichnis