Fachbereich 4: Informatik

DiaLex – Die Entstehung von Dialekten

vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz–Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation

vorgelegt von

Dipl.-Inform. Christian Klein

Vorsitzendes Mitglied des Promotionsausschusses

Prof. Dr. Rüdiger Grimm

Vorsitzendes Mitglied der Promotionskommission

Prof. Dr. Karin Harbusch

Berichterstattende Mitglieder der Promotionskommission

Prof. Dr. Klaus G. Troitzsch

Prof. Dr. Steffen Staab

Datum der wissenschaftlichen Aussprache

09.02.2011

Koblenz, im Mai 2011

Vorwort

An dieser Stelle möchte ich den Personen danken, die mir bei der Erstellung meiner Doktorarbeit geholfen haben.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Klaus G. Troitzsch, der mir stets neue Ideen und wertvolle Ratschläge gab und mir den Freiraum ließ, das Thema und die Fokussierung meiner Doktorarbeit selbst zu gestalten. Ich danke Prof. Dr. Steffen Staab als berichterstattendem Mitglied der Promotionskommission für die Anfertigung des Zweitgutachtens meiner Arbeit genau so wie Prof. Dr. Karin Harbusch als vorsitzendem Mitglied der Promotionskommission.

Auch danke ich meinen Eltern, meiner Schwester und meiner ganzen Familie für die langjährige sowohl moralische als auch finanzielle Unterstützung, ohne die die Realisierung dieser Arbeit nicht möglich gewesen wäre.

Als Letztes danke ich allen Korrekturlesern für die minutiöse Fehlersuche.

Koblenz, im Mai 2011 Christian Klein

Kurzfassung

Die vorliegende Dissertation untersucht die Entstehung von Dialekten innerhalb des in dieser Arbeit entworfenen Modells einer Multi-Agenten-Simulation, basierend auf neuronalen Netzen.

Zunächst werden die linguistischen Grundlagen detailliert dargestellt. Dazu gehört neben der Diskussion einiger Definitionen von Sprache ein Überblick über die sprachlichen Evolutionsstufen, eine Übersicht der Bestandteile menschlicher Hochsprachen mitsamt den bekannten Möglichkeiten ihrer Veränderung und eine Übersicht tierischer Sprachen mit kommunikativer Funktion und ihrer Dialektbildung.

Im Anschluss werden die Grundlagen der Informatik und Mathematik erläutert. Diese umfassen neben den Begriffen Modell und Simulation den Aufbau einer Multi-Agenten-Simulation und die Funktionsweise künstlicher neuronaler Netze. Aus den hier genannten Grundlagen wird daraufhin ausführlich das Simulationsmodell dieser Arbeit hergeleitet und beschrieben.

Die Ergebnisse vieler hundert verschiedener Simulationsdurchläufe werden im Anschluss erläutert. So werden die im Modell definierten Störfaktoren einzeln untersucht und in Werteintervalle mit unterschiedlichen Auswirkungen eingeteilt. Weiterhin werden vorhandene Wechselwirkungen der Faktoren aufgezeigt und der Prozess der Sprachwiedervereinigung nach einer vorherigen Dialektspaltung gezeigt.

Einem Fazit und Ausblick folgen als Anhänge Anforderungsdefinition, Entwurf, Architektur, eine detaillierte Erläuterung der Implementierung und ein englischsprachiges Benutzerhandbuch des Werkzeugs DiaLex. DiaLex ist ein komplett in Java konzipiertes und implementiertes Programm und bietet dem Benutzer die Möglichkeit, den Einfluss verschiedener Störfaktoren auf die Dialektbildung innerhalb einer gemeinsam genutzten Sprache einer oder mehrerer Agentengemeinschaften zu simulieren und zu analysieren.

Die Arbeit entstand im Zeitraum Oktober 2007 bis Juni 2010 an der Universität Koblenz-Landau.

Abstract

This dissertation investigates the emergence of dialects in a model of a multi-agent simulation based on neural networks that is developed within this thesis.

First the linguistic foundation of language is illustrated. Besides discussing some important definitions of language, this is achieved by giving a summary of the evolutionary steps in language evolution followed by an overview of the elements of human modern languages including the ways of language change. Subsequently some examples of socially learned animals' communicative behaviour and its formations of dialects are shown.

In the following the computational and mathematical basis are to be explained. Besides the expressions model and simulation, these comprehend the setup of multi-agent simulations and the functionality of artificial neural networks. Based on the abovementioned basics the model of this dissertation is derived and described in a detailed way.

Results drawn out of several hundreds of simulation runs are explicated thereafter. Each destructive factor defined in the model is examined separately and its domain is divided into intervals with different effects on the outcome of the simulation. Furthermore, existing interdependences between the single factors and the process of language merging after a prior dialect divide are shown.

Results and outlook are followed by specification, draft, architecture, a detailed illustration of the implementation and a user guide of the tool named DiaLex. DiaLex is a java based tool providing users the opportunity to simulate and analyse the influence of different destructive factors on dialect formation within a commonly used language of one or multiple communities of agents.

This thesis was written at University of Koblenz-Landau from October 2007 until June 2010.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

„The Sanskrit language, whatever be its antiquity, is of a wonderful structure; more perfect than the Greek, more copious than the Latin, and more exquisitely refined than either, yet bearing to both of them a stronger affinity, both in the roots of verbs and in the forms of grammar, than could possibly have been produced by accident; so strong, indeed, that no philologer could examine them all three, without believing them to have sprung from some common source, which, perhaps, no longer exists."

Mit diesen Worten, die er am 2. Februar 1876 an die Asiatische Gesellschaft in Kolkata richtete, begründete Sir William Jones, ein in Indien arbeitender britischer Richter, die Untersuchung der Ähnlichkeit von Sprachen und die daraus hergeleitete Einteilung in Sprachfamilien. Er stellte eine Verwandtschaft zwischen dem altindischen Sanskrit und den klassischen europäischen Sprachen Griechisch und Latein fest und publizierte seine Erkenntnisse in seinem Werk „*The Sanscrit Language*" [Jon86]. Im Jahre 1816 wies Franz Bopp die von Jones entdeckte Verwandtschaft in seinem Werk „*Über das Conjugationssystem der Sanskritsprache in Vergleichung mit jenem der griechischen, lateinischen, persischen und germanischen Sprache*" [Bop16] erstmals anhand wissenschaftlicher Methoden nach und etablierte damit formal die *historisch vergleichende Sprachwissenschaft*.

Innerhalb der letzten 200 Jahre ist es gelungen, nahezu alle bekannten Sprachen der Welt in Sprachfamilien einzuteilen, die jeweils aus einer bis zu weit über 1000 verwandten Sprachen bestehen. Aufgrund der Menge der zugehörigen Sprachen unterteilt man Sprachfamilien in Zweige, die sich wiederum in mehrstufige Untergruppen aufteilen, deren beinhaltete Sprachen mit steigender Diversifikation zueinander immer ähnlicher werden. Aktuell gelten weltweit 6909 Sprachen als dokumentiert. (vgl. [Lew09])

Nahezu zeitgleich wurde damit begonnen, die bekannten Sprachen in Dialekte zu unterteilen. So begannen die Gebrüder Grimm im Jahre 1838 mit dem „*Deutschen Wörterbuch*" [Gri54], in dem auch mundartliche Varianten des Deutschen zu finden waren. Ab 1875 erfasste Georg Wenker systematisch alle Dialekte des deutschen Sprachgebiets. Seine Arbeit wird bis heute fortgeführt und ist unter dem Namen „*Digitaler Wenker-Atlas*" [Wen75] frei im Internet verfügbar.

Die genaue Abgrenzung zwischen Dialekten und Sprachen ist teilweise umstritten bzw. nicht möglich. Als Beispiel ist die luxemburgische Sprache zu nennen, die

erwiesenermaßen zum moselfränkischen Dialektkontinuum gehört, daher als Dialekt des Hochdeutschen gesehen werden kann. Dennoch ist sie eine der Amtssprachen Luxemburgs und wird in standardisierter Form in den Medien genutzt.

Die Klassifizierung der menschlichen Sprache ist weit fortgeschritten und so genannte Ursprachen, aus denen sich die modernen Sprachen mit all ihren verschiedenen Dialekten entwickelten, wurden bereits rekonstruiert. Es existiert ebenfalls eine Vielzahl schriftlich belegter Sprachveränderungen der letzten 2000 Jahre. Als Beispiel hierfür ist die lateinische Sprache zu nennen, die, mittlerweile ausgestorben, die Grundlage aller romanischen Sprachen bildet. Dennoch ist weitgehend ungeklärt, unter welchen Umständen sich eine Sprache in mehrere verschiedene Nachfolgesprachen aufspaltet und welche Bedingungen diese Entwicklung begünstigen bzw. verlangsamen oder gar eindämmen. Folglich existieren bisher kaum computergestützte Simulationsergebnisse hierzu.

Ein weiterer Effekt in der Sprachentwicklung ist die Entstehung von Mischsprachen. Gerade in Grenzgebieten mit häufig unklarem territorialem Grenzverlauf von Staaten mit miteinander verwandten Sprachen bildet sich häufig eine Mischform der dort gesprochenen Sprachen. Dies ist beispielsweise beim Spanischen und Portugiesischen zu beobachten. So ist die galicische Sprache, die in der autonomen Gemeinschaft Galicien, einer Region Spaniens, gesprochen wird, eine Mischform aus Portugiesisch und Spanisch. Ein sehr ähnlicher Effekt ist auch in Uruguay zu beobachten, wo sich eine als Portuñol bezeichnete Mischsprache der beiden Sprachen gebildet hat. Dieses Phänomen ist noch nicht anhand computergestützter Simulationen nachgebildet worden.

Die Gründe für die Veränderung der menschlichen Sprache sind weitgehend bekannt. Jakob Hornemann Bredsdorff listete bereits im Jahre 1821 in seinem Werk „*Über die Ursachen der Sprachveränderung*“ [Bre21] einige für die Evolution von Sprachen relevante Aspekte auf. Er unterscheidet dabei zwischen Ursachen mit und Ursachen ohne den Einfluss fremder Nationen oder Sprachen. Die beiden genannten Hauptgründe Bredsdorffs, nämlich ein fehlerhaftes Hören und Verstehen, und ein unzuverlässiges Gedächtnis werden auch heute noch als die Hauptauslöser des sprachlichen Wandels gesehen. Die fehlerhafte Aussprache kann dabei mehrere Gründe haben. Einerseits besteht die Möglichkeit, dass der Sprecher nicht in der Lage ist, die Lautabfolge wie gefordert wiederzugeben, sei es durch die Begrenztheit seiner Organe und ihrer lautmalerischen Fähigkeiten, oder aber durch das Abrufen einer fehlerbehafteten Wiedergabeform aus seinem Gedächtnis. Andererseits kann der Sprecher auch aus Gründen der Ressourceneffizienz in Bezug auf Zeit und Energie eine vereinfachte Ausspracheform wählen. Dies beschrieb Helmut Lüdtke im Jahre 1980 in seinem Werk „*Sprachwandel als universales Phänomen*“ [Lüd80]. Das fehlerhafte Hören kann ebenfalls mehrere Gründe haben. So kann aufgrund der Umgebung ein „rauschfreies“ Verstehen nicht möglich sein, die Kapazität der Organe des Hörers

kann begrenzt sein oder aber es ist dem Hörer nicht möglich, das Gehörte fehlerfrei in seinem Gedächtnis zu speichern.

Es ist bereits gezeigt worden, dass in Multi-Agenten-Simulationen Dialekte innerhalb einer Population von Agenten auftreten können. Hierzu zählen die Arbeiten von Livingstone und Fyfe, *„A Computational Model of Language-Physiology Coevolution"*, [LF98] und Hutchins und Hazlehurst, *„How to invent a lexicon: the development of shared symbols in interaction*", [HH95]. Den bereits genannten Simulationsmodellen ist gemein, dass sie relativ simpel gehalten sind und sich daher stets ein Störfaktor als Grund der Sprachveränderung an ihnen untersuchen lässt, jedoch sind Vergleiche der Auswirkungen und Wechselwirkungen der verschiedenen Faktoren schwer möglich.

Wünschenswert wäre folglich ein Model, das alle hier genannten Effekte einer Sprache nachbilden kann. Ein Modell, das in einem ersten Schritt die Entstehung einer oder mehrerer parallel existierender Sprachen simulieren kann, die von entsprechend einer oder mehreren Agentengruppen gesprochen werden. Diese Sprachen sollen jedoch nicht starr existieren, sondern wie natürliche Sprachen „leben" und sich permanent verändern. Diese Veränderungen sollen einerseits innerhalb einer einzigen Sprache geschehen, als auch andererseits durch den Einfluss fremder Sprachen ausgelöst werden können. Desweiteren sollen einerseits Dialektbildungen bei räumlich separierten Agentengruppen, die eine einzige Sprache sprechen, beobachtbar gemacht werden können, die bei fortwährender Trennung in eine Sprachspaltung übergehen. Andererseits sollen auch Sprachvereinigungen wie die Entstehung von Mischsprachen simuliert werden können.

Aus diesem Grunde wird das Simulationsmodell dieser Arbeit mit einer Reihe von Störfaktoren ausgestattet, die es der Agentengemeinschaft nicht erlauben, einen statischen Wortschatz auszubilden. Das Ziel dieser Arbeit soll es folglich sein, die folgenden Fragen zu klären:

- Wie stark wirken sich die einzelnen Störfaktoren auf die von den Agenten genutzte Sprache aus? Wie stark verändern sie sie?

- Innerhalb welches Werteintervalls der jeweiligen Stellgrößen zerfällt eine Sprache in Dialekte? Was geschieht bei anderen Werten der Variablen?

- Welche Wechselwirkungen besitzen die einzelnen Faktoren der Sprachveränderung untereinander?

Dazu werden als Fundament dieser Arbeit in Kapitel 2 zuerst die sprachwissenschaftlichen Grundlagen detailliert dargestellt. Zu Beginn werden einige Definitionen von Sprache bekannter Linguisten in Bezug auf diese Arbeit diskutiert. Die daraus entstandene Fragestellung, ob Sprache ein Naturphänomen

oder ein Kulturphänomen ist, wird im Anschluss erörtert. Im Folgenden werden die sprachlichen Evolutionsstufen gezeigt, beginnend bei primitivsten Lautäußerungen, bis hin zu den menschlichen Hochsprachen. Das nächste Unterkapitel befasst sich mit den menschlichen Sprachen. Es werden ihre Beziehungen untereinander und ihre Bestandteile genauer dargestellt. Ein besonderer Fokus wird auf die Sprachveränderungen innerhalb der Lautlehre gelegt. Zum Abschluss dieses Kapitels werden einige Tiersprachen mit einer kommunikativen Funktion gezeigt, die ebenfalls eine Dialektausprägung vorweisen.

Kapitel 3 steht im Zeichen der Entwicklung des Simulationsmodells. Zuerst werden die Begriffe Simulation und Modell im Sinne der Informatik erläutert, bevor der für diese Arbeit genutzte Teilbereich der Multi-Agenten-Simulation detaillierter dargestellt wird. Im Anschluss daran werden der Aufbau und die Funktionsweise künstlicher neuronaler Netze erklärt, in denen das Wissen der Agenten in dieser Arbeit kodiert wird. Nachdem darauf die bereits vorhandenen Modelle zur Simulation von Sprache erläutert und ihre Stärken und Schwächen diskutiert werden, schließt das Kapitel mit der detaillierten Beschreibung des Simulationsmodells dieser Arbeit ab. In dieser wird, basierend auf den linguistischen Grundlagen der realen Welt und biologischen Fähigkeiten des Menschen, ein Modell für eine Multi-Agenten-Simulation hergeleitet, indem menschliche Individuen und ihre räumliche Umgebung in einer Art nachgebildet werden, die es ermöglicht, den Einfluss verschiedener Störfaktoren auf die lexikalische Sprachveränderung zu untersuchen.

Im vierten Kapitel werden einige Simulationen dargestellt und detailliert ausgewertet. So werden sämtliche im Modell definierten Faktoren, die eine Sprachspaltung bedingen können, einzeln untersucht und in Werteintervalle mit unterschiedlichen Auswirkungen eingeteilt. Weiterhin werden vorhandene Wechselwirkungen gezeigt und ebenfalls eine Sprachwiedervereinigung nach einer vorherigen Spaltung erläutert.

Das fünfte Kapitel liefert neben einem Fazit einen Ausblick auf weitere offene Fragen des Themengebiets, zeigt mögliche zukünftige Schritte in der Forschung auf und befasst sich mit weiteren Anwendungsmöglichkeiten des Modells dieser Arbeit.

In einem ausführlichen Anhang werden die Anforderungen an das Simulationsmodell und wichtige Teile seiner Implementierung beschrieben.

Anhang A stellt den Übergang vom Simulationsmodell hin zur Implementierung dar. Die Anforderungen an das Simulationstool werden im ersten Teil definiert. Die sich daraus ergebende Architektur des Programms und seine Komponenten werden im Anschluss gezeigt und schließlich als Entwurf weiter verfeinert und detailliert erläutert.

Anhang B beschreibt die daraus resultierende Implementierung des Simulationstools *DiaLex*, aufgeteilt in die Bereiche Simulation und grafische Benutzeroberfläche mit ihren jeweils wichtigsten Klassen. *DiaLex* ist komplett in der Programmiersprache Java geschrieben und somit auf einer Vielzahl von Systemen lauffähig.

Anhang C besteht aus einem in englischer Sprache geschriebenen Handbuch des Simulationstools *DiaLex*, das einem größeren Benutzerkreis den Einsatz dieses Programms ermöglichen soll.

2. Linguistische Grundlagen

In diesem Kapitel werden die linguistischen Grundlagen dieser Arbeit erläutert. Diese umfassen neben den Sprachdefinitionen einiger bekannter Linguisten eine Darstellung des Konflikts Natur- versus Kulturphänomen, eine Erläuterung der sprachlichen Evolutionsstufen, eine Übersicht sprachlicher Gemeinsamkeiten und Beziehungen, eine detaillierte Auflistung der Bestandteile natürlicher Sprachen und ihre zugehörigen Arten der Veränderungen innerhalb der Lautlehre und schließlich eine Darstellung von Tiersprachen mit kommunikativer Funktion, in denen ebenfalls eine Dialektbildung nachweisbar ist.

2.1 Definitionen von Sprache

Das Wort „Sprache“ hat in der deutschen Sprache, im Gegensatz zu vielen anderen europäischen Sprachen zwei Bedeutungen. Zum einen bezeichnet Sprache eine bestimmte Einzelsprache wie beispielsweise die deutsche Sprache oder die englische Sprache. Diese Bedeutung des Worts „Sprache“ kann ebenfalls im Plural gebraucht werden, so z.B. im Ausdruck „die germanischen Sprachen“. Die zweite Bedeutung des Worts ist die in dieser Arbeit wichtigere. Das Wort „Sprache“ steht, wenn es ausschließlich im Singular genutzt werden kann, für eine aus Symbolen bestehende Kommunikation im weiteren Sinne. Die Fähigkeit, eine „Sprache“ nutzen zu können bedeutet also, sich mit anderen Individuen verständigen zu können. Es existiert eine Vielzahl von Sprachdefinitionen. An dieser Stelle werden fünf häufig zitierte Definitionen renommierter Linguisten wiedergegeben, die in [Lyo81] diskutierten Kritikpunkte dieser Definitionen dargestellt und derart unter einem informatischen Gesichtspunkt erweitert, dass sie im Rahmen dieser Arbeit genutzt werden können.

Sapir definiert Sprache in seinem Werk [Sap21] wie folgt: „*Sprache ist eine ausschließlich dem Menschen eigene, nicht im Instinkt wurzelnde Methode zur Übermittlung von Gedanken, Gefühlen und Wünschen mittels eines Systems von frei geschaffenen Symbolen.*“
Diese älteste hier erläuterte Definition hat jedoch einige Schwächen. Zum einen decken die Begriffe „Gedanke“, „Gefühl“ und „Wunsch“ bei weitem nicht alles ab, was durch Sprache übermittelt werden kann, zum anderen ist die Einschränkung, dass eine Sprache nur zwischen Menschen existieren kann, in der heutigen Zeit nicht mehr gültig, da sowohl weitere natürliche Lebensformen als auch künstliche Intelligenzen Sprachen ausgebildet haben bzw. ausbilden können. Die positiven Aspekte seiner Definition sind jedoch erwähnenswert. Er verzichtet im Gegensatz zu vielen anderen Linguisten auf den Ausdruck „Lautsymbol“ und

nutzt die allgemeinere Form „Symbol“. Somit werden nicht-akustische Sprachformen durch seine Definition eingeschlossen, beispielsweise die Schriftsprache oder die Gebärdensprache.

In [BT42] schreiben Bloch und Trager: *„Eine Sprache ist ein System willkürlicher Lautsymbole, mit deren Hilfe eine soziale Gruppe gemeinsam handelt.“* Es fällt auf, dass diese Definition den Schwerpunkt auf die soziale Komponente der Sprache legt und weitaus weniger auf die kommunikative Funktion. Somit wird nicht geklärt, was die Lautsymbole beinhalten und vor allem, wie sie sich auf das gemeinsame Handeln einer Gruppe auswirken. Ein weiterer Schwachpunkt ist die Verwendung des Ausdrucks „Lautsymbol“. Sprache wird demnach auf die gesprochene Sprache beschränkt. Eine Stärke ihrer Definition ist die Erwähnung der Willkürlichkeit der Zuordnung vom Bezeichnenden (Symbol, Wort) zu Bezeichnetem (reales Objekt). Sie ist nicht zu verwechseln mit der Willkür, sondern sagt aus, dass die Zuordnung Objekt-Symbol rein auf menschlicher Konvention basiert und keinerlei natürlichen Gesetzmäßigkeiten folgt.

In seiner Arbeit [Hal68] vertritt Hall die Ansicht, Sprache sei *„die Institution, mit deren Hilfe Menschen miteinander kommunizieren und unter Verwendung gewohnheitsmäßig benutzter, oral-auditiver, willkürlicher Symbole in Interaktion treten*“. Offensichtlich beschränkt sich auch Hall darauf, nur die menschliche Kommunikation als Sprache zu deklarieren. Ein weiterer Schwachpunkt ist die Benutzung des Terminus „oral-auditiv“, der auf eine rein lautliche Kommunikation hindeutet. Die Einführung des Ausdrucks „Interaktion“ ist jedoch ein Fortschritt verglichen mit den vorherigen Definitionen, da er umfassender ist als die noch von Bloch und Trager benutze Bezeichnung Kooperation bzw. das gemeinsame Handeln. Sprache muss ja nicht stets im kooperativen Sinne verstanden werden, eine Kommunikation ohne gegenseitige Interaktion ist jedoch nicht möglich.

Robins gibt in seinem Werk [Rob79] keine formale Definition von Sprache, sondern weist darauf hin, dass solche Definitionen *„dazu neigen, trivial und uninformativ zu sein, wenn sie nicht auf einer allgemeinen Sprachtheorie und einer linguistischen Analyse basieren*“. Er führt jedoch einige Tatsachen an, die seiner Meinung nach in einer Sprachtheorie Berücksichtigung finden müssen, so seien Sprachen *„Symbolsysteme, die fast auf reiner oder willkürlicher Konvention beruhen“*. Im Allgemeinen bergen seine Deutungen im Vergleich zur vorherigen Definition keine Neuerungen, er verdeutlicht noch einmal die Willkürlichkeit der sprachlichen Namenskonvention.

Eine vollkommen andere Definition lieferte Chomsky in [Cho57]. Er äußerte sich zu Sprachen wie folgt: *„Von jetzt an werde ich unter einer Sprache eine (endliche oder unendliche) Menge von Sätzen verstehen, jeder endlich in seiner Länge und konstruiert aus einer endlichen Menge von Elementen*.“ Mit dieser Definition beabsichtigte er weit mehr abzudecken, als die natürlichen Sprachen in ihrer

gesprochenen und geschriebenen Form. Er definierte Sprachen relativ simpel über folgende zwei Eigenschaften:

1. Jede Sprache besitzt ein endliches System von Lauten (Buchstaben).
2. Ein Satz kann als eine endliche Folge der Laute aus 1. dargestellt werden.

Diese Definition unterscheidet sich stark von den vorherigen vier genannten Definitionen. Sie sagt nichts über eine kommunikative Funktion oder gar über ein gemeinsames soziales Handeln aus. Diese Definition hat den Zweck, *„auf die rein strukturellen Eigenschaften von Sprachen aufmerksam zu machen und anzudeuten, dass diese Eigenschaften in mathematisch präziser Weise untersucht werden können.“* ([Lyo81], S. 17)

Chomskys Definition liefert gerade für nicht natürliche Sprachen einen bedeutenden Beitrag. Während sich die vorangegangenen Definitionen nahezu ausschließlich auf menschliche Kommunikation als Sprache beriefen, erweitert Chomsky diese auch auf einen Austausch zwischen Tieren und künstlichen Intelligenzen, beispielsweise die der Agenten in einer Multi-Agenten-Simulation, die Kommunikation zwischen mehreren Maschinen oder die Sprachen im Tierreich. Dadurch liefert er die für diese Arbeit brauchbarste Auslegung von Sprache. Eine genauere Erläuterung des in dieser Arbeit verwendeten Modells, seiner Sprache und die verwendeten Analogien zu seiner Definition befinden sich in Unterkapitel 3.5.

2.2 Konflikt Naturphänomen – Kulturphänomen

Obwohl im vorangegangenen Unterkapitel einige Definitionen von Sprache erläutert und diskutiert wurden, bleibt immer noch die Frage nach der Herkunft und der Entstehung von Sprache zu klären. Vereinfacht gesagt stehen sich dort zwei Ansichten, gar zwei Weltanschauungen gegenüber, die gegensätzlicher nicht sein könnten. Keller[1] spricht gar von einer von „*Dichotomie geprägten Kultur*“ ([Kel90], S.59) deren Ansichten aber dennoch kombinierbar zu sein scheinen und daher sehr wohl beide ihre Richtigkeit haben.

2.2.1 Soziobiologische Erklärungen – Sprache als Naturphänomen

Der erste Ansatz zur Herkunft von Sprache wurde von Hurford in [Hur89] geprägt, der dort den Begriff „soziobiologisch“ eingeführt hat. Seiner Ansicht nach sind es die genetischen Eigenschaften und die natürliche Selektion, die zu einer

[1] Rudi Keller (*1942), deutscher Linguist und Professor für Germanistische Sprachwissenschaft an der Universität Düsseldorf.

Ausprägung und permanenten Verbesserung der Sprachfähigkeit geführt haben. Linguisten führen dazu stets zwei Faktoren an, die diese Herkunft belegen sollen.

- Viele unterschiedliche menschliche Sprachen tragen strukturelle Ähnlichkeiten in sich, als ob sie genetisch miteinander verwandt wären. (vgl. [Gre63], S. 255: *„cut from the same pattern*“)

- Kinder erlernen Sprachen in sämtlichen verschiedensten Kulturen der Welt mit minimalem Aufwand erstaunlich schnell.

Gerade der letzte Punkt deutet auf eine instinktive, hoch spezialisierte, genetische Veranlagung hin, Sprachen erlernen zu können. Chomsky nennt sie *„specific language faculty*“ und beschreibt sie detailliert in [Cho72]. Mit der Ansicht, dass die Fähigkeit des Spracherwerbs genetisch kodiert und damit angeboren ist und genau wie andere biologischen Merkmale der Evolutionslehre unterliegt, ergibt sich folgendes Szenario, das Steels in [Ste05] wie folgt zusammengefasst hat:

1. Individuen, die eine größere kommunikative Fähigkeit besitzen, haben einen selektiven Vorteil gegenüber anderen Individuen, die nicht über derart ausgeprägte Fähigkeiten verfügen und besitzen folglich eine höhere Fortpflanzungswahrscheinlichkeit.

2. Eine angeborene Kommunikationsstrategie setzt sich aufgrund ihrer Überlegenheit gegenüber anderen angeborenen Strategien durch, da ihre Anhänger einen selektiven Vorteil gegenüber anderen Individuen haben. Damit erhöht sich der relative Anteil der überlegenen Kommunikationsstrategie von Generation zu Generation.

3. Aufgrund der vorangegangenen Punkte setzt sich die überlegene Kommunikationsstrategie auf lange Sicht durch und verdrängt alle anderen Strategien. Dadurch ergibt sich ein einheitliches Kommunikationsverfahren.

Hurford benutzte diese Denkweise zur Klärung der besten Strategie, mit der Agenten in Simulationen unterschiedliche Objekte benennen können. Es stellte sich heraus, dass eine von Ferdinand de Saussure[2] vorgeschlagene Strategie, bei der die Assoziationen zwischen Namen und Objekten bidirektional angeordnet sind, optimal sind. Daher könnte es sein, dass sich diese Strategie auch biologisch durch natürliche Selektion durchgesetzt hat.

Auch weitergehende Aspekte von Sprache wie eine Kompositionalität könnten derart eine Vormachtstellung erhalten haben. Möglicherweise gab es einen Wettbewerb um die beste Kodierungsstrategie zum Ausdruck komplexerer

[2] Ferdinand de Saussure (1857-1913), Schweizer Sprachwissenschaftler, der den Strukturalismus prägte.

Sachverhalte, der evolutionär gelöst wurde. Individuen mit einer besseren Strategie wären dadurch in der Lage gewesen, ein mächtigeres, stabileres Kommunikationssystem zu kreieren, definitiv ein evolutionärer Vorteil.

Dies würde bedeuten, dass ein Mensch bereits bei der Geburt ein angeborenes Verständnis für eine Universalgrammatik in sich trägt. Diese Ansicht vertritt Chomsky, der sein „*Language Acquisition Device Model*“ in [Cho65] beschreibt. Seiner Ansicht nach hat ein Kind vier angeborene Prädispositionen, die es zum Spracherwerb nutzt.

Die Prädispositionen sind:

- **Formale Universalien**
  Die formalen Universalien betreffen den Charakter von grammatischen Regeln und ihre Verknüpfungen. Sie beziehen sich auf linguistische Konzepte wie Tiefenstruktur, Oberflächenstruktur und Transformationsregeln.

- **Substanzielle Universalien**
  Die substantiellen Universalien helfen dabei, die Substanz einer Sprache zu erkennen, indem Laute, die nicht zur übermittelten Sprache gehören (Rauschen), herausgefiltert werden können.

- **Hypothesenbildungsverfahren**
  Aufgrund der beiden vorangegangenen Punkte bildet das Kind Hypothesen über die Struktur der Sprache. Das Kind stellt dabei selbstständig Hypothesen über die Sprachstruktur auf, die bestätigt oder widerlegt werden können. Mit zunehmender Sprachkenntnis verbessert das Kind seine Annahmen, bis es die Sprache erlernt hat.

- **Hypothesenbewertungsverfahren**
  Da das Kind in der Lage ist, eine gehörte Sprache zu abstrahieren und Hypothesen über deren Struktur aufstellen kann, muss davon ausgegangen werden, dass das Kind einen Mechanismus besitzt, mit dem es in der Lage ist zu entscheiden, welche Regelmengen die der Sprache zugehörigen Grammatik darstellen.

2.2.2 Soziokulturelle Erklärungen – Sprache als Kulturphänomen

Ein zweiter Ansatz zur Entstehung von Sprache basiert auf soziokulturellen Gründen. Im Gegensatz zur soziobiologischen Erklärung laufen dabei folgende Schritte bei Individuen einer Population ab: (vgl. [Ste05])

1. Jedes Individuum besitzt mehrere Strategien zur Benutzung des Kommunikationssystems.

2. Einige Strategien sind erfolgreicher als andere und beeinflussen damit die Ausdrucksstärke einer Sprache, den Kommunikationserfolg der Individuen und/oder den Aufwand den sie benötigen.

3. Die Individuen versuchen ihre Ausdrucksstärke zu maximieren, den kommunikativen Erfolg zu optimieren und ihren Aufwand zu minimieren. Daher wählen sie die am breitesten genutzte Strategie aus. Aus diesem Grund setzt sich eine Kommunikationsstrategie in einer kulturellen Zeitspanne gegen alle ihre Rivalen durch und wird die Strategie, mit der Kommunikationssysteme aufgebaut und betrieben werden.

Es gibt einige offensichtliche Unterschiede im Vergleich zum vorherigen Ansatz. In erster Linie unterscheiden sich beide in Bezug auf die Zeitspanne in der Veränderungen stattfinden. Während im Rahmen der soziobiologischen Erklärung von *evolutionären Zeitspannen* die Rede ist, also nach allgemeinem Verständnis von Zeiträumen zwischen vielen tausend Jahren bis hin zu Jahrmillionen, handelt es sich hier um eine *kulturelle Zeitspanne*, einen kürzeren Zeitabschnitt, der sich zwischen mehreren Wochen bis hin zu einigen Jahren abspielt.

Dies ist leicht nachvollziehbar, da eine genetische Selektion erst über eine erfolgreiche Reproduktion ablaufen kann und sich Verbesserungen im Erbgut erst nach sehr vielen Nachfolgegenerationen abzeichnen. Hingegen bietet die soziokulturelle Evolution sehr viel kürzere Verbesserungsmechanismen wie einer direkten Rückmeldung des Gegenübers über den Erfolg der Kommunikation. Hinzu kommt, dass das Erfinden neuer und Abändern bereits bekannter Kommunikationsstrategien im soziokulturellen Kontext nicht auf genetischer Mutation oder auf Rekombination basiert, sondern auf eigener Kreativität bzw. der Nachahmung anderer, wenn auch in, ob gewollt oder ungewollt, leicht veränderter Weise.

Keller bezeichnet den Unterschied der natürlich-biologischen Kreativität auf der einen und der individuell-kulturellen Kreativität auf der anderen Seite, den Konflikt „*Instinkt versus Vernunft*“ (vgl. [Kel90]). Mit Hilfe der soziokulturellen Erklärung kann daher die Entstehung einzelner Idiolekte oder aber regionaler Dialekte erklärt werden, in denen sich eine bestimmte Kommunikationsstrategie gegenüber anderen auf lokal begrenztem Raum durchgesetzt hat.

In Simulationen stellte sich, wie auch schon beim soziobiologischen Ansatz heraus, dass eine bidirektionale assoziative Zuordnung zwischen einem realen Objekt und der entsprechenden Lautäußerung der erfolgreichste Ansatz war, in dem sich Agenten auf ein gemeinsam genutztes Lexikon einigen konnten. Dies geschah im Rahmen einer Selbstorganisation der Agenten, in der die

Gewichtsveränderung und Optimierung der Kantengewichte eines künstlichen neuronalen Netzes als Informationsspeicher dienten. Im Rahmen dieser Arbeit ist hierbei insbesondere das Modell von Hutchins und Hazlehurst zu nennen (vgl. [HH95]). Positive Verstärkung nach einer erfolgreichen und Abschwächung nach nicht erfolgreicher Kommunikation führen nach einer gewissen Zeit zur Emergenz eines gemeinsam genutzten Lexikons. Ähnliche Methoden dienen den Agenten auch bei der Einigung auf eine gemeinsam genutzte Syntax in dem von Kirby und Hurford in [KH02] beschriebenen Modell.

2.2.3 Ein gemischter Ansatz

Der dritte und damit letzte Ansatz zur Entstehung von Sprache ist der gemischt biologisch-soziokulturelle Ansatz, der beide vorherigen Theorien miteinander verbindet. Dies ist auf den ersten Eindruck überraschend, da sich beide Ansichten in wesentlichen Punkten unterscheiden. Betrachtet man beispielsweise die Existenz einer bestimmten Syntax und die damit verbundene Verdrängung anderer Syntaxen, so argumentieren Anhänger des soziobiologischen Ansatzes, dass dies durch natürliche Selektion und der dadurch bedingten Verdrängung anderer Formen erklärt werden kann, während der soziokulturelle Ansatz dies durch die Überlegenheit einer Syntax, sei es durch größere Ressourceneffizienz, Ausdrucksstärke oder Fehlerresistenz, begründet.

Betrachtet man jedoch die stark unterschiedlichen Zeitspannen, mit denen Änderungen in beiden Varianten greifen, so scheint ein gemischter Ansatz realistisch. Es liegt im Bereich des Möglichen, dass die Fähigkeit Sprache im Sinne der Fähigkeit des Kommunizierens durchaus genetisch kodiert ist. Dies wäre eine Erklärung dafür, warum Menschen auf allen Erdteilen, auch in den entlegensten Regionen, stets eine Sprache benutzten. Es würde die Tatsache erklären, warum viele Linguisten Sprache als eine angeborene Fähigkeit, einen Instinkt deklarieren, so wie es beispielsweise Chomsky in [Cho72] beschrieben hat. Alle Menschen wären dadurch in der Lage, die Strukturen von Sprache zu verstehen, genau so wie sie in der Lage sind Lautäußerungen anderer zu hören und sich phonologisch bemerkbar zu machen.

Daher ist der Einsatz genetisch kodierter Information in Simulationen über die Entstehung der Sprachfähigkeit sinnvoll. Betrachtet man jedoch sprachliche Varietäten der „evolutionären Neuzeit“, also Prozesse der letzten 10000 Jahre, sei es beispielsweise die Dialektbildung, Verwandtschaftsbeziehungen innerhalb von Sprachen, die Emergenz von Soziolekten, die Entstehung von Pidginsprachen oder anderes, so scheinen genetische Prozesse weniger sinnvoll zu sein, da sie zeitlich gar nicht in der Lage sind, Veränderungen schnell genug abzubilden. Daher ist in diesem Falle die Verwendung von soziokulturellen Modellen zu bevorzugen. Dies bedeutet aber nicht, dass die Grundlagen des Sprachverständnisses in derartigen Simulationen nicht soziobiologisch seien, viel mehr ist der Einfluss biologischer Veränderungen auf dort simulierte Prozesse so gering, dass sie zu vernachlässigen

sind und aus Gründen der Einfachheit in einem Modell keine Beachtung finden müssen. Biologische Prozesse bilden eher eine Basis, die Fähigkeit soziokulturelle Prozesse zu interpretieren. Daher sind beide Ansätze nicht unabhängig voneinander zu sehen, sondern stets miteinander, aber dennoch jeweils mit anderen Schwerpunkten.

2.3 Sprachliche Evolutionsstufen

2.3.1 Unterscheidung in lexikalische und grammatische Bestandteile

Die einfachste und älteste Unterscheidung der Bestandteile von Sprache ist die Unterteilung in lexikalische und grammatische Elemente. Diese Unterscheidung hält bis heute an. Derek Bickerton[3] unterscheidet in seinem Werk [Bic90] ebenfalls nur zwei Stufen, eine rein lexikalische, die er „*Protolanguage*“ nennt und eine additionale grammatikalische Stufe, die, zusammen mit der lexikalischen, eine moderne Sprache ausmacht.

Als Lexikon bezeichnet man dabei den Wortschatz einer Sprache oder eines Sprechers zu einem bestimmten Zeitpunkt. Es ist dabei wichtig zu betonen, dass eine rein lexikonbasierte Sprache keinerlei Satzstrukturen beherbergt, so ist beispielsweise die Reihenfolge der Symbolabfolge vollkommen irrelevant. Durch das Hinzufügen von grammatikalischen Strukturen erhält eine rein lexikalische Sprache die Mächtigkeit einer modernen Sprache. Sie enthält damit nicht nur eine semantische Bedeutung der Symbole des Lexikons, diese können nun auch mit Hilfe einer Syntax zu größeren, komplexeren Strukturen zusammengefasst werden, womit sich der Informationsgehalt einer Nachricht erheblich steigern lässt.

In den folgenden beiden Abschnitten werden zwei detaillierte Gliederungen der Entwicklung und damit auch der Komplexität des Sprachvermögens vorgestellt. Mit ihrer Hilfe wird das in dieser Arbeit verwendete Simulationsmodell (vgl. Unterkapitel 3.5) bewertet und eingeordnet.

2.3.2 Mögliche Schritte in der Entwicklung des Sprachvermögens

Ray Jackendoff[4] unterteilt in [Jac99] die Entwicklung des Sprachvermögens in elf Schritte, die im Folgenden erläutert werden. Kommunikationssysteme ohne Syntax reichen dabei bis zu einer Zwischenstufe, die er „*Protosprache*“ nennt, weiter

[3] Derek Bickerton (*1926), emeritierter Professor für Linguistik der University of Hawaii, USA.
[4] Ray Jackendoff (*1945), von 1971-2005 Professor der Linguistik an der Brandeis University, seit 2005 Professor für Philosophie an der Tufts University, beide Massachusetts, USA.

entwickelte Systeme beinhalten stets syntaktische Elemente. Da die Schritte nicht nur linear angeordnet sind, werden ihre Abhängigkeiten in Abb. 1 illustriert.

Schritt I: Die Benutzung von situationsunabhängigen Symbolen

Als ersten Schritt in der Entwicklung von Kommunikation sieht Jackendoff die nicht-situationsspezifische Benutzung symbolischer Vokalisationen. Es existieren also Lautäußerungen, die zu jeder Zeit die gleiche Bedeutung tragen und stets benutzt und auch verstanden werden können. Ein derartiges Kommunikationssystem bildet ein Lexikon aus, das die Zuordnung der Vokalisationen zu ihrer Bedeutung beinhaltet.

Die nächsten Verwandten des Menschen, die Menschenaffen, kennen diese Art des Nutzens von Symbolen nicht. Es existieren zwar Lautäußerungen, die situationsbedingt ausgestoßen werden, beispielsweise beim Auffinden von Nahrung, jedoch wird diese Lautäußerung nicht gebraucht, um andere Aussagen bezüglich Nahrung zu treffen, wie beispielsweise das Auffordern zur Nahrungssuche. (vgl. [Hau96])

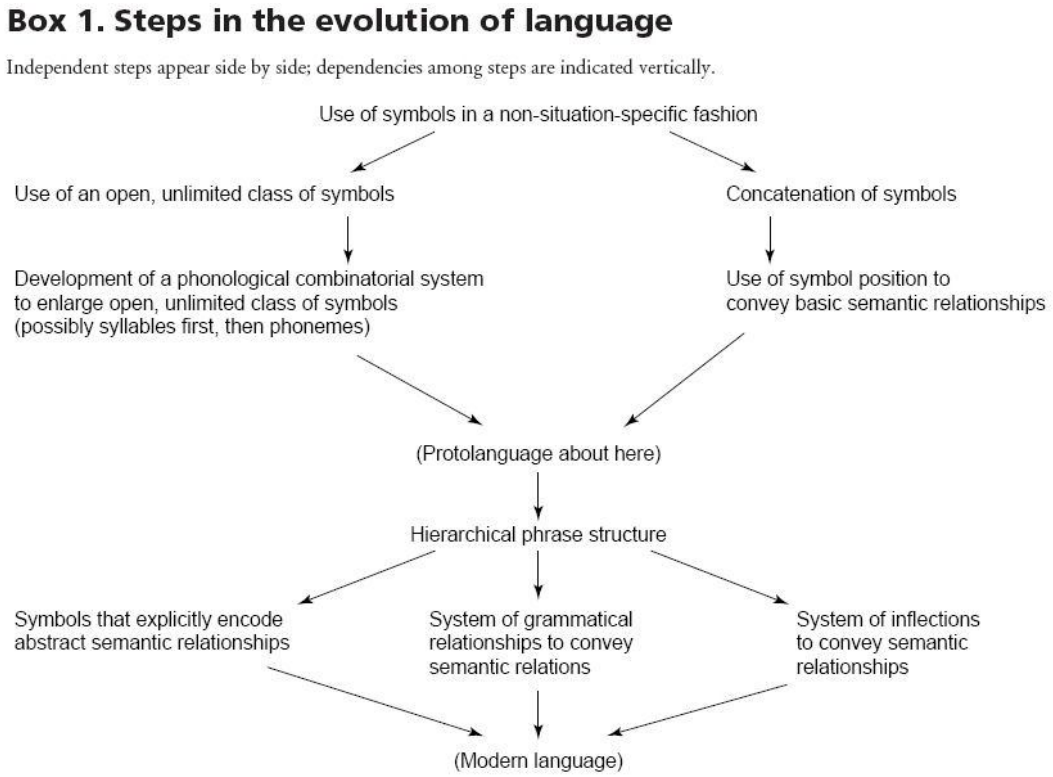


Abb. 1 Schritte in der Sprachentwicklung
Quelle: [Jac99]

Schritt II a: Die Existenz einer unbegrenzten Klasse von Symbolen

Eine wichtige Erweiterung auf dem Weg zu einer Sprache ist die Fähigkeit der Sprecher, einen nahezu unbegrenzt großen Wortschatz benutzen zu können, sprich, sie müssen in der Lage sein, eine jede Äußerung in ihrem Gedächtnis abzuspeichern und sie auch wieder in kürzester Zeit abrufen zu können. Dies stellt hohe Anforderungen an die kognitiven Fähigkeiten des Sprechers.

Menschliche Erwachsene besitzen einen Wortschatz von mehreren zehntausend Wörtern, während ausgewachsene Menschenaffen durch intensives Training einen Wortschatz von mehreren hundert Symbolen erreichen können, in freier Wildbahn sind es dennoch wesentlich weniger.

Schritt II b: Die Konkatenation von Symbolen

Die zweite wichtige Erweiterung eines einfachen Lexikons ist die Fähigkeit mehrere Symbole zu größeren Ausdrücken zu verketten. Dies stellt die Grundlage einer Syntax dar.

Jackendoff ist es wichtig zu betonen, dass diese Fähigkeit unabhängig vom vorherigen Schritt ist. Es ist durchaus möglich einen großen Wortschatz zu besitzen, ohne jedoch irgendeine Syntax zu benutzen.

Biologisch bewiesen wurde dies durch die Entdeckung des Wolfkindes Genie im Jahre 1970, die gleich nach ihrem Auffinden begann, in mit Kleinkindern vergleichbarer Geschwindigkeit, Wörter zu erlernen, obwohl sich ihre syntaktischen Fähigkeiten selbst zehn Jahre danach noch kaum ausgebildet hatten. (vgl. [Cur77])

Bei der Verkettung zweier Symbole multipliziert sich die Anzahl pragmatischer Bedeutungsmöglichkeiten. Man nehme als Beispiel, ein Kleinkind gibt die Wörter „Tina“ und „Apfel“ von sich. Die Bedeutung ist nicht eindeutig bestimmbar. Ist Tina ein Apfel? Möchte Tina einen Apfel? Mag Tina Äpfel? Fiel ein Apfel auf Tina? Isst Tina einen Apfel? Die Reihe möglicher Bedeutungen kann ins Unendliche fortgesetzt werden. Es ist jedoch wichtig zu betonen, dass die hier genannte Art der Konkatenation Symbole, die bereits eine Bedeutung tragen, zu einer komplexeren Aussage zusammenfasst. Im nächsten Schritt wird eine andere Art der Verkettung zu bedeutungstragenden Symbolen erläutert.

Schritt III a: Die Entwicklung eines phonologisch-kombinatorischen Systems zur Erweiterung der offenen unbegrenzten Symbolklasse

Mit steigender Größe der Symbolklasse, also der unterschiedlichen Laute innerhalb eines Kommunikationssystems, steigt auch die Schwierigkeit sie voneinander zu unterscheiden. Daher bestehen die Bedeutungsträger moderner Sprachen, Wörter, auch zumeist aus mehr als nur einem Laut, sie werden aus einer Reihe, generell zwischen einem und mehreren dutzend verschiedener bedeutungsloser Sprechlaute gebildet.

Die daraus abgeleiteten biologischen Anforderungen an einen Teilnehmer eines Kommunikationssystems sind die Fähigkeiten eine Abfolge dieser Sprechlaute gut unterscheidbar zu produzieren und die Lautäußerungen anderer Sprecher zu hören und wieder in ihre Bestandteile zerlegen zu können.

Ein Zwischenschritt in der Entwicklung eines phonembasierten Vokabulars könnte die Benutzung der Silbe, also einer Konkatenation von Phonemen, als Einheit des Systems zur Erzeugung von Wörtern sein. Die so erstellten Bausteine können beliebig zu neuen Wörtern zusammengesetzt werden, dabei sind sowohl Länge als auch die Phonem- bzw. Silbenabfolge eines jeden Wortes frei wählbar. Dieser Schritt ist in der Entwicklung von Sprache unabdingbar, nur so kann die notwendige Größe eines Vokabulars erreicht werden.

Schritt III b: Die Symbolposition als Träger semantischer Beziehungen

Eine Erweiterung zu Schritt II b ist eine genauere Festlegung der Symbolposition innerhalb einer Konkatenation zur Verdeutlichung der semantischen Beziehungen. Dabei stehen zwei Arten von Innovation orthogonal zueinander, die lineare Ordnung verketteter Symbole um Beziehungen zwischen ihnen zu verdeutlichen und die Einführung einer neuen Art von Vokabular, das direkt die Beziehung mehrerer Symbole kodiert.

Dabei gibt es in modernen Sprachen einige Prinzipien zur Verkettung verschiedener Symbole, die als eine Vorstufe zu Syntax bezeichnet werden können. Das bekannteste unter ihnen ist „Agent zuerst“ „Fokus zuletzt“, in anderen Worten, das Subjekt steht stets vor dem Objekt. Eine Aussage wie „schlägt Tim Tom“ bedeutet also, dass Tim Tom schlägt und nicht das Gegenteil.

Schritt IV: Protosprache

Fasst man alle der vorher erläuterten Schritte zusammen, so ergeben sich eine Menge von Eigenschaften, die denen einer „*Protosprache*“ entsprechen. Klein und Perdue haben in [KP97] einige generelle Eigenschaften der Sprachnutzung

erwachsener Fremdsprachenlerner zusammengestellt, die sich diese Sprache ohne Mithilfe eines Lehrers aneigneten. Sie nennen diese Art des Sprachgebrauchs „*Basic Variety*“. Ihre Eigenschaften sind im Folgenden aufgelistet:

- Lexikalische Kompetenz
- Fehlen inflexionaler Morphologie (Verben werden z.B. weder konjugiert noch werden verschiedene Zeiten genutzt)
- Fehlen von Nebensätzen
- Einfache, stark semantisch basierte Prinzipien der Wortreihenfolge

Bis auf den letzten Punkt ist diese Auflistung identisch mit Bickertons Definition einer „*Protosprache*“, unter der er die Organisation von Pidginsprachen, die grammatische Kompetenz von Genie und sprachtrainierten Menschenaffen zusammenfasst. Bickerton sieht jedoch keine starre Wortreihenfolge innerhalb der von ihm definierten Protosprache.

Die Protosprache ist ein logischer Schritt auf dem Weg zu einer modernen Sprache. Obwohl noch keine bzw. kaum syntaktische Elemente in ihr vorhanden sind, verfügt sie bereits über eine beachtliche Mächtigkeit. Alle weiteren Schritte behandeln ausschließlich die Einführung syntaktischer Strukturen in das bisherige Kommunikationssystem und gehen damit zu weit über das Ziel und das benutzte Modell dieser Arbeit hinaus. Daher wird auf eine Erläuterung der Folgeschritte an dieser Stelle verzichtet.

2.3.3 Stufen der Evolution eines Kommunikationssystems

Luc Steels[5] gibt in [Ste05] ebenfalls eine mögliche Einteilung sprachlicher Evolutionsstufen, welche im Folgenden dargestellt werden. Er unterscheidet dabei ebenfalls in lexikalische, syntaxfreie Kommunikationssysteme, die in die Stufen I-III unterteilt werden und in syntaxbasierte Systeme, die in den Stufen IV-VII genauer beschrieben werden.

Stufe I: Das „Namengebungs-Spiel“

Die erste Stufe repräsentiert die Evolution einer Sprache auf der niedrigsten Ebene. Auf dieser Stufe weist jedes Individuum einer Gemeinschaft jedem Objekt der Umwelt einen *Namen* in Form eines eindeutigen Signals oder Wortes zu. Um ein gemeinsam genutztes Lexikon aufzubauen und aufrechtzuerhalten, halten die Individuen einer Gemeinschaft laut Luc Steel folgende Vereinbarungen innerhalb der Sprachspiele ein:

[5] Luc Steels, Professor für Informatik an der Freien Universität Brüssel.

- **Erfindung**
  Wird ein neuer Name benötigt, generiert der Sprecher ein neues Wort für ein Objekt.

- **Aneignung**
  Trifft ein Hörer auf ein neues Wort und ist er in der Lage das entsprechende Objekt durch ein Sprachspiel zu referenzieren, so assoziiert ab diesem Zeitpunkt immer dieses neue Wort mit diesem Objekt.

- **Anpassung**
  Alle Individuen sollen die Stärke ihrer Assoziationen auf den Kommunikationserfolg innerhalb einer Gemeinschaft anpassen.

Stufe II: Kategorisierung

In dieser Stufe der Evolution eines lexikalischen Kommunikationssystems werden die in Stufe I erzeugten Namen durch *Kategorien* ersetzt. Demnach müssen Individuen in der Lage sein, einzelnen Objekten Kategorien wie z. B. „rot“ oder „Auto“ zuzuordnen. Um ein Repertoire an solchen Kategorien aufzubauen und aufrecht zu erhalten, müssen die Individuen einer Gemeinschaft ähnliche Vereinbarungen wie in Stufe I einhalten:

- **Erfindung**
  Wird eine neue Kategorie benötigt, generiert der Sprecher eine neue Kategorie für ein Objekt.

- **Aneignung**
  Trifft ein Hörer auf eine neue Kategorie und ist er in der Lage das entsprechende Objekt durch ein Sprachspiel zu referenzieren, so assoziiert er ab diesem Zeitpunkt immer diese neue Kategorie mit diesem Objekt.

- **Anpassung**
  Alle Individuen sollten die Stärke ihrer Assoziationen auf den Kommunikationserfolg innerhalb einer Gemeinschaft anpassen.

Stufe III: Komposition

Individuen eines Kommunikationssystems, dessen Sprache sich auf dieser Stufe befindet, nutzen Kompositionen von Kategorien um ein Objekt der Umwelt zu beschreiben. Ein Signal besteht somit aus einer Verkettung von Wörtern, dessen einzelne Wörter für Kategorien stehen. Dies setzt eine gewisse strukturelle

Vielseitigkeit der Objekte der Umwelt voraus. Der Unterschied zu Stufe II ist somit die Tatsache, dass die Information nicht mehr in einem Wort kodiert wird, sondern eine Konkatenation von Wörtern einzelner Kategorien erstellt wird. Um diese Stufe zu erreichen benötigen die Agenten einer Simulation komplexere Berechnungsmechanismen. Neubauer hat gezeigt, dass das Nutzen von Kompositionen zur Objektbeschreibung bei ausreichender Struktur der Objekte der Umwelt vorteilhaft ist (siehe [Neu03]).
An dieser Stelle ist auch in der Einteilung von Steels eine Mächtigkeit erreicht, die der von Jackendoffs Protosprache sehr nahe kommt. Die folgenden vier Schritte

- Konstruktionen
- Metagrammatik
- Prädikate höherer Ordnung
- Metasprache

beschreiben die schrittweise Einführung einer Syntax in eine Sprache, startend bei einer möglichen Optimierung der Kommunikation durch syntaktische Konstruktionen bis hin zum finalen Schritt der Metasprache, in der eine Sprache mächtig genug ist, um sich selbst zu beschreiben. Da diese Stufen jedoch in dem Modell dieser Arbeit nicht benötigt werden, wird von einer weitergehenden detaillierten Beschreibung an dieser Stelle abgesehen.

2.4 Menschliche Sprache und ihre Veränderungen

In diesem Unterkapitel wird der Fokus auf die Klassifizierung, die Bestandteile und die Veränderungen der menschlichen Sprache gelegt. Der erste Abschnitt gibt eine Zusammenfassung über die Zusammenhänge zwischen Sprachen, Sprachfamilien, Ursprachen und Dialekten. In zweiten Abschnitt werden die Bestandteile der menschlichen Sprache dargelegt, während im letzten Abschnitt dieses Unterkapitels die Bestandteile und Veränderungen der Lautlehre dargelegt werden.

2.4.1 Sprachliche Gemeinsamkeiten und Beziehungen

Heutzutage sind zirka 6900 Sprachen weltweit klassifiziert, die in über 100 Sprachfamilien, die aus *genetisch verwandten Sprachen* bestehen, eingeteilt werden (siehe [Lew09]). Aufgrund der Vielzahl verschiedener Sprachen werden Sprachfamilien weiter unterteilt, in *Untergruppen* und *Zweige*. Dabei werden die Beziehungen der Sprachen innerhalb einer Sprachfamilie genauer untersucht und so aus einer Vielzahl teilweise ineinander verschachtelter Untergruppen ein sogenannter Sprachstammbaum erstellt.

Einteilung in Sprachfamilien

Die Standardmethode zur Einteilung der Sprachfamilien ist die sogenannte *vergleichende Methode*. Sie basiert auf der Tatsache, dass viele Wörter verwandter Sprachen in ihrer phonologischen und morphologischen Struktur in Zusammenhang gebracht werden können. Abbildung 2 zeigt beispielhaft eine Tabelle von Wortformen mehrerer moderner romanischen Sprachen und ihrer Ursprache, der lateinischen Sprache.

	Latei-nisch (L.)	Französ-isch (Fr.)	Italie-nisch (It.)	Spanisch (Sp.)
„Ding“	*c*ausa	*ch*ose	*c*osa	*c*osa
„Kopf“	*c*aput	*ch*ef	*c*apo	*c*abo
„Pferd“	*c*aballus	*ch*eval	*c*avallo	*c*aballo
„singen“	*c*antare	*ch*anter	*c*antare	*c*antar
„Hund“	*c*anis	*ch*ien	*c*ane	
„Ziege“	*c*apra	*ch*èvre	*c*apra	*c*abra
„Pflanze“	*pl*anta	*pl*ante	*pi*anta	*ll*anta
„Schlüssel“	*cl*avis	*cl*ef	*chi*ave	*ll*ave
„Regen“	*pl*uvia	*pl*uie	*pi*oggia	*ll*uvia
„acht“	oc*t*o	hui*t*	ot*t*o	och*o*
„Nacht“	nox/noc*t*is	nui*t*	no*tt*e	noc*h*e
„Tatsache“	fac*t*um	fai*t*	fa*tt*o	hec*h*o
„Milch“	lac*t*e	lai*t*	la*tt*e	lec*h*e
„Tochter“	*f*ilia	*f*ille	*f*iglia	*h*ija
„schön“	*f*ormosus			*h*ermoso

Abb. 2 Wortformen einiger romanischer Sprachen
Quelle: [Lyo81], S. 177

An diesem Beispiel lässt sich das Prinzip der vergleichenden Methode verdeutlichen. Die dort aufgelisteten Worte sind sich „ähnlich“. Ähnlichkeit bedeutet in diesem Fall, dass zwischen den einzelnen Sprachen Lautentsprechungen existieren, die im Folgenden aufgelistet werden. (vgl. Abb. 3) Man beachte hierbei, dass es sich um Lautsymbole, nicht um Buchstaben handelt.

(1) L. [k] = Fr. [ʃ] = It. [k] = Sp. [k]
(2) L. [pl], [kl] = Fr. [pl], [kl] = It. [pʲ], [kʲ] = Sp. [ʎ]
(3) L. [kt] = Fr. [it] = It. [tt] = Sp. [tʃ]
(4) L. [f] = Fr. [f] = It. [f] = Sp. [h]

Abb. 3 Lautentsprechungen in den romanischen Sprachen
Quelle: [Lyo81], S.179

Die Regeln, nach denen sich Laute verändern, werden *Lautgesetze* genannt. Es existiert beispielsweise ein Lautgesetz, das besagt, dass sich das lateinische [k] im Französischen zu einem [ʃ] verändert hat. Mit Hilfe mehrerer Lautgesetze ist es folglich möglich, die Veränderung der aufgelisteten Wörter aus der Ursprache in jede moderne Nachfolgesprache nachzubilden. Diese Art des Verwandtschaftsbelegs, mit dem besonders die Verwandtschaftsbeziehungen innerhalb der indogermanischen Sprachfamilie untersucht wurden, geht auf die Gruppe der *Junggrammatiker* zurück.

Bereits in Abb. 2 ist ersichtlich, dass sich nicht alle Wörter der aufgelisteten Sprachen ähneln. Das spanische Wort „perro“ (dt. „Hund“) hat keinerlei Gemeinsamkeiten mit den Wörtern „canis“, „chien“ oder „cane“ anderer romanischer Sprachen. Das Wort „perro“ stammt offensichtlich nicht vom lateinischen Wort „canis“ ab, sondern hat eine andere Wortherkunft. Diese so genannte *zufällige Mutation* des Wortes entstammt nicht einer gesetzmäßigen Änderung, sie kann daher auch nicht mit einem Lautgesetz nachempfunden werden. Da zufällige Mutationen dieser Art jedoch bei größeren Zeitspannen die regulären Veränderungen überdecken, ist es irgendwann nicht mehr möglich, die Verwandtschaft von Sprachen regulär zu belegen, es existieren schlicht zu viele Ausnahmen für eine Regel. Verschiedene wissenschaftliche Quellen legen sich hier auf eine ungefähre Zeitspanne von 5000 bis 10000 Jahren fest.

Der Linguist Joseph Greenberg entwickelte daher den *lexikalischen Massenvergleich* mehrerer Sprachen, dessen Ziel es nicht mehr ist, Gesetzmäßigkeiten für eine Transition von einer Sprache in eine andere zu finden, sondern lediglich Ähnlichkeiten innerhalb der Klangmuster bedeutungsgleicher Wörter verschiedener Sprachen. Greenberg ermöglichte mit seiner Arbeit die Klassifikation vieler Sprachfamilien, deren Sprachen sich bereits vor mehreren tausenden Jahren trennten und von denen kaum schriftliche Zeugnisse vorliegen, zu nennen sind beispielsweise die afrikanischen und amerikanischen Sprachen.

Ursprachen

Die Mitglieder einer Sprachfamilie stammen alle von einer gemeinsamen *Ursprache* (auch *Vorgängersprache*, *Protosprache*) ab. Meist sind die Vorgängersprachen aufgrund fehlender Schriftzeugnisse nicht schriftlich überliefert. Dennoch ist es meist möglich, die jeweilige *Protosprache* einer Sprachfamilie durch einen systematischen Vergleich der nachfolgenden Einzelsprachen zu rekonstruieren. Károly Rédei veröffentliche im Jahre 1988 das „*Uralisch Etymologische Wörterbuch*“ [Red88], in dem er versucht, die Ursprache der uralischen Sprachfamilie zu rekonstruieren. Dies ist das Ziel eines Teilgebiets der Linguistik, nämlich der sogenannten *Vergleichenden Sprachwissenschaft*.

In manchen Fällen ist es jedoch einfacher, die Verwandtschaft innerhalb einer Sprachfamilie zu belegen. So ist es unverkennbar, dass die romanischen Sprachen vom Lateinischen abstammen und dies aus zwei Gründen: Erstens sind sowohl alle modernen romanischen Standardsprachen als auch ihre mittlerweile ausgestorbene Protosprache, die lateinische Sprache, ausreichend schriftlich dokumentiert, um eine genetische Verwandtschaft zu belegen, zweitens ist die Abstammung historisch belegbar: So gehörten sämtliche europäische Länder, in denen heute romanische Sprachen gesprochen werden, zum antiken Römischen Reich. Ihre modernen Sprachen haben sich demnach aus dem sogenannten „Vulgärlatein", dem gesprochenen Latein der Spätantike, nach dem Zerfall des Römischen Reiches entwickelt. Die weltweite Verbreitung der romanischen Sprachen ist der Kolonialisierungspolitik der europäischen Länder geschuldet. Erstaunlicherweise gehört die lateinische Sprache nicht zur romanischen Sprachfamilie. Sie wird zur mittlerweile komplett ausgestorbenen Familie der italischen Sprachen gezählt.

Dialekte

Neben der Klassifizierung von Sprachen und ihrer Einteilung in Sprachfamilien entwickelte sich ein weiteres Teilgebiet der Linguistik, die *Dialektologie*, die im Gegensatz zur historisch vergleichenden Sprachwissenschaft nicht verschiedene Sprachen zu kategorisieren versucht, sondern die verschiedenen Dialekte einer einzelnen Sprache untersucht.

Die Dialektologie wurde durch die Gebrüder Grimm im Jahre 1854 mit dem *„Deutschen Wörterbuch"* [Gri54] begründet, in dem auch mundartliche Varianten des Deutschen zu finden waren. Ab 1875 erfasste Georg Wenker systematisch alle Dialekte des deutschen Sprachgebiets. Aufgrund der Vielzahl unterschiedlicher Dialekte alleine im deutschen Sprachgebiet existiert eine sehr große Menge an Literatur zu diesem Thema. Die meisten Veröffentlichungen betrachten daher ausschließlich die Dialekte einer bestimmten Stadt oder Region. So existiert beispielsweise ein „*Frankfurter Wörterbuch*" [Brü71], das die dialektalen Eigenarten der Stadt Frankfurt am Main beschreibt, das Werk „*Op Kölsch jesaat*" [Cas94] widmet sich der Mundart der Stadt Köln. Innerhalb einer Sprache unterscheidet man heute zwischen verschiedenen *Umgangssprachen*, *Regionalsprachen* und *Dialekten*. Im Gegensatz zu einer *Hochsprache* sind diese drei Formen nicht standardisiert und werden folglich nur im mündlichen Sprachgebrauch verwendet.

Die genaue Abgrenzung zwischen Dialekten und Hochsprachen ist teilweise umstritten bzw. nicht möglich. Als Beispiel ist die Luxemburgische Sprache zu nennen, die erwiesenermaßen zum moselfränkischen Dialektkontinuum gehört, daher als Dialekt des Hochdeutschen gesehen werden kann. Dennoch ist sie eine der Amtssprachen Luxemburgs und wird in standardisierter Form in den Medien genutzt. Max Weinreich, ein auf Jiddisch spezialisierter Linguist, soll den

Unterschied zwischen einer Hochsprache und einem Dialekt wie folgt erklärt haben: *„Eine Sprache ist ein Dialekt mit einer Armee und einer Marine.*“ Diese Definition, obgleich sie in historischem Zusammenhang mit der Judenverfolgung im Dritten Reich gesehen werden muss, beschreibt treffend, dass eine genaue Abgrenzung zwischen Hochsprache und Dialekt nicht anhand der beiden Sprachen selbst geklärt werden kann, sondern, dass das Attribut Hochsprache alleine der Eigenschaft geschuldet ist, dass diese Sprachvarietät die Nationalsprache eines Staates ist und folglich standardisiert wurde.

2.4.2 Die Bestandteile der menschlichen Sprache

Ausgehend von den sprachlichen Evolutionsstufen aus Unterkapitel 2.3 ergibt sich eine Liste der Bestandteile moderner Sprachen. Sie bauen evolutionär aufeinander auf und werden daher von Stufe zu Stufe komplexer. Im Folgenden werden sie kurz vorgestellt, sortiert nach ihrem zeitlichen Auftreten und somit auch ihrer Komplexität. Eine genauere Erläuterung der Lautlehre erfolgt im folgenden Abschnitt. Zwar ist die menschliche Lautlehre nicht als Grundlage für die Entwicklung des Simulationsmodells dieser Arbeit zu sehen, alleine schon, da die Agenten des Modells ihre Wörter nicht unter den Bedingungen des menschlichen Artikulationsapparats bilden. Vielmehr sollen die Abschnitte 2.4.2 und 2.4.3 deutlich machen, welche Aspekte der menschlichen Sprache im Modell dieser Arbeit ausgeblendet werden, beziehungsweise ausgeblendet werden müssen. (vgl. Abschnitt 3.5.2) Da die beiden höheren Stufen im Modell dieser Arbeit nicht benutzt werden wird auf eine detaillierte Erläuterung in Form eigener Abschnitte verzichtet.

1. Stufe: Lautlehre

Die Lautlehre befasst sich mit der Basis verbaler Kommunikation. Sie wird unterteilt in zwei Untergebiete, die *Phonetik* und die *Phonologie*. Die Phonetik betrachtet dabei die Erzeugung, Übertragung und den Empfang akustischer Signale, während sich die Phonologie der Untersuchung der Phoneme, der kleinsten bedeutungsunterscheidenden Elemente der Sprache, widmet.

2. Stufe: Grammatik

Die Grammatik unterscheidet sich ebenfalls in zwei Untergebiete. Zum einen beinhaltet sie den Einsatz der *funktionalen Morpheme*, also Morpheme, die selbst keine Wörter bilden, sondern diese in einer bestimmten Art verändern, so beispielsweise das Morphem „en“ bei „Frauen“ mit am Wortstamm „Frau“ als Kodierung des Plurals. Der zweite Bestandteil ist die *Syntax*, jene Regeln, nach

denen Wörter zu größeren funktionellen Einheiten wie Sätzen zusammengesetzt werden können.

3. Stufe: Lexikologie

Die Lexikologie als letzte und höchste Stufe der natürlichen Sprachen untersucht die Bedeutung und Bildung von Wörtern, die in ihrer Gesamtheit das Lexikon, also den Wortschatz einer Sprache bilden. Im Gegensatz zur Phonologie betrachtet die lexikalische Semantik das System der *Lexeme* (*lexikalischen Morpheme*), die die Wortstämme, die „*Grundbestandteile eines Wortes*“ (siehe [DS06], S. 83) bilden.

2.4.3 Die Lautlehre: Phonetik und Phonologie

In diesem Abschnitt werden die Untergebiete Phonetik und Phonologie betrachtet. Die Phonetik setzt dabei einen stark biologischen Schwerpunkt, nämlich die Wirkungsweise und Fähigkeiten der menschlichen Organe. Die Phonologie hingegen widmet sich dem System der Phoneme und ist damit weit stärker linguistisch ausgerichtet.

Phonetik

In diesem Abschnitt werden die phonetischen Merkmale eines Kommunikationssystems betrachtet. Hierbei handelt es sich um die Erzeugung, die Übertragung und den Empfang von sprachlichen Lauten.

Erzeugung

Die *artikulatorische Phonetik* betrachtet die Erzeugung der Laute. Daher liegt der Fokus dieses Teilgebiets auf dem Sprechapparat eines Individuums. Da in dieser Arbeit ein Vergleich zu menschlicher Kommunikation gezogen werden soll, werden die Fähigkeiten des menschlichen Sprechapparats beleuchtet. Dieser ist in der Lage, eine Vielzahl von sprachlichen Lauten zu produzieren. Eine Auflistung und Kategorisierung dieser Laute befindet sich in einem Lautschriftsystem. Das weltweit am meisten genutzte ist das „*Internationale Phonetische Alphabet*“, welches von der „*International Phonetic Association*“ entwickelt wurde. Selbst wenn man nur die Anzahl der Vokale und der hauptsächlich genutzten pulmonalen Konsonanten, die von einem Menschen erzeugt werden können, betrachtet, so summiert sich ihre Anzahl auf weit über 100 verschiedene Laute. Die folgenden Abbildungen zeigen die erzeugbaren Vokale und Konsonanten des menschlichen Sprechapparats.

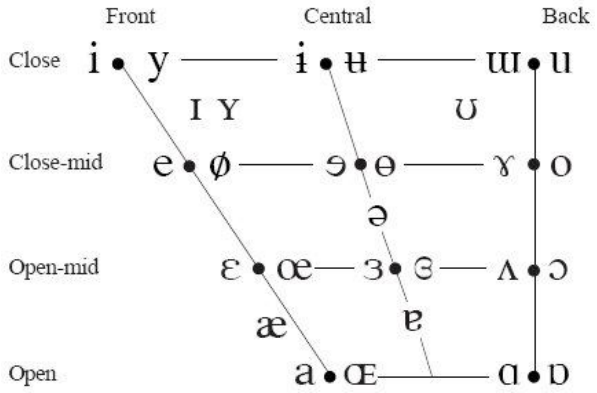


Abb. 4 Vokale im Internationalen Phonetischen Alphabet
Quelle: [IPA05]

Vokale werden durch drei Angaben definiert. Es wird dabei zwischen der Stelle der Lauterzeugung im Mund (x-Achse), der Öffnung des Mundes (y-Achse) und der Rundung der Lippen (ungerundet/gerundet) unterschieden. Der Laut /a/, der den ersten Buchstaben des Alphabets repräsentiert, befindet sich beispielsweise unten links in der Abbildung. Er wird erzeugt mit einem geöffneten Mund bei ungerundeten Lippen und seine Lauterzeugung findet im vorderen Bereich des Mundes statt.

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Abb. 5 Pulmonale Konsonanten im Internationalen Phonetischen Alphabet
Quelle: [IPA05]

Bei den pulmonalen Konsonanten wird zwischen Artikulationsart und Artikulationsort unterschieden. Die Ortsangabe befindet sich auf der x-Achse, die Art auf der y-Achse. Befindet sich mehr als ein Laut in einer Zelle, so stellt der rechte Laut die stimmhafte Variante der beiden dar. Schattierte Zellen erscheinen aus heutiger Sicht von einem Menschen nicht erzeugbar zu sein, da es ihm biologisch nicht möglich ist. Der Laut /b/, der sich an zweiter Stelle des Alphabets befindet, findet sich hier in der Zelle oben links an zweiter Stelle. Er wird also bilabial-plosiv erzeugt und ist die stimmhafte Variante des Lautes /p/.

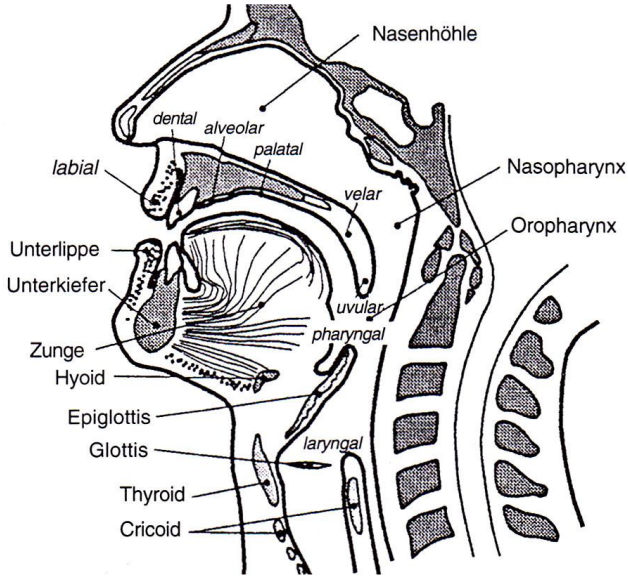


Abb. 6 Die menschlichen Artikulationsarten
Quelle: [Gra06]

Da die Erzeugung sprachlicher Laute vollständig von den biologischen Fähigkeiten des Sprechers abhängt, erscheint eine Betrachtung der möglichen Veränderungen nur innerhalb evolutionärer Zeiträume sinnvoll zu sein. Zum gegenwärtigen Zeitpunkt nicht erzeugbare Laute können auch, da die Unfähigkeit der Erzeugung an die biologischen Eigenschaften gekoppelt ist, nicht antrainiert werden. Lediglich die Tatsache, dass ein größeres Lautrepertoire zu einem biologischen Vorteil führt, könnte sich über eine große Zeitspanne in einer Änderung des Lautrepertoires bemerkbar machen. Betrachtet man jedoch die Sprachen der evolutionären Neuzeit und ihre Veränderungen, können Veränderungen an diesem Punkt eindeutig vernachlässigt werden.

Übertragung

Das zweite Teilgebiet der Phonetik ist die *akustische Phonetik*. Sie befasst sich mit der Veränderung der Laute nach dem Verlassen des Senders bis zu ihrem Eintreffen beim Empfänger. Hierbei sind neben der Abschwächung und Verformung der gesendeten Schallwellen ebenfalls Störgeräusche, auch Rauschen genannt, zu berücksichtigen. (vgl. Abb. 7) Veränderungen in diesem Punkt sind jedoch auch zu vernachlässigen, da sie auf unveränderlichen physikalischen Gesetzmäßigkeiten beruhen.

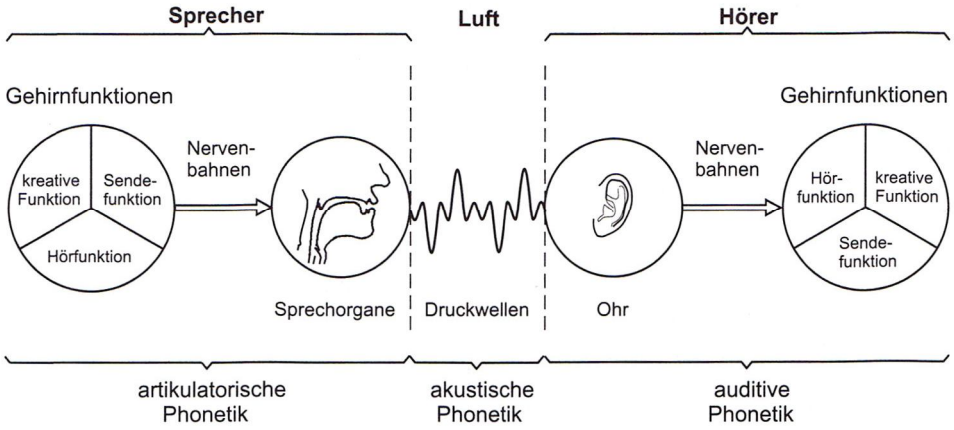


Abb. 7 Die drei Teilgebiete der Phonetik
Quelle: [Gra06]

Empfang

Als drittes Teilgebiet der Phonetik ist die *auditive Phonetik* zu nennen. Sie befasst sich mit der auditiven Wahrnehmung des Hörers und der Verarbeitung der Laute im Gehirn. Das menschliche Gehör ist dabei in der Lage zwischen den oben aufgeführten Lauten zu unterscheiden und sie im Gehirn als solche wieder zu erkennen. Ein leichtes Rauschen kann ebenfalls vom Gehirn herausgefiltert werden. Die Verarbeitung der aufgenommenen Laute geschieht in mehreren Schritten: (vgl. Abb. 8)

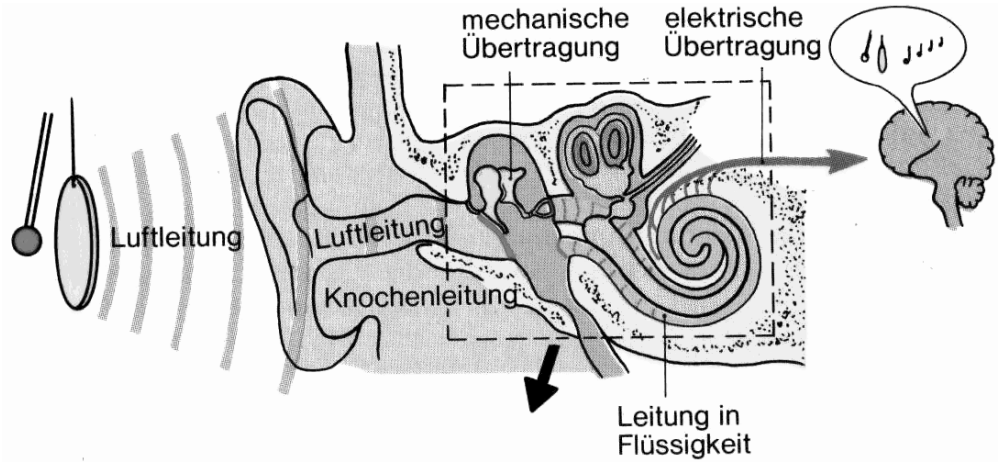


Abb. 8 Das menschliche Gehör
Quelle: [Sil91], S. 319

1. Das Außen- und Mittelohr dienen der Schallaufnahme und seiner Weiterleitung.

2. Das Innenohr wandelt Schallreize in neuronale Impulse um, die vom Hörnerv weitergeleitet werden.

3. Das Gehirn wertet die ankommenden Signale aus, dies beinhaltet eine Filterung und weitere Verarbeitung der eingehenden Nervenimpulse.

Da diese Fähigkeit ebenfalls auf den biologischen Eigenschaften des Hörers basiert, ergibt sich eine ähnliche Betrachtung der Veränderungen wie bei der Lauterzeugung: Evolutionär sind durchaus Veränderungen denkbar, betrachtet man jedoch die Sprachen der Neuzeit sind diese zu vernachlässigen.

Phonologie

Die Phonologie als zweiter Teil der Lautlehre untersucht ein System von Phonemen, den kleinsten bedeutungsunterscheidenden Elementen einer Sprache. Phoneme sind eine Teilmenge der im Internationalen Phonetischen Alphabet aufgeführten Laute, jedoch mit der Bedingung, dass sie zu einem Bedeutungsunterschied in einer Sprache beitragen. Die Feststellung der bedeutungsunterscheidenden Funktion erfolgt über das Finden von Minimalpaaren, also zweier Wörter, die sich nur durch einen einzigen Laut unterscheiden. Als Beispiel sind die deutschen Wörter „Seil“ und „Teil“ zu nennen. Der einzige Unterschied sind die beiden Laute /s/ und /t/ an erster Position des Worts. Somit ist dieses Wortpaar ein Minimalpaar und die beiden Laute /s/ und /t/ bedeutungsunterscheidende Elemente, also Phoneme dieser Sprache.

Innerhalb der Phonologie gibt es nach [McM99] sechs Arten des Lautwandels, die im Folgenden detailliert erläutert werden. Lautveränderungen können dabei einerseits konditioniert, also nur in bestimmten abgrenzbaren Fällen, oder aber auch unkonditioniert, also bei allen Vorkommen eines bestimmten Lauts auftreten. Sie können auch regulär, irregulär oder aber sporadisch auftreten. (vgl. [McM99], S. 14-16)

Assimilation

Eine typischerweise reguläre und konditionierte Veränderungsform des Lautwandels ist die Assimilation, die bewirkt, dass ein Laut einem anderen in seiner Umgebung ähnlich wird. Eine Assimilation kann partiell oder aber vollständig geschehen. Als Beispiele sind die Veränderungen des altenglischen Worts „efn“ (dt.: „eben“) ins westsächsische „emn“ für eine partielle und der Übergang des lateinischen Worts „septem“ (dt.: „sieben“) ins italienische „sette“ für eine vollständige Assimilation zu nennen.

Dissimilation
Im Gegensatz zu Assimilation treten Dissimilationen eher sporadisch auf und erscheinen nur in isolierten Wörtern. Sie sind Prozesse, in denen innerhalb eines Worts „*Segmente einander unähnlicher gemacht werden*“. (siehe [Mei07], S. 99) Ein Beispiel ist die Veränderung des lateinischen Worts „purpura“ (dt.: „Purpur“) ins altfranzösische „purpre“ und von dort weiter als englisches Lehnwort in „purpel“ und letztendlich zu „purple“.

Epenthese
Die Epenthese ist ein Prozess, in dem Segmente eingefügt werden um die Aussprache der Wörter zu erleichtern. Exemplarisch wird dies am lateinischen Wort „schola“ (dt.: „Schule“) gezeigt, dass zuerst ins altfranzösische „escole“ transformiert wurde und später zum heutigen „école“ wurde.

Elision
Der vollständige Gegensatz zur Epenthese ist die Elision, bei der Laute innerhalb eines Worts komplett wegfallen. Diese Veränderung der Lautstruktur tritt ebenfalls sporadisch auf und dient ebenfalls einer vereinfachten Artikulation. Es wird bei der Elision zwischen drei verschiedenen Varianten unterschieden. Eine Apokope ist ein Lautwegfall am Wortende, so beispielsweise bei den umgangssprachlichen deutschen Formen „nich“ von „nicht“ und „is“ von „ist“. Eine Synkope ist ein Wegfall eines Lautes im Wortinneren. Als Beispiel ist hier das altenglische Wort „munecas“ (dt.: Mönche) zu nennen, das im modernen Englisch zu „monks“ wurde. Eine dritte Variante ist die Haplologie. Bei ihr fällt eine ganze Silbe aus einer Folge von ähnlich klingenden Silben weg. Dies geschah unter anderem mit der altlateinischen Wortfolge „stipi-pendium“, die zum lateinischen Wort „stipendium“ transformiert wurde.

Abschwächung
Anders als die Elision existiert ebenfalls eine regelmäßige Form der Lauttilgung über eine Reihe von Veränderungen, die einen Laut abschwächen. Dies geschieht zumeist entlang der Sonoritätshierarchie (vgl. Abb. 9), in der sich harte Konsonanten langsam zu weichen Vokalen abschwächen.

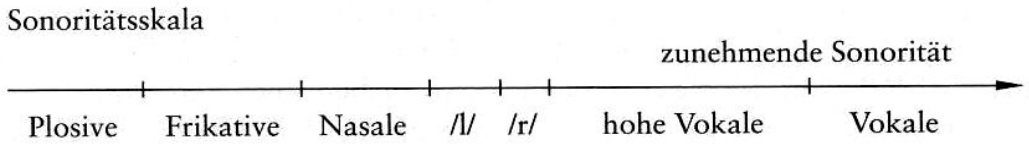


Abb. 9 Sonoritätshierarchie
Quelle: [Mei07]

Metathese
Die letzte Variante des Lautwandels ist die Metathese, bei der die Lautabfolge innerhalb eines Wortes vertauscht wird. Als Beispiele sind hierzu die beiden altenglischen Wörter „acsian“ und „brid“ zu nennen, die zu den modernen englischen Wörtern „ask“ (dt.: „fragen“) und „bird“ (dt.: „Vogel“) wurden.

2.5 Erlernte Sprachen und Dialekte im Tierreich

Im folgenden Unterkapitel wird die Dialektbildung innerhalb erlernter verbaler Kommunikationsformen im Tierreich erläutert. Trotz einer Unmenge verschiedener Arten hat sich nur bei sehr wenigen Familien überhaupt die Fähigkeit zu einer verbalen Kommunikation entwickelt. Oliphant, der dieses Phänomen in [Oli97] untersucht gibt dazu drei mögliche Erklärungen: Zum einen fehlt den meisten Tieren sicherlich die Notwendigkeit miteinander zu kommunizieren. Die Evolution war gut genug, um den meisten Tieren die wenigen notwendigen Laute, wie das Anlocken von paarungswilligen Artgenossen, die Revierverteidigung und mögliche Alarmrufe als Reaktion auf einen Angriff von Raubtieren, angeboren mitzugeben.

Als zweiten Grund führt Oliphant die hohen Kosten einer erlernten Fähigkeit an. Im Gegensatz zu angeborenen Fähigkeiten, die einem Tier nahezu direkt nach der Geburt zur Verfügung stehen, erfordern erlernte Fähigkeiten eine relativ lange Dauer des Erlernens und der Übung bis sie zuverlässig funktionieren. Das Erlernen einer Kommunikationsform, gerade in Situationen, die die biologische Fitness betreffen, kann daher als Luxus der Lebewesen angesehen werden, die in der Lage sind, ihren Nachkommen im großen Maße elterliche Pflege in der ersten Phase ihres Lebens zukommen zu lassen.

Die dritte und letzte mögliche Erklärung ist, dass die Fähigkeit des Erlernens durch Nachahmen bei den meisten Tieren wahrscheinlich nicht ausgebildet ist. Dies bedeutet, dass die Tiere sehr wohl in der Lage wären eine Aktion zu erlernen, dieses jedoch nie versuchen, da ihnen die Fähigkeit, andere Artgenossen zu observieren und ihre Fähigkeiten zu imitieren, fehlt.

Im Folgenden werden die erlernten Kommunikationsformen der Primaten, Wale, Vögel und sonstiger Tiere dargestellt.

2.5.1 Dialekte bei Schimpansen

Untersuchungen des Pant-Hoot-Rufs (vgl. Abb. 10) bei Schimpansen legen die Vermutung nahe, dass dieser Ruf im Laufe eines Lebens erlernt und dadurch einer Dialektbildung unterworfen ist.

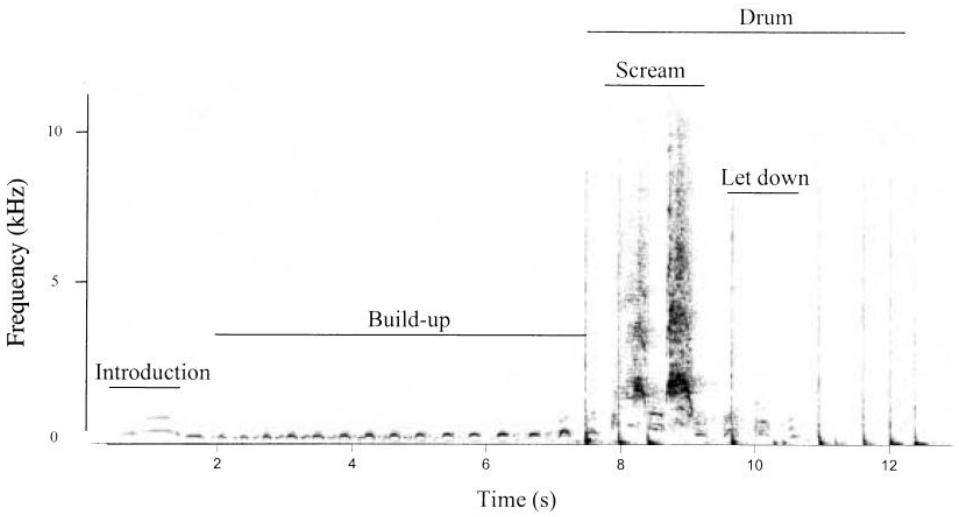


Abb. 10 Der Aufbau eines Pant-Hoot-Rufs. Quelle: [CHVB04]

Marshall, Wrangham und Arcadi untersuchen daher in [MWA99] die Rufe männlicher Schimpansen in zwei Tierparks in den USA. Da die Schimpansen beider Gruppen in unterschiedlichen Regionen Afrikas gefangen wurden, kann die gruppeninterne Ähnlichkeit der Rufe nicht auf genetischer Nähe der Individuen beruhen. Außerdem ist der Lebensraum beider Populationen aus menschlicher Sicht nahezu identisch, sodass Unterschiede zwischen beiden Gruppen nicht auf unterschiedliche Lebensräume zurückzuführen sind. Desweiteren vergleichen sie die Rufe beider in Gefangenschaft lebenden Gruppen mit den Rufen einer Gruppe Schimpansen im Nationalpark Kanyawara in Uganda.

Die Tatsache, dass sich die Rufe der Schimpansenmännchen innerhalb einer Gruppe, in den Fällen der Tierparks in den USA trotz genetischer Unterschiede der Individuen, weit weniger unterscheiden als gruppenferne Rufe deutet auf ein erlerntes Verhalten hin. Es kann also von einem gruppeninternen Prozess der Sprachvereinigung der einzelnen Wortschätze der verschiedenen Schimpansen ausgegangen werden, in dem sich die Rufe der Individuen im Laufe der Zeit stets ähnlicher wurden, bis sie ein gemeinsames Rufrepertoire teilten. Weitere Beweise für ein erlerntes Verhalten innerhalb der Rufausbildung in Schimpansengruppen liefert die Tatsache, dass ein neu zur einer der in Gefangenschaft lebenden Gruppen hinzugestoßenes Männchen eine bis zu diesem Zeitpunkt nicht gekannte Variante des Pant-Hoot-Rufs in dieser Gruppe unter fünf weiteren männlichen

Individuen etablierte. Abbildung 11 zeigt einen grafischen Vergleich des Aufbaus der Pant-Hoot-Rufe der drei Schimpansenpopulationen, unterschieden nach der Anzahl der Aufbauelemente des Rufs und der Länge des Klimaxelements.

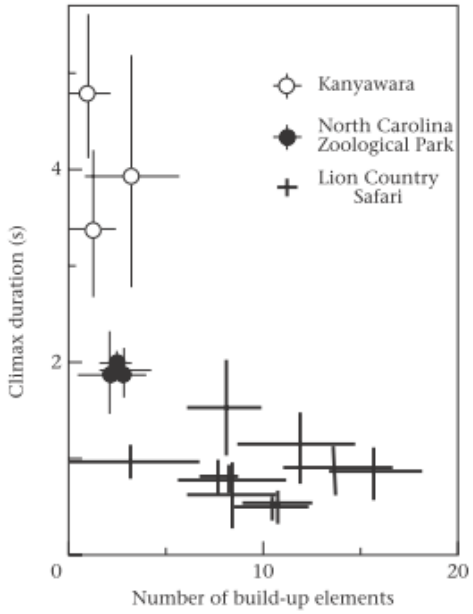


Abb. 11 Vergleich der Pant-Hoot-Rufe der drei Populationen. Quelle: [MWA99]

Mitani, Hasegawa, Gros-Louis, Marler und Byrne präsentieren in [MHGMB92] ähnliche Ergebnisse bei einem Vergleich der Pant-Hoot-Rufe zweier benachbarter Schimpansenpopulationen in Tansania. Bei der Untersuchung sechs verschiedener Merkmale des Rufs stellen sie fest, dass Individuen in den Mahale-Bergen einen Teil des Rufs in schnellerer Abfolge mit kürzeren Elementen als Tiere des Gombe Stream Nationalparks versehen. Außerdem senden die Tiere der Mahale-Population breitbandigere hochtönigere Klimaxelemente als Schimpansen der anderen Population. Dies zeigt sich in Abbildung 12, die die Lautäußerungen jeweils dreier Tiere einer jeden Gruppe miteinander vergleicht.

Da in dieser Studie durch das natürliche Auftreten der Tiere in ihrem Lebensraum eine genetische Begründung für unterschiedliche Rufvarianten nicht ausgeschlossen werden kann, wird das Erlernen verschiedener Rufe als einer der möglichen Gründe für eine unterschiedliche Ausprägung in Betracht gezogen.

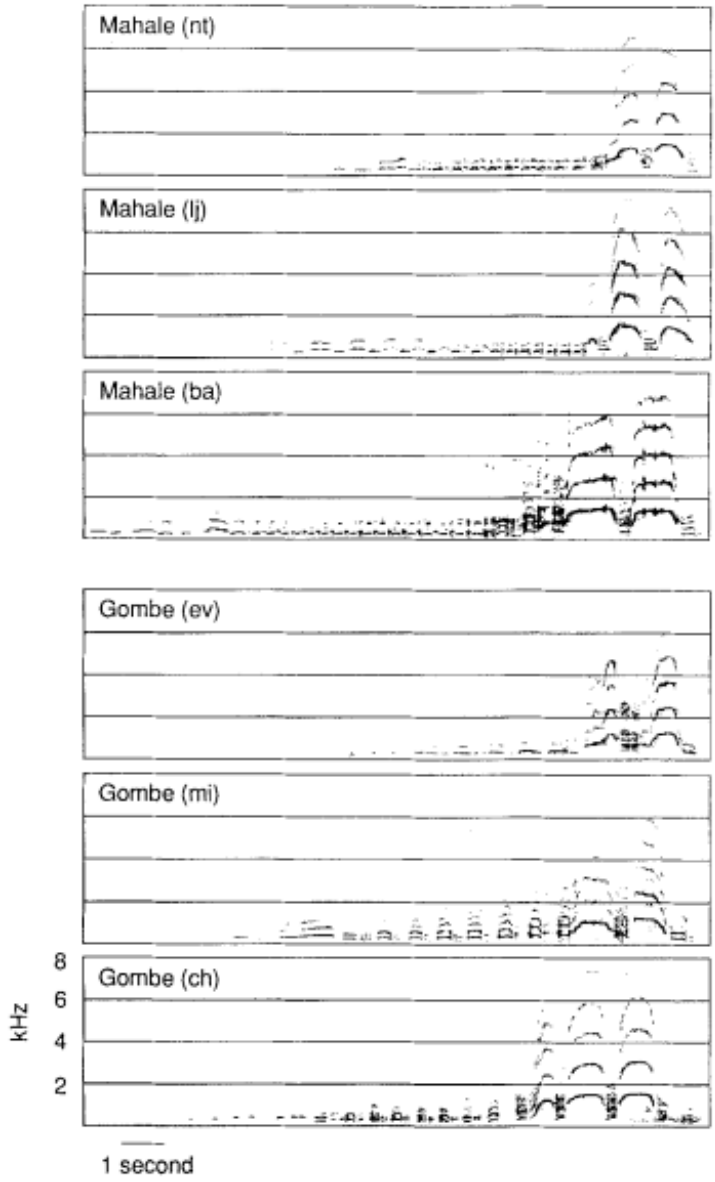


Abb. 12 Vergleich der Pant-Hoot-Rufe beider Populationen. Quelle: [MHGMB92]

In einer kompletteren Untersuchung von Crockford, Herbinger, Vigilant und Boesch, deren Ergebnisse in [CHVB04] präsentiert werden, werden die bisher vermuteten Erkenntnisse untermauert. Sie untersuchen vier Schimpansenpopulationen in der Elfenbeinküste, von denen drei in angrenzenden Gebieten leben, während die vierte in 70 km Entfernung beheimatet ist. Unter Annahme eines erlernten Verhaltens wird also eine Situation, in der die Rufe der Individuen der drei benachbarten Gruppen aus Gründen der eindeutigen Identifizierung mit ihrer Gruppe deutlich unterscheiden, während die Unterschiede der Pant-Hoot-Rufe zwischen den drei beieinander liegenden und der weit

entfernten Population durch den Zufall bedingt sind. Dies wird durch umfassende Untersuchungen bestätigt. Abbildung 13 zeigt die erwartete Struktur der Rufe der Mitglieder der Schimpansenpopulationen, die anhand zweier Funktionen auf Ähnlichkeit untersucht werden.

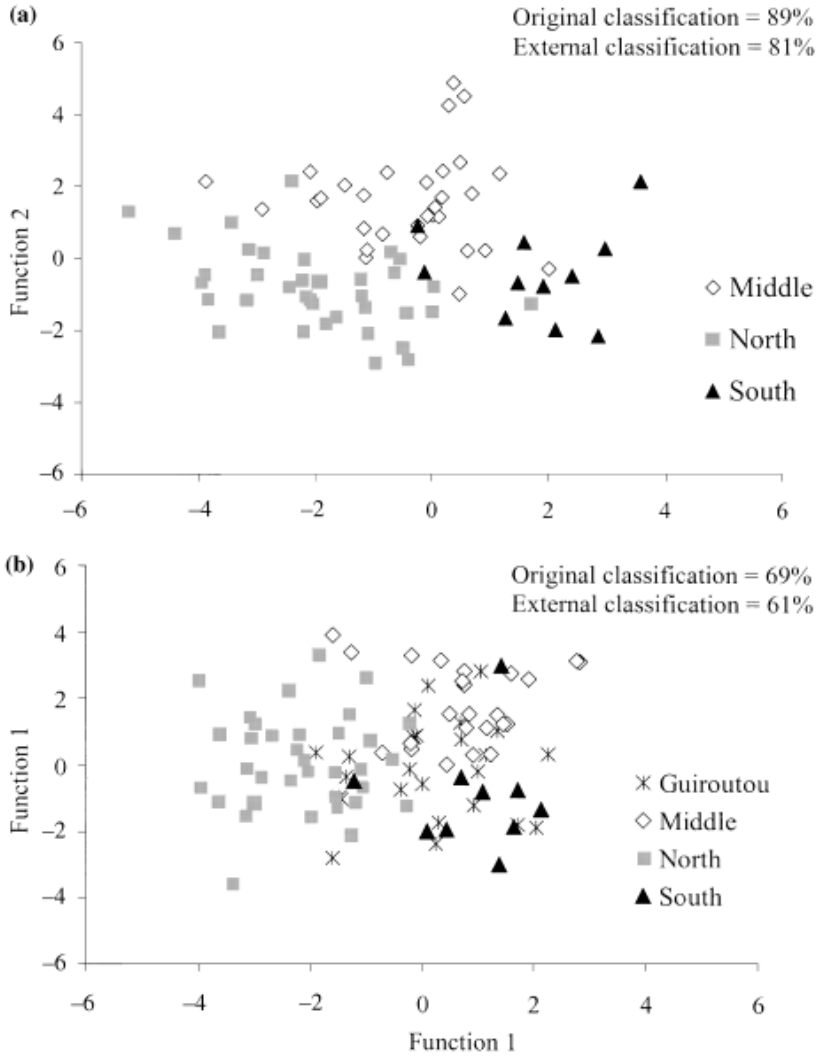


Abb. 13 Vergleich der Pant-Hoot-Rufe der vier Gruppen. Quelle: [CHVB04]

Zusätzlich korrelieren die akustischen Distanzen zwischen den Rufen der einzelnen Individuen nicht mit ihrer genetischen Distanz, die durch DNA-Analysen festgestellt wird. Daraus schließen die Forscher, dass Schimpansen aktiv ihre Lautäußerungen derart modifizieren, dass sie sich von benachbarten Gruppen unterscheiden. Dies untermauert die These, dass die Rufe der Schimpansen einem Lernprozess unterliegen.

2.5.2 Dialekte bei Walen

Eine weitere Tierfamilie mit belegtem dialektbehafteten Erlernen kommunikativer Laute ist die der Wale. Die Weibchen leben mit ihren Jungtieren in Familienverbänden von bis zu einem Dutzend Mitgliedern zusammen und sind daher sowohl für eine Beobachtung des Lernprozesses selbst als auch für eine Analyse möglicher familiärer Dialekte geeignet. Ausgewachsene Männchen ziehen meist alleine durch die Ozeane, was eine familiäre Zuordnung erschwert.

Weilgart und Whitehead zeigen in [WW97] die Ergebnisse ihrer Forschung, in der sie Codas, eine musterbehaftete Abfolge von Klicks, von weiblichen und heranwachsenden Pottwalen, die sie an verschiedenen Orten in der Karibik und im Südpazifik aufnahmen und analysierten. Insgesamt werden 3644 Codas kategorisiert, basierend auf der Anzahl der Klicks und ihrer Abfolge. Insgesamt können sie 30 verschiedene Codas in vier Coda-Gruppen (vgl. Abb. 14) nachweisen, die sie in Coda-Repertoires, bestimmt durch Tag, Walgruppe, Ort, Gegend, Region und Ozean einteilen.

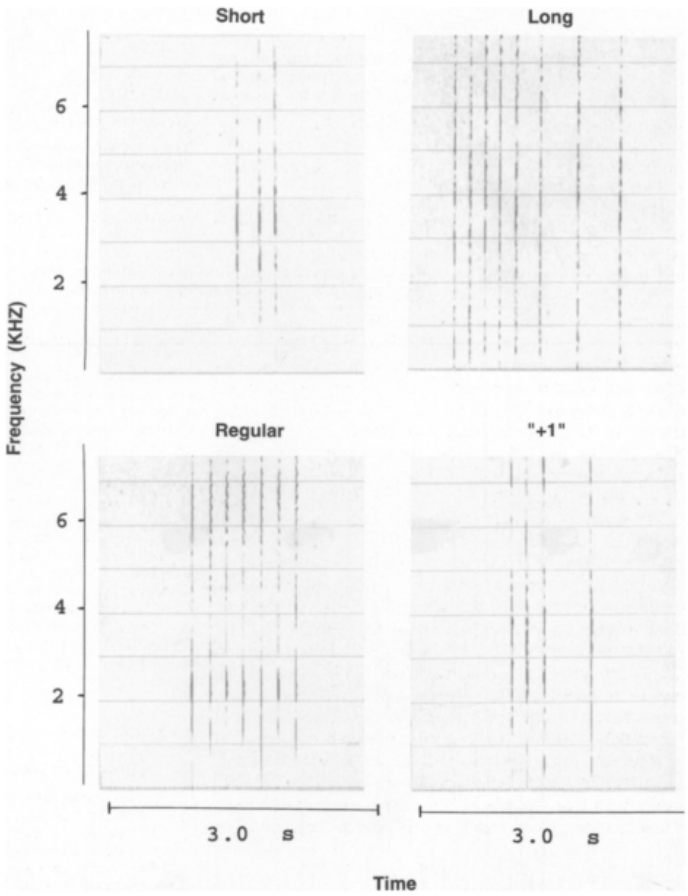


Abb. 14 Die vier Coda-Gruppen Quelle: [WW97]

Diese Bedeutungsmengen werden anhand einer Funktion r_s auf Ähnlichkeit untersucht. Dabei werden starke gruppenspezifische Dialekte nachgewiesen, die über einen Zeitraum von mehreren Jahren anhalten. Zusätzlich zu diesen existiert eine schwächere geografische dialektale Ausprägung. Die Unterschiede im Repertoire der durch den amerikanischen Kontinent getrennten Gruppen in der Karibik und im Pazifik sind signifikant. Abbildung 15 zeigt die relativen Häufigkeiten der genutzten Coda-Repertoires.

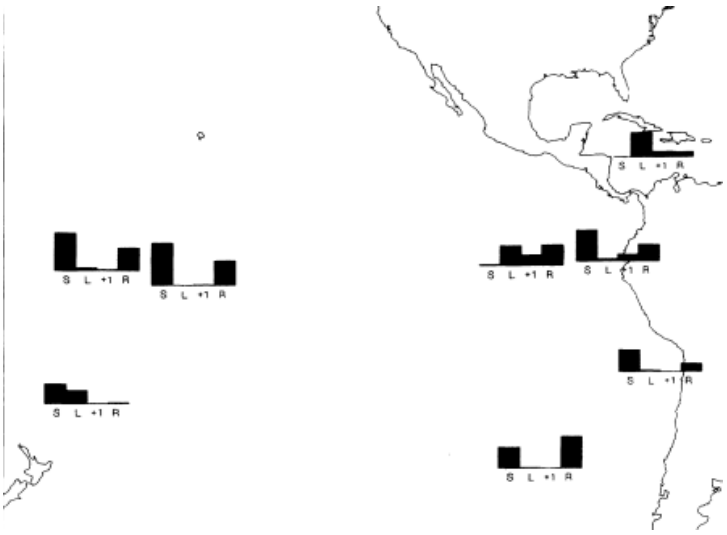


Abb. 15 Relative Häufigkeiten der Coda-Repertoires Quelle: [WW97]

Coda-Repertoires einer selben Gruppe, die an verschiedenen Tagen untersucht werden, erreichen einen Ähnlichkeitswert von r_s = 0.5-0.85 und sind sich damit viel ähnlicher als Repertoires anderer Gruppen am gleichen Ort (r_s = 0.15), oder aber anderer Gruppen an anderen Orten in der gleichen Gegend (r_s = 0.02) oder anderer Gruppen in anderen Regionen (r_s = 0.0) oder gar verschiedener Ozeane (r_s = 0.0). (vgl. Abb. 16)

Pottwale sind damit neben Killerwalen die einzige Walspezies, bei der Dialekte, also Unterschiede im Vokalrepertoire zwischen benachbarten, möglicherweise interagierenden Gruppen nachgewiesen worden sind.

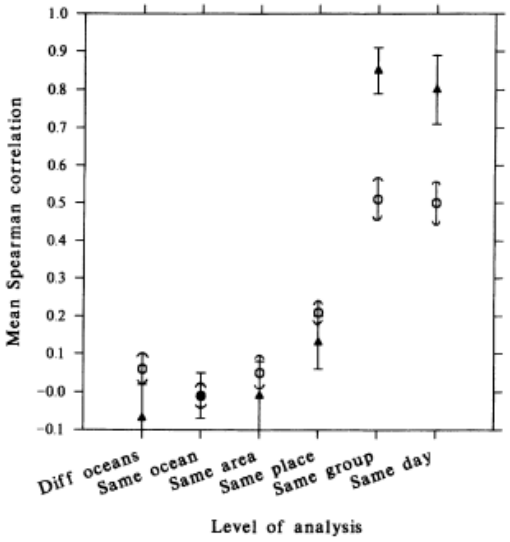


Abb. 16 Ähnlichkeit der Coda-Repertoires Quelle: [WW97]

Ford, Deecke und Spong präsentieren in [FDS00] die Ergebnisse einer Analyse zweier verschiedener Rufe der Killerwale innerhalb zweier matrilinearer sozialer Gruppen. Die Rufe werden anhand einer Funktion auf Ähnlichkeit untersucht. Abbildung 17 zeigt einen Ruf mit den zur Klassifizierung untersuchten Parametern.

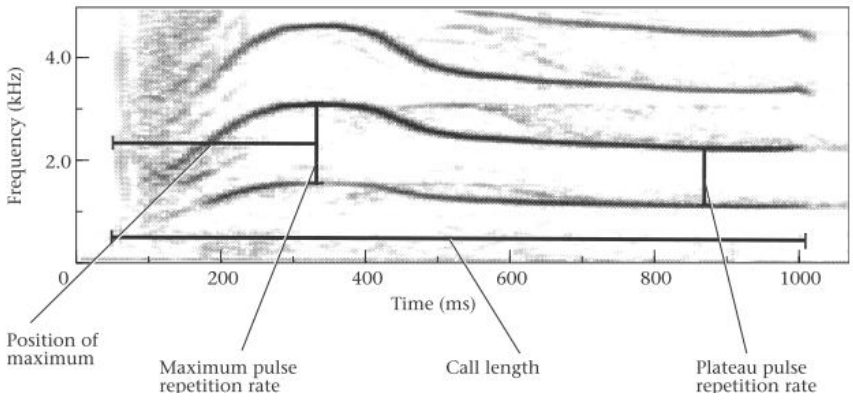


Abb. 17 Untersuchte Parameter eines Orca-Rufs Quelle: [FDS00]

Ein Test auf strukturelle Modifikation weist bei einem der beiden Rufe signifikante Veränderungen in einer der beiden Gruppen nach, jedoch nicht in der anderen. Die Stärke der Divergenz des modifizierten Rufs zwischen beiden Gruppen ist signifikant niedriger als die Stärke der Modifizierung innerhalb einer Gruppe, was zeigt, dass die Rufe in ähnlicher Weise in beiden Gruppen modifiziert

werden. Abbildung 18 zeigt die Stärke der Ruf-Modifizierung beider Gruppen und beider Rufe im Laufe der Zeit und einen Vergleich der jeweiligen Rufe zwischen beiden Gruppen.

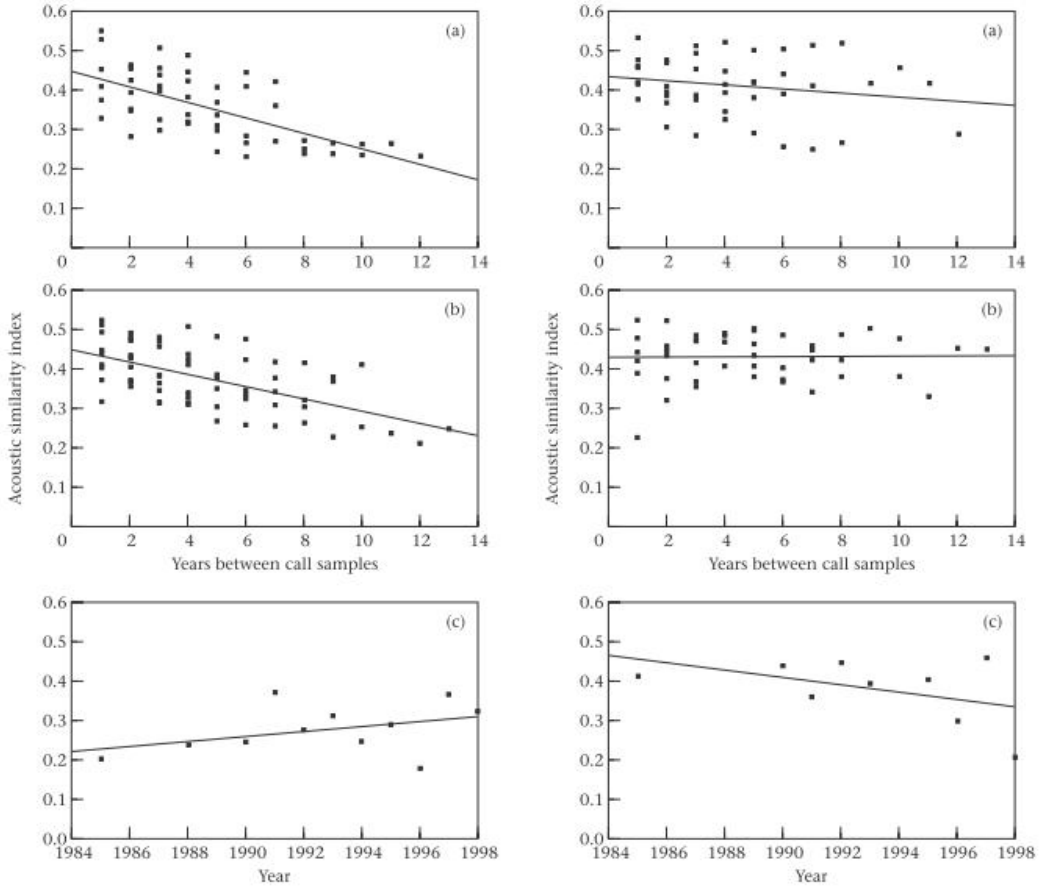


Figure 2. The rate of call modification for the N4 call type of (a) the A12 matriline, 1986–1998, and (b) the A30 matriline 1985–1998 and (c) the rate of acoustic divergence between the two groups.

Figure 3. The rate of call modification for the N9 call type of (a) the A12 matriline, 1986–1998 and (b) the A30 matriline, 1985–1998 and (c) the rate of acoustic divergence between the two groups.

Abb. 18 Rufmodifizierungen beider Rufe in beiden Gruppen Quelle: [FDS00]

Eine Analyse der strukturellen Parameter zeigt keine starke Richtung innerhalb dieses Wandels. Er könnte durch das Altern der Wale oder, falls Orca-Dialekte verhaltensbedingte Eigenschaften sind, durch kulturellen Drift innerhalb der Rufe, zusammen mit horizontalem Austausch der Modifizierungen zwischen beiden Gruppen ausgelöst worden sein.

Solch ein „Matching“ zwischen Mitgliedern verschiedener Matrilinien würde es wahrscheinlich erscheinen lassen, dass das Erlernen der Rufe nicht alleine durch vertikale Überragung von Mutter zu Nachkommen beschränkt ist und somit rein genetische Modelle des Ruf-Erlernens widerlegen.

2.5.3 Dialekte im Vogelgesang

Die Gesänge männlicher Singvögel sind ebenso als Beispiele erlernter Kommunikation anzuführen, da sie nur dann ihre typischen Gesänge entwickeln, wenn sie anderen Singvögeln zuhören können. Marler zeigt in [Mar70], dass ein in Isolation gehaltener Singvogel einen weit simpleren Gesang ausbildet. Dennoch bauen Vogelgesänge nicht auf einer Zuordnung zwischen Gesang und kommunikativer Funktion auf. Sie dienen lediglich dazu, die Aufmerksamkeit der Weibchen zu erlangen und territoriale Besitzansprüche der Männchen zu verdeutlichen. Bei den meisten Spezies ist ein bestimmtes Gesangsmuster angeboren (vgl. [Kon65]), dass den Schluss zulässt, dass für Singvögel „die Fähigkeit einen Ton zu imitieren genau so reflexiv und kognitiv unkompliziert sein könne wie die Fähigkeit zu atmen“ (Zitat [WK96], S.172)

Im Vogelgesang zeigt sich ebenfalls eine Ausbildung regionaler Dialekte innerhalb einer Art. In dieser Arbeit wird der Fokus dabei auf die am genauesten untersuchte Gattung der Ammern gelegt, im Speziellen auf die Dachs- und die Morgenammer.

Tubaro und Segura zeigen in [TS94], dass sich die Gesänge der Morgenammer in der Villarino Region in der Provinz Buenos Aires in Argentinien je nach Habitatsstruktur stark unterscheiden. Die Gesänge im offenen Grasland zeichnen sich durch längere Trill-Intervalle, höhere Trill-Bandbreiten und minimale Trill-Frequenzen aus als die Gesänge der Tiere, die in den bewaldeten Gebieten leben. Abbildung 19 zeigt einen Graphen, der die Hauptkomponente des Gesangs mit der Länge des Trill-Intervalls in Relation stellt.

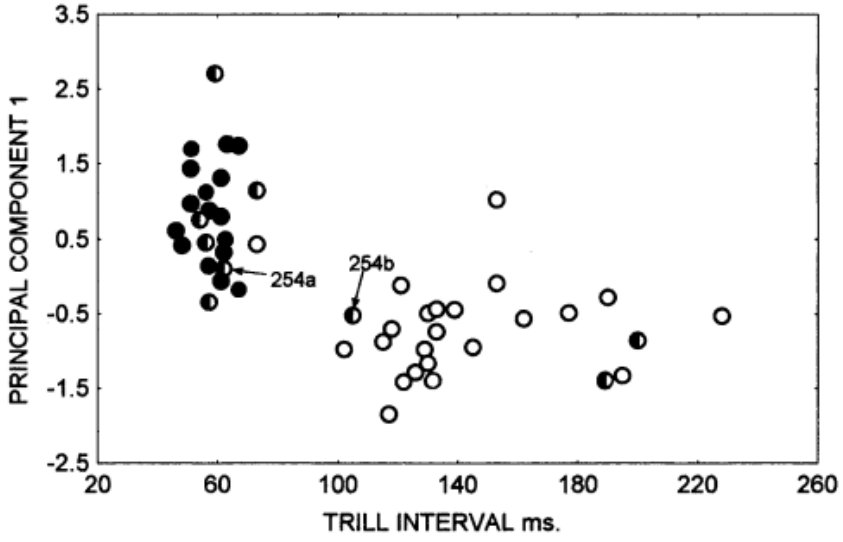


Abb. 19 Hauptkomponente vs. Trill-Intervall Quelle: [TS94]

In einem schmalen Streifen im Übergang von Grasland zu Waldgebiet beherrschen die Vögel entweder den einen oder den anderen der beiden Dialekte. Wenige Exemplare beherrschen gar beide Gesänge, beispielsweise Vogel 254. (vgl. Abb. 19, 254a und 254b)

Marler und Tamura veröffentlichen in [MT62] Untersuchungen zur Dialektbildung bei der Dachsammer anhand von drei verschiedenen Populationen im Bundesstaat Kalifornien der USA. Zwei davon befinden sich in relativer Nähe zueinander, während die dritte weiter entfernt liegt. Die Analyse der Gesangsmuster zeigt einen ungewöhnlich hohen Grad an Homogenität innerhalb einer jeden Population, während die gruppenexternen Vergleiche eine negative Korrelation zwischen Ähnlichkeit und geografischer Nähe nachweisen. Abbildung 20 zeigt den grafischen Vergleich des Vogelgesangs von acht Exemplaren der Dachsammer bei Inspiration Point, Contra Costa County im Gegensatz zu dem Gesang von Exemplaren in Berkeley, Alameda County.

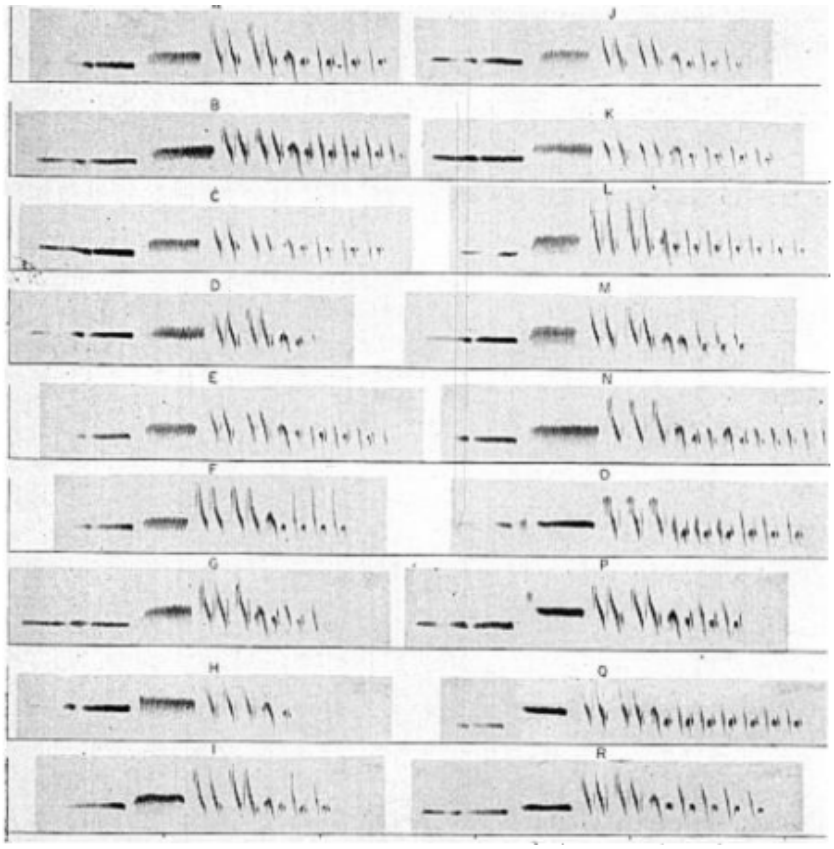


Abb. 20 Grafischer Vergleich des Gesangs der Dachsammer Quelle: [MT62]

2.5.4 Dialekte sonstiger Arten

Neben den bereits genannten Beispielen von Dialekten in Tiersprachen existieren noch weitere Arten, bei welchen ebenfalls das Vorkommen von dialektalen Ausprägungen nachgewiesen worden ist. Slobodchikoff, Ackers und van Ert in [SAE98] und Perla und Slobodchikoff in [PS02] weisen Dialekte innerhalb der Alarmrufe der Präriehunde in verschiedenen US-amerikanischen Bundesstaaten nach. Obwohl sich die Alarmrufe einer Kolonie im Laufe der Jahre verändern, bleiben sie gruppenintern stets ähnlich zueinander. Der gruppenexterne Vergleich der Alarmrufe belegt einen eindeutigen Zusammenhang zwischen geographischer und dialektaler Distanz. (vgl. Abb. 21)

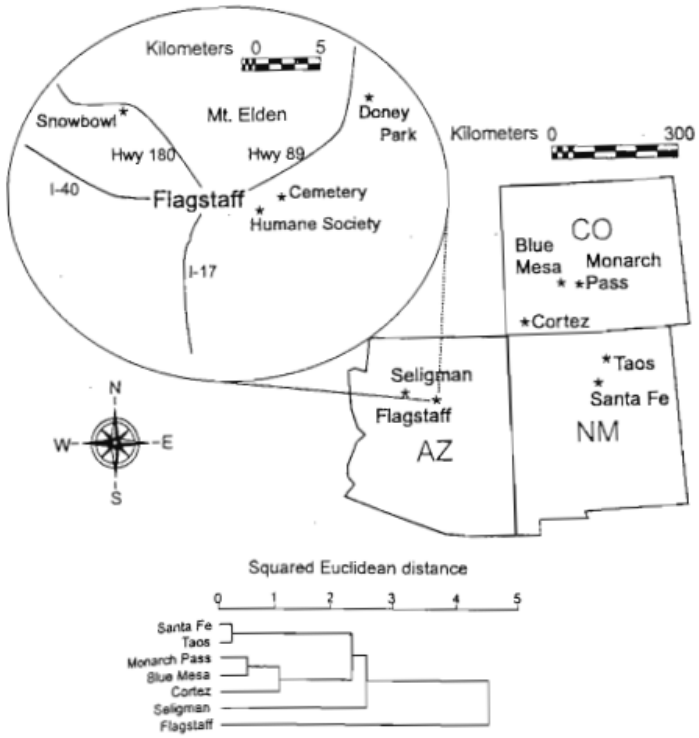


Abb. 21 Geografische und Dialektale Distanz Quelle: [SAE98]

Nevo, Heth, Beiles und Frankenberg weisen in [NHBF87] unterschiedliche Dialekte innerhalb der männlichen Balzrufe von Nacktmullen in Israel nach. Weibchen einer der vier chromosomal verschiedenen Spezies bevorzugen den männlichen Balzruf von Individuen der eigenen Art. Die hier genannten Dialekte jedoch scheinen daher weniger erlernt als durch genetische Unterschiede der Individuen bedingt und sind daher für diese Arbeit von untergeordneter Bedeutung.

3. Entwicklung des Simulationsmodells

In diesem Kapitel werden die einzelnen Schritte der Entwicklung des Simulationsmodells angeführt. Zuerst werden die Begriffe *Simulation* und *Modell* im Sinne der Informatik erklärt bevor die in dieser Arbeit verwendete Art der Multi-Agenten-Simulation dargestellt wird. Die mathematischen Grundlagen der neuronalen Netze werden im Anschluss daran angeführt. Nach einer Übersicht bereits vorhandener Modelle der Sprachentwicklung schließt dieses Kapitel mit einer detaillierten Erläuterung des Simulationsmodells dieser Arbeit. Aufgrund der thematischen Überschneidung finden sich die ersten drei Unterkapitel dieses Kapitels in leicht abgewandelter Form ebenfalls in [FK07] wieder.

3.1 Simulation und Modell

Dieses Unterkapitel orientiert sich an den Ausführungen von K. G. Troitzsch und N. Gilbert in [TG05] S. 15-27. Simulation ist eine Vorgehensweise zur Analyse dynamischer Phänomene innerhalb real existierender Systeme, welche *Zielsysteme* genannt werden. Da ein Zielsystem in der Realität generell sehr komplex ist und dadurch eine computergestützte Simulation in gleicher Komplexität kaum in der Lage wäre, zukunftsbezogene Daten zu liefern, wird das Zielsystem vereinfacht, es wird ein Modell von ihm erstellt. Das Bestreben dieser Modellierung ist es, ein möglichst einfaches Modell zu kreieren, das sowohl komplex genug ist, alle in der Realität für einen Sachverhalt relevanten Faktoren abzubilden, als auch kompakt genug ist, um das Modell mit einem vertretbaren Zeitaufwand simulieren zu lassen. Simulation bedeutet in diesem Sinne ein temporäres Fortschreiten des Modells, in dem sich die Parameter der Bestandteile des Modells, beispielsweise Agenten und ihre Umgebung, anhand stochastischer Methoden verändern. Das Ziel der Simulationstechnik ist, dass die Phänomene, die in einer zukunftsgerichteten Simulation des Modells eintreten, ebenfalls in der Realität, also im Zielsystem, in der Zukunft zu beobachten sein werden.

Innerhalb der Sozialwissenschaften ist das Zielsystem stets ein dynamisches Gebilde, bestehend aus einer *Struktur* und einem *Verhalten*. Dadurch bedingt muss das Modell ebenfalls dynamisch sein, es muss sich mit der Zeit verändern können. Wie aber kann abgeschätzt werden, welche Parameter des Zielsystems mit in ein

Modell übernommen werden müssen und auf welche verzichtet werden kann? Wie aber kann ein Modell „validiert[6]“ werden?

Die wichtigste Voraussetzung für ein gelungenes Modell ist die Existenz zuverlässiger und präziser Daten des zu simulierenden Systems über eine möglichst lange Zeitspanne. Denn dann besteht die Möglichkeit, mit dem abstrahierten Modell genau die Periode ex post zu simulieren, von der bereits Daten des Endzustands vorhanden sind. Ähneln die durch die Simulation erhaltenen Daten den erhobenen Daten der realen Welt, hat das Modell seine Ähnlichkeit für den gegebenen Zeitraum belegt.

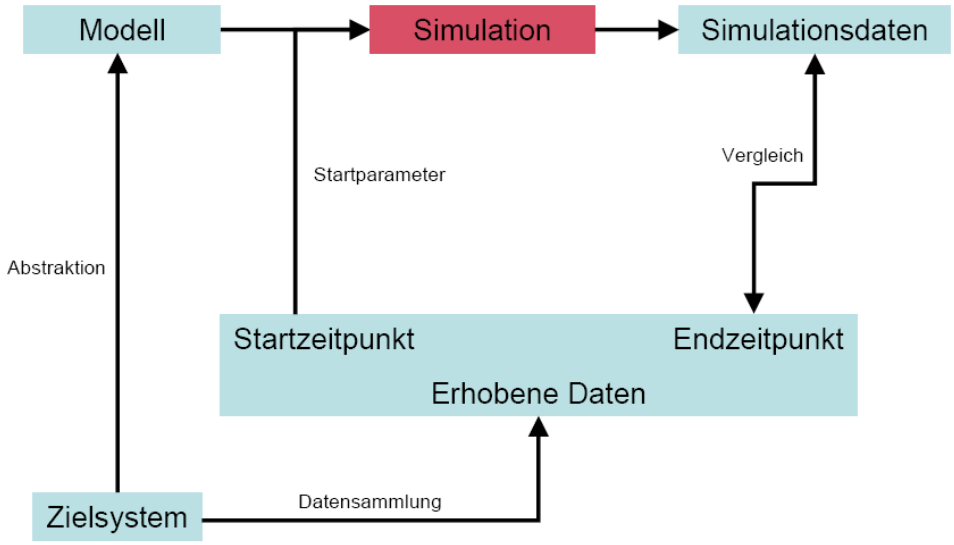


Abb. 22 Verifikation von Simulationsmodellen
Quelle: Neuzeichnung [TG05, S. 17, Abb. 2.2]

Der soeben beschriebene Effekt hat keine Aussagefähigkeit über eine zukunftsbezogene Ähnlichkeit zwischen Zielsystem und Modell, jedoch ist im Allgemeinen davon auszugehen, dass Modelle, die über einen längeren Simulationszeitraum ihre Zuverlässigkeit bewiesen haben, auch eine verlässliche Prognose des Verhaltens des realen Systems in zumindest nahe gelegener Zukunft erstellen können.

Die Simulationen dieser Modelle enden demnach nicht mit dem Endzeitpunkt der verfügbaren erhobenen Daten, sondern sie simulieren die Zukunft. Dies wird in Abbildung 23 verdeutlicht. Die aktuellsten erhobenen Daten sind demnach der

[6] Validierung ist hier im Sinne der externen Validierung, der analytischen Beweisführung, ob eine konsistente Beziehung zwischen den Ergebnissen des Software-Maßes und den empirisch verfügbaren, externen Attributen besteht (vgl. [EB96], S. 260), zu sehen.

Startdatensatz der zukunftsgerichteten Simulation. Die in der Simulation auftretenden zukünftigen Phänomene geben eine Idee dafür, was im Zielsystem in Zukunft geschehen kann. Es können also Rückschlüsse auf zukünftiges Verhalten der modellierten Objekte gezogen werden.

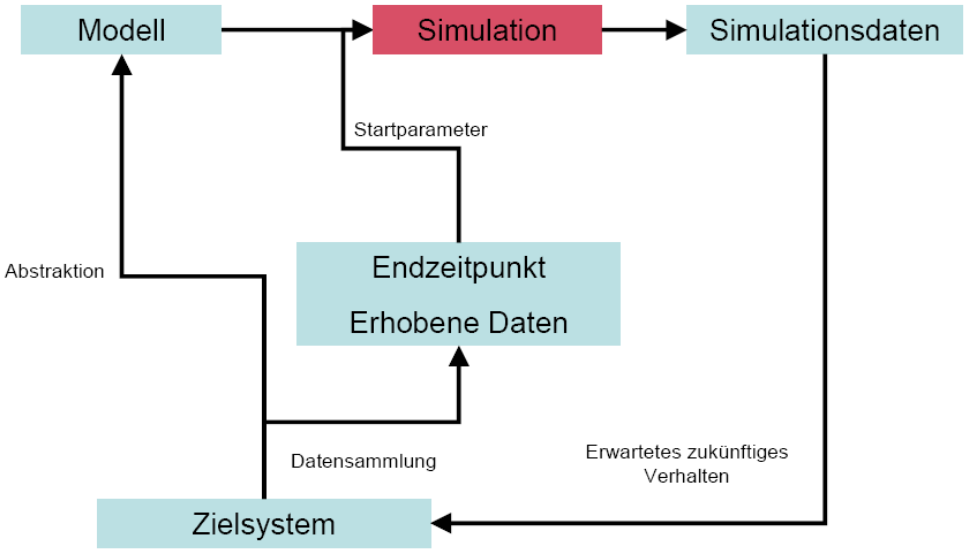


Abb. 23 Rückschlüsse durch eine Simulation

Eine Simulation der Entwicklung der Bevölkerungsstruktur ist ein triviales Beispiel, um den Nutzen einer solchen Simulation zu verdeutlichen. Sie erlaubt es, ex ante eine zukünftige Gefahr der Überalterung einer Gesellschaft zu zeigen, und anhand veränderter Parameter innerhalb der Simulation lassen sich ebenfalls mögliche Auswege finden.

Die hier beschriebenen Grundsätze sind auf das in dieser Arbeit erstellte Modell nur bedingt übertragbar, da es keine zuverlässigen erhobenen Daten über den weit in die Vergangenheit zurückliegenden Prozess der Sprachentstehung gibt. Versuche, diesen Prozess an realen Menschen wieder zu erleben, sind in der heutigen Zeit ethisch nicht vertretbar und wurden beispielsweise von Friedrich II[7] durchgeführt, der eine Gruppe Säuglinge von Ammen säugen ließ, denen es jedoch verboten war, mit den Kindern zu sprechen oder ihnen körperliche Zuwendung zukommen zu lassen. Das Ziel dieses Versuchs bestand darin, den Entstehungsprozess der „menschlichen Ursprache“ zu rekonstruieren, die die Säuglinge unter sich ausbilden sollten. Er scheiterte, da die Neugeborenen aufgrund mangelnder Zuwendung und menschlicher Nähe zu Grunde gingen. (vgl. [Sal13])

[7] Friedrich II (1194 – 1250), römisch-deutscher Kaiser von 1220 bis 1250.

3.2 Multi-Agenten-Simulation

Dieses Unterkapitel widmet sich speziell der Technik der Multi-Agenten-Simulation, einem mächtigen Simulationskonzept, welches die Möglichkeit bietet, komplexe Verhaltensweisen von Individuen und ihre Interaktion sowohl miteinander als auch mit ihrer Umwelt zu simulieren. Im Folgenden werden diese beiden Bestandteile der Multi-Agenten-Modelle genauer erklärt. Der Unterpunkt Agent orientiert sich dabei an den Erläuterungen von K. G. Troitzsch und N. Gilbert in [TG05, S. 172-198].

3.2.1 Agent

Während es noch keine allgemein gültige Definition für einen Software-Agenten gibt, wird der Begriff gewöhnlich dazu benutzt, ein Programm zu beschreiben, das seine eigenen, auf seiner Sichtweise der Operationsumgebung basierenden Aktionen kontrollieren kann. (siehe [HS98])

Das Ziel ist es, einen Software-Agenten zu kreieren, der intelligent mit seiner Umwelt interagiert. Das Konzept eines Software-Agenten wird oft dazu benutzt, eine zielgerichtete Aufgabe eigenständig zu erfüllen. Das Vorbild für eine solche Struktur eines Programms ist der Mensch selbst. So sollen auch Agenten Konzepte wie einen freien Willen und eine Absicht, seine eigenen Ziele zu verwirklichen, besitzen.

Eigenschaften

Software-Agenten besitzen folgende Eigenschaften (Woolbridge und Jennings 1995, in Analogie zu [TG05, S. 173]):

- **Eigenständigkeit**
  Agenten agieren, ohne dass andere direkte Kontrolle über ihre Aktionen oder inneren Zustände haben.

- **Sozialvermögen**
  Agenten interagieren mit anderen Agenten durch eine Sprache.

- **Reaktionsfähigkeit**
  Agenten sind in der Lage ihre Umwelt zu erkennen und Schlüsse daraus zu ziehen.

- **Eigeninitiative**
  Agenten können eine Eigeninitiative ergreifen, indem sie ein zielgerichtetes Verhalten ausüben.

Simulationsattribute

Die in diesem Modell benutzten Agenten sollen die folgenden beiden Simulationsattribute ausbilden können, um einerseits ihr antrainiertes Wissen speichern zu können und andererseits mit anderen Agenten kommunizieren zu können.

Wissensrepräsentation
Die Agenten benötigen eine Struktur, um das Wissen und die subjektiven Annahmen der Welt in einer geeigneten Weise zu speichern. Dazu gibt es drei vorherrschende Ansätze:

- die Speicherung in prädikatenlogischen deklarativen Aussagen;
- das Halten in semantischen baumartigen Strukturen, in der die allgemeinen Fakten an der Wurzel liegen und die Spezifizierung der Information mit der Baumtiefe zunimmt;
- die Wissensrepräsentation durch neuronale Netze. Diese Variante ist im hier behandelten Modell vorgesehen und daher wird die Funktionsweise der neuronalen Netze in Unterkapitel 3.3 genauer erklärt.

Sprache
Alle Multi-Agenten-Modelle beinhalten eine Form von Interaktion zwischen Agenten oder mindestens eine indirekte Interaktion, indem Agenten nur mit der Umwelt interagieren und andere Agenten eine veränderte Umwelt antreffen. Es existieren vielfältige Interaktionsmöglichkeiten, von einem einfachen Austausch von Informationen bis hin zu einem komplexen Aushandeln von Verträgen. Solche Interaktionen können entweder über eine, eventuell fehleranfällige, formale Sprache bewerkstelligt werden oder aber durch den direkten Austausch von Informationen von „Gehirn zu Gehirn“. Die Auswahl einer der beiden Kommunikationsarten ist abhängig vom Modell. Es kann durchaus gewünscht sein, eine fehleranfällige Kommunikationsform zu benutzen, beispielsweise um die Auswirkungen von Sprachveränderungen zu simulieren. Im hier behandelten Modell ist daher eine direkte Interaktion zwischen zwei Individuen durch Sprache vorgesehen. Die von den neuronalen Netzen der Agenten berechneten Wörter einer Sprache werden, nachdem sie durch mehrere Störfaktoren verzerrt wurden, den Kommunikationspartnern zur Verfügung gestellt. Die Existenz einer fehleranfälligen Art der Interagentenkommunikation scheint für eine

Dialektausbildung unabdingbar zu sein. Einzelheiten hierzu befinden sich im Abschnitt 3.5.5 und in den Simulationsdurchführungen in Kapitel 4.

3.2.2 Umwelt

Die Simulationsumgebung ist das Universum, in dem die Agenten existieren. Es beinhaltet sowohl eine räumliche als auch eine zeitliche Komponente.

Raum

Räumlich gibt es die Möglichkeit, die Agenten sich in einer gitternetzartigen Simulationswelt bewegen zu lassen. Dieses Verfahren bildet eine geografische Umgebung ab und so ist es möglich, eine Umwelt zu kreieren, in der an bestimmten Koordinaten bestimmte Ereignisse eintreten, in gewissen Regionen andere Umstände herrschen als in anderen Regionen. Hierbei ist eine starke Ähnlichkeit zum zellulären Automaten erkennbar, jedoch ist der Agentenaufbau den weniger komplexen Zellen überlegen.

Ein einfaches Beispiel ist eine Landkarte, die aus Gitterzellen besteht, die entweder Erdboden oder aber Wasser repräsentieren. So ist es denkbar, ein einfaches Modell eines Landesabschnitts mit Flüssen und Seen nachzubilden. Es ist leicht ersichtlich, dass, sobald ein Agent auf eine Wasserzelle stößt, andere Aktionen möglich sind als bei einer Zelle aus festem Untergrund. Ein triviales Beispiel ist die Entscheidung der Fortbewegungsart, ob der Agent in diesem Fall besser geht oder schwimmt.

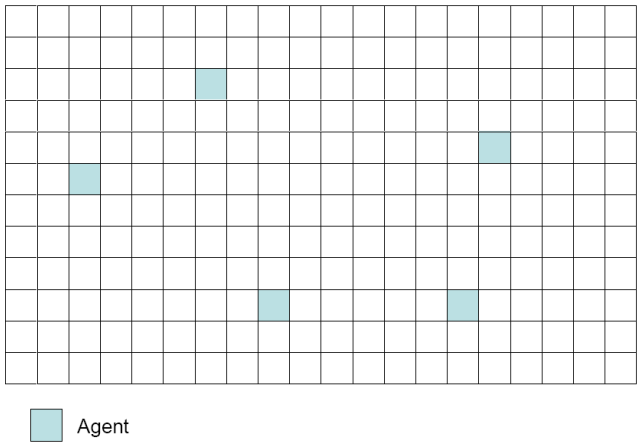


Abb. 24 Agenten in Gitternetzumgebung

Eine weitere Möglichkeit der Darstellung eines Agentennetzwerks ist die eines gewichteten Graphen. Die Agenten als Knoten des Graphen sind durch Kanten

miteinander vernetzt. Die Entfernung zwischen den Agenten kann in diesem Falle durch ein Kantenattribut repräsentiert werden. Der Vorteil dieser Darstellungsmöglichkeit liegt darin, dass Graphen nicht nur zur räumlichen Entfernungsangabe benutzt werden können, sondern auch andere Attribute beherbergen können, beispielsweise gegenseitige Sympathie oder aber Wahrscheinlichkeiten der Kommunikation mit einem anderen Agenten.

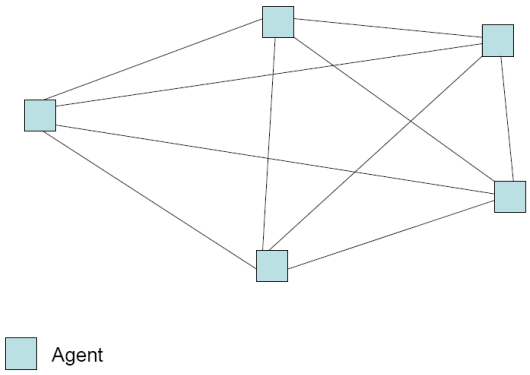


Abb. 25 Agenten verbunden durch Graphen

Die beiden hier genannten Möglichkeiten sind ebenfalls kombinierbar. Es ist oft von Vorteil, beide Topologien in einem einzigen Modell zu implementieren, beispielsweise in einer Simulationswelt, in der sich Agenten in einem Raum fortbewegen können und in dem jeder Agent eine gewisse Sympathie den anderen Agenten entgegenbringt. Während der Raum durch ein Gitternetz repräsentiert wird, lässt sich die Sympathie durch einen gerichteten Graphen darstellen. In diesem Modell ist es fortan möglich, diese beiden Distanzen zu verknüpfen, indem die Wahrscheinlichkeit, dass eine Kommunikation zwischen zwei Agenten stattfindet, die sowohl von der räumlichen Entfernung im Gitternetzmodell als auch von der gegenseitigen Sympathie zweier Agenten abhängt.

Nachdem Agenten in einer Umgebung positioniert worden sind, benötigen sie die Fähigkeit ihre Umgebung wahrzunehmen und zu verändern. Diese „Sinneswahrnehmungen“ werden beispielsweise für eine Inter-Agenten-Kommunikation benötigt, sodass Agenten die Fähigkeit zu *sprechen* und zu *hören* besitzen müssen. Das Hören einer Äußerung eines anderen Agenten entspricht in diesem Falle der Wahrnehmung, das Sprechen dem Verändern der Umwelt.

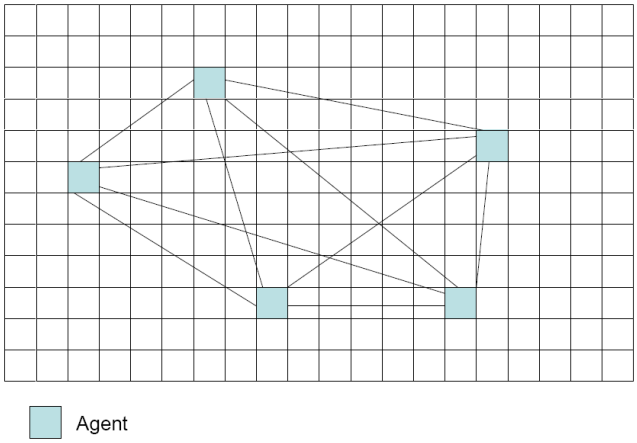


Abb. 26 Agenten in einer kombinierten Umgebung

Zeit

Es gibt zwei Möglichkeiten, den Ablauf der Zeit in einer Simulation nachzubilden. In einer *kontinuierlichen Simulation* schreitet die Zeit iterativ voran. Innerhalb eines Zeitschritts arbeiten alle Agenten idealerweise parallel. Da dies in der Praxis, aufgrund sequenziell arbeitender Computer, nicht ohne weiteres möglich ist, muss die Parallelität simuliert werden, im einfachsten Falle durch ein Rundlaufverfahren, in dem der nächste Agent zufällig ausgesucht wird. Da diese Simulationstechnik keine echte Parallelität gewährleistet, sondern die zufällige Reihenfolge der Agentenwahl den Ablauf der Simulation erheblich beeinflusst, wird das Modell im Falle einer parallel ablaufenden Simulation um einen Nachrichten- und Aktionspuffer in der Umgebung erweitert. Dort werden alle in einer Runde ausgesendeten Nachrichten und durchgeführten Aktionen der Agenten zwischengespeichert und den Agenten gänzlich im nächsten Simulationsschritt geliefert.

Eine zweite Möglichkeit ist die *diskrete Simulation* von Zeit. Hier arbeiten die Agenten ähnlich, sie werden jedoch nicht in jedem Simulationsschritt angesprochen. Vielmehr senden sie, nachdem sie eine Aktion ausgeführt haben, eine Nachricht an den Simulationszeitgeber, die die Dauer der aktuell ausgeführten Aktion beinhaltet und aus der sich somit der Zeitpunkt des nächsten Aufrufs dieses Agenten errechnen lässt.

3.3 Neuronale Netze

Im folgenden Unterkapitel wird eine Einführung bis hin zur detaillierten Beschreibung von Lernalgorithmen künstlicher neuronaler Netze gegeben. Die

Idee, vereinfachte mathematische Modelle, die den Nervenzellvernetzungen im menschlichen Gehirn ähneln, zu entwickeln (vgl. [Rum94], S. 87), ist ein Zweig der künstlichen Intelligenz und Forschungsgegenstand der Neuroinformatik[8]. Neben der hohen Fehlertoleranz und Parallelität ist die Lernfähigkeit künstlicher neuronaler Netze ihr größter Vorteil, da sie unterschiedliche Aufgaben anhand von Trainingsbeispielen erlernen. Innerhalb des Modells dieser Arbeit werden Agenten jeweils mit einem künstlichen neuronalen Netz ausgestattet, sodass eine genauere Betrachtung unabdingbar ist.

3.3.1 Aufbau und Bestandteile

Künstliche neuronale Netze bestehen aus zwei verschiedenen Mengen: einer Menge *N* von stark idealisierten *Neuronen*, die als Basisprozesseinheit gesehen werden können, und einer Menge *V* von *Verbindungen*. Die Struktur der Netze ist die eines gerichteten Graphen, wobei die Neuronen als Knoten und die Verbindungen als Kanten dienen. Jeder Knoten (Neuron) besitzt eine beliebige Menge ein- und ausgehender Kanten (Verbindungen) und einen *Aktivierungswert*, welcher im Allgemeinen ein reeller Wert zwischen 0.0 und 1.0 ist. Alle Verbindungen sind mit einer *Gewichtung*, einem beliebig wählbaren reellen Zahlenwert, versehen (vgl. [Lip05], S. 51).

Jedes künstliche neuronale Netz soll in einer vorgegebenen Umwelt mit Hilfe von *Lernalgorithmen* und *Trainingsdurchläufen* durch Verändern der eigenen Verbindungsgewichtungen ein vordefiniertes Verhalten zeigen, welches als die *Funktion* des Netzes bezeichnet wird.

Bei den für eine Implementierung des Modells dieser Arbeit benötigten künstlichen neuronalen Netzen handelt es sich um *Autoassociator Netze*, welche aus drei verschiedenen Schichttypen bestehen: einer *Eingabeschicht*, beliebig vieler *verborgener Schichten* und einer *Ausgabeschicht*. Die Funktion dieses Netzes ist es, ein von der Umwelt vorgegebenes Muster (in Form eines Vektors reeller Zahlen) nach Einlesen in der Eingabeschicht in der Ausgabeschicht exakt wiederzugeben. Abbildung 27 zeigt den Aufbau eines solchen Autoassociator Netzes, für das gelten muss:

$$x_1 = o_1, x_2 = o_2, \ldots, x_i = o_i .$$

[8] Die Neuroinformatik, als Teil der Informatik, nimmt sich als Ziel, die führenden Prinzipien der Lösung von Problemen durch Gehirne auf Computersysteme zu übertragen (vgl. [Rus90], S. 3).

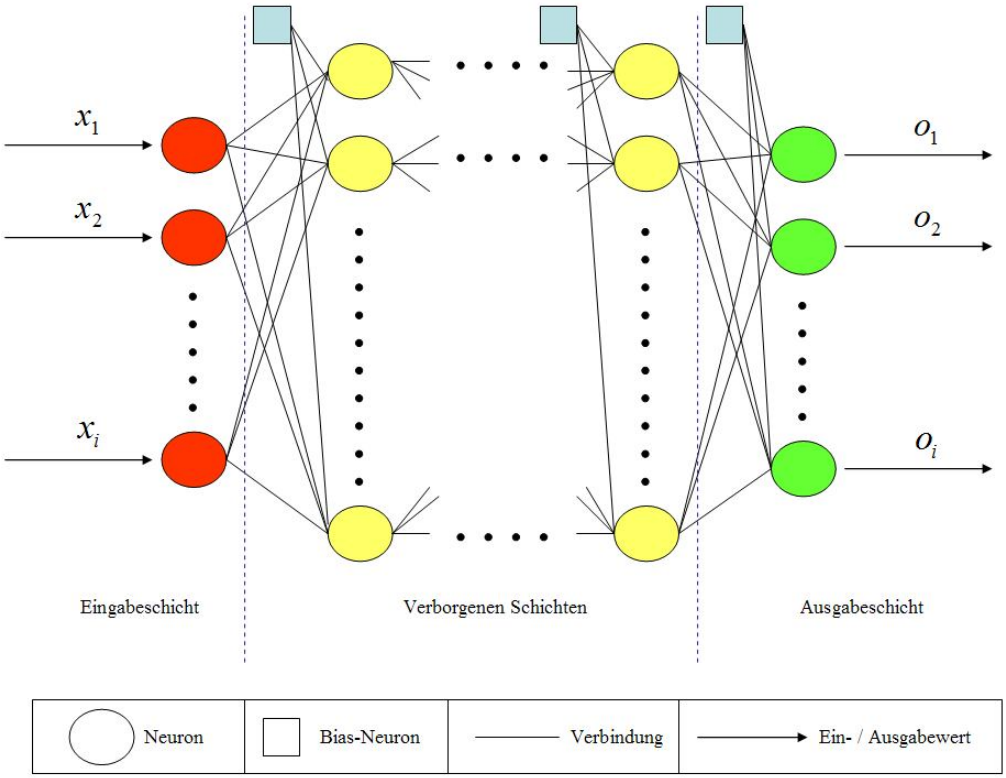


Abb. 27 Aufbau neuronales Netz
Quelle: Neuzeichnung in Anlehnung an [Lip05], S. 51, Abb. 2.8

Dabei werden die Eigenschaften von *Feedforward*[9] Netzen genutzt, um Aktivierungen innerhalb des neuronalen Netzes zu propagieren. Die Ausgangslage ist dabei ein untrainiertes Netz, dessen Verbindungsgewichtungen alle mit einem zufällig gewählten reellen Wert (idealerweise zwischen -0.5 und +0.5) belegt wurden.

Die verborgenen Schichten und die Ausgabeschicht beinhalten zusätzlich ein Bias-Neuron. Dieses besondere Neuron besitzt keine eingehenden Verbindungen und gibt als Aktivierungswert den Wert 1.0 zurück (siehe [Lip05]). Ihre Existenz ist für den Lernprozess entscheidend, ohne sie kann ein neuronales Netz ein Eingabemuster, welches nur aus Nullwerten besteht, nicht sinnvoll verarbeiten.

[9] Feedforward Netze ist der Begriff für künstliche neuronale Netze, bei denen kein Pfad, der von einem gegebenen Neuron direkt oder über zwischengeschaltete Neuronen wieder zu diesem Neuron zurückführt, existiert. Die mathematische Topologie eines solchen Netzes ist die eines azyklischen Graphen (siehe [Lip05], S. 53).

3.3.2 Aktivierungspropagierung durch Feedforward Netze

Feedforward Netze bieten eindeutige Funktionen, die Aktivierungswerte einzelner Neuronen von der Eingabeschicht zur Ausgabeschicht weiterzuleiten; hierbei spielen die Neuronen als Hauptprozesseinheit eine wesentliche Rolle.

Ein künstliches Neuron mit dem Index *j* wird durch fünf Eigenschaften charakterisiert (vgl. [Lip05], S. 47):

- Eingabevektor $\vec{x}$
- Gewichtungsvektor $\vec{w}$
- Netzeingabefunktion net_j
- Aktivierungsfunktion φ_j
- Ausgabefunktion o_j

Jeder dieser fünf Eigenschaften ist abhängig von der Schicht, in der sich das entsprechende Neuron befindet. Abbildung 28 zeigt den Aufbau eines einzelnen Neurons.

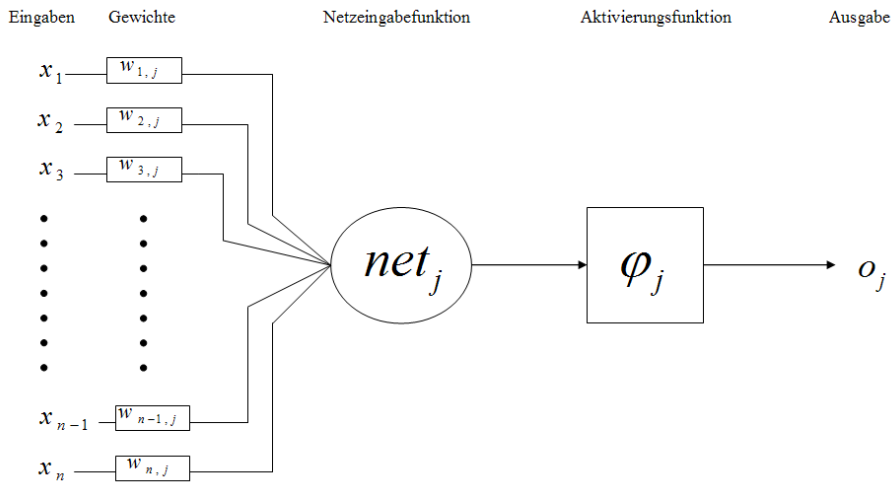


Abb. 28 Künstliches Neuron mit Index j
Quelle: Neuzeichnung in Anlehnung an [Lip05], S. 46, Abb. 2.2

Der Kern eines Neurons besteht aus der *Netzeingabefunktion*, welche die Summe der Produkte der einzelnen Eingaben und den dazugehörigen Gewichtungen zurückgibt. Sie ist für alle Neuronen mit dem Index j, außer für die der Eingabeschicht definiert als:

$$net_j = \sum_{i=1}^{n} w_{ij} x_i + \beta_j$$

wobei w_{ij} die Gewichtung der Verbindung von Neuron *i* zu Neuron *j*, β_j der Aktivierungswert des Bias-Neurons und x_i die Ausgabe des Neurons *j* ist. Die Netzeingabe verringert sich bei negativen Gewichtungen zu Vorgängerneuronen, bei positiven wird sie im Gegenzug erhöht. Die Netzeingabe für Neuronen der Eingabeschicht beschränkt sich auf die Übernahme des Eingabevektors, der in diesem Fall nur aus einem von der Umwelt vorgegebenen reellen Wert besteht.

Die *Ausgabe* eines Neurons besteht in den meisten Fällen nicht aus einer einfachen linearen Weitergabe des Netzeingabewertes an eventuell folgende Neuronen. *Aktivierungsfunktionen* schränken die Ausgabe auf einen bestimmten Wertebereich ein. Alle Neuronen der verborgenen Schichten nutzen die sigmoide Funktion:

$$\varphi_j = \frac{1}{1 + e^{\frac{-net_j}{T}}} .$$

Neuronen der Eingabe- bzw. Ausgabeschicht nutzen die Identitätsfunktion als Aktivierungsfunktion. Abbildung 29 zeigt den Verlauf der sigmoiden Funktion mit Streckfaktor $T = 1$.

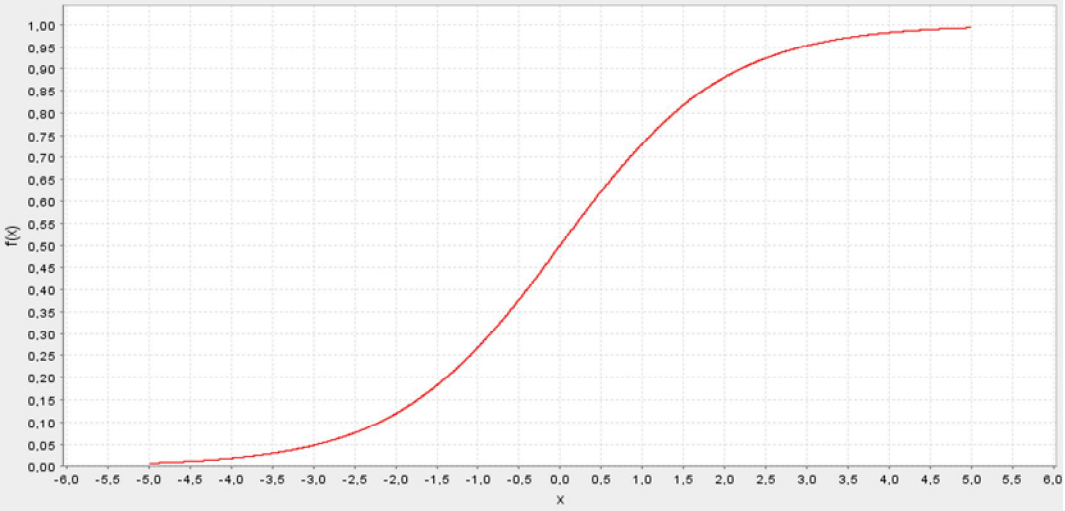


Abb. 29 Sigmoide Funktion mit T=1

Für die Ausgabefunktion aller Neuronen ergibt sich somit:

$$o_j = \varphi_j(net_j)$$

Der iterative Algorithmus zur Berechnung aller Aktivierungswerte und Ausgaben, der häufig auch unter dem Namen „*Forward Pass*“ zu finden ist (siehe [Lip05], S. 89), kann innerhalb eines neuronalen Netzes, wie es in Abbildung 28 illustriert ist, angewendet werden. Er kann halbformal wie im Folgenden zusammengefasst werden (es sei S die Anzahl der Schichten und #S die Anzahl der Neuronen innerhalb einer Schicht, H ist die aktuelle Schicht):

1. H = 1: Für alle Neuronen j = 1 bis #H

$$net_j = x_j$$

$$\varphi_j = net_j$$

$$o_j = \varphi_j(net_j)$$

2. 1 < H < S: Für alle Neuronen j = 1 bis #H

$$net_j = \sum_{i=1}^{\#(H-1)} w_{ij} o_i + \beta_j$$

$$\varphi_j = \frac{1}{1 + e^{\frac{-net_j}{T}}}$$

$$o_j = \varphi_j(net_j)$$

3. H=S: Für alle Neuronen j = 1 bis #H

$$net_j = \sum_{i=1}^{\#(H-1)} w_{ij} o_i + \beta_j$$

$$\varphi_j = net_j$$

$$o_j = \varphi_j(net_j)$$

3.3.3 Gewichtungsmodifikation durch die generalisierte Delta-Regel

Im folgenden Unterkapitel wird die *Gewichtungsmodifikation* und die damit verbundene *generalisierte Delta-Regel*[10] in Anlehnung an Rumelhart, Williams und Hinton (vgl. [Rum86], S. 322-327) erläutert.

Das Problem des *Lernens* innerhalb eines neuronalen Netzes besteht hauptsächlich aus dem Finden einer geeigneten Verteilung der Verbindungsgewichtungen, sodass die Funktion des Netzes nach Berechnung aller Aktivierungen der Ausgabeschicht erfüllt wird.

Einem untrainierten neuronalen Netz wird ein Trainingsmuster als Eingabe gegeben. Nach Berechnung aller Aktivierungen werden die Werte der Ausgabeschicht mit den gewünschten Werten des Zielmusters verglichen. Stimmen diese im Sinne der Funktion des neuronalen Netzes überein, werden keine Veränderungen vorgenommen. Falls sich die Ausgaben von dem Zielmuster unterscheiden, müssen Verbindungsgewichtungen innerhalb des Netzes geändert werden. Um zu ermitteln, welche Gewichtungen für fehlerhafte Aktivierungen der Ausgabeschicht verantwortlich sind, wird ein *Fehlermaß* E_p für die Leistungsfähigkeit eines neuronalen Netzes mit Eingabemuster *p* definiert:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 ,$$

wobei *j* die Anzahl der Ausgabeneuronen, $t_{j,p}$ der Wert des Zielmusters an der Stelle *j* und $o_{j,p}$ die Ausgabe des j-ten Neurons repräsentiert.
Für die Funktionsfähigkeit *E* des gesamten Netzes werden die Fehlermaße für jedes Trainingsmuster aufsummiert und es gilt:

$$E = \sum_p E_p .$$

Die Modifikation der Verbindungsgewichte innerhalb eines neuronalen Netzes mit beliebig vielen verborgenen Schichten erfolgt nach dem B*ackpropagation of Error*[11] Verfahren, welches auf der generalisierten Delta-Regel

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}$$

[10] Im Gegensatz zur einfachen Delta-Regel, welche auf neuronale Netze ohne verborgene Schichten anzuwenden ist, erweitert die generalisierte Delta-Regel den Anwendungsbereich auf neuronale Netze mit beliebig vielen verborgenen Schichten (vgl. [Rum86], S. 324).
[11] Dieses Verfahren wurde ab 1970 von mehreren Autoren unabhängig vorgeschlagen, u. a. in der Dissertation von Paul Werbos. Anfang der 80er Jahre rückte es durch Rumelhart, Hinton und Williams [Rum86] wieder in den Fokus (siehe [Lip05], S. 87).

basiert, die rückwärts gerichtet auf alle Verbindungsgewichtungen zwischen den einzelnen Schichten, beginnend bei der Ausgabeschicht, angewandt wird. Innerhalb der Regel gibt p an, welches Trainingsmuster gerade an die Eingabeschicht angelegt wird, w_{ji} ist die Verbindungsgewichtung zwischen dem *j*-ten Neuron der Schicht *H* und dem *i*-ten Neuron der Vorgängerschicht *H-1*.

δ_{pj} ist das *Fehlersignal* des *j*-ten Neurons der Schicht *H* und muss zuvor berechnet sein, wobei sich die Berechnung der Fehlersignale der Ausgabeneuronen von denen der anderen unterscheidet. o_{pi} ist die Ausgabe des *i*-ten Neurons der Schicht *H-1*. η ist die *Lernrate* des neuronalen Netzes.

Eine Veränderung der Verbindungsgewichtungen nach der oben genannten generalisierten Delta-Regel minimiert das Fehlermaß E_p, da die Ableitung von E_p hinsichtlich der Gewichtungen zur vorgegeben Delta-Regel mit negativem Proportionalitätsfaktor proportional ist. Diese negative Proportionalität gleicht dem größtmöglichen Gradientenabstieg[12] auf einer Oberfläche, deren Höhe dem Fehlermaß E_p entspricht.

Es muss also gezeigt werden, dass

$$\Delta_p w_{ij} \propto -\frac{\delta E_p}{\delta w_{ji}}.$$

Wie in Abschnitt 3.3.2 bereits erläutert, geben Neuronen der verborgenen Schicht nicht direkt ihre Netzeingabe

$$net_{pj} = \sum_i w_{ji} o_{pi}$$

mit Hilfe der linearen Identitätsfunktion an nachfolgende Neuronen weiter, sondern wenden die semilineare, differenzierbare, steigende sigmoide Funktion φ_j als Aktivierungsfunktion auf die Netzeingabe an. Dieses hat für die Ausgabe dieser Neuronen zur Folge:

$$o_{pj} = \varphi_j(net_{pj}).$$

Die Ableitung des Fehlermaßes E_p hinsichtlich der Verbindungsgewichtung w_{ji} wird mit Hilfe der Kettenregel in ein Produkt

[12] Das Backpropagation of Error Verfahren beruht auf dem Gradientenabstiegsverfahren. In einem Punkt $\vec{w}$ wird die Tangente der Fehleroberfläche bestimmt und um eine gewisse Länge abgestiegen, wodurch man eine neue Gewichtungsverteilung $\vec{w}$ erhält. Dieses Verfahren wird so lange wiederholt, bis ein lokales Minimum der Fehleroberfläche erreicht ist (siehe [Lip05], S. 95).

$$\frac{\delta E_p}{\delta w_{ji}} = \frac{\delta E_p}{\delta net_{pj}} \frac{\delta net_{pj}}{\delta w_{ji}}$$

umgeformt. Der erste Faktor reflektiert die Wirkung der Netzeingabe net_{pj} auf das Fehlermaß E_p, der zweite Faktor den Einfluss einer geänderten Verbindungsgewichtung auf die Netzeingabe net_{pj}.
Aus dem zweiten Faktor wird die Ableitung

$$\frac{\delta net_{pj}}{\delta w_{ji}} = o_{pi}$$

gebildet.
Als Fehlersignal δ_{pj} des *j*-ten Neurons der Schicht *H* wird nun festgelegt:

$$\delta_{pj} = -\frac{\delta E_p}{\delta net_{pj}} \ .$$

Nun ergibt sich für die Ableitung des Fehlermaßes E_p durch Substitution der beiden Faktoren die äquivalente Form

$$-\frac{\delta E_p}{\delta w_{ji}} = \delta_{pj} o_{pi},$$

welche die Proportionalität zwischen der Ableitung und der generalisierten Delta-Regel zeigt.

Bevor die generalisierte Delta-Regel zur Gewichtungsmodifikation genutzt werden kann, müssen zunächst für alle Neuronen des neuronalen Netzes, bis auf die der Eingabeschicht, Fehlersignale rekursiv rückwärts gerichtet berechnet werden. Es unterscheidet sich dabei die Berechnung der Fehlersignale für Neuronen der Ausgabeschicht von den Neuronen der verborgenen Schichten. Das Fehlersignal δ_{pj} wird durch Anwendung der Kettenregel wieder in ein Produkt zerlegt. Es gilt:

$$\delta_{pj} = -\frac{\delta E_p}{\delta net_{pj}} = -\frac{\delta E_p}{\delta o_{pj}} \frac{\delta o_{pj}}{\delta net_{pj}} .$$

Der erste Faktor gibt den Fehler als Funktion der Ausgabe eines Neurons wieder, wohingegen der zweite Faktor die Ausgabe eines Neurons als Funktion seiner Netzeingabe sieht. Der zweite Faktor kann mit Hilfe der Ableitung der Aktivierungsfunktion φ notiert werden als:

$$\frac{\delta o_{pj}}{\delta net_{pj}} = \varphi'(net_{pj}) .$$

Für die Ableitung des ersten Faktors werden zwei Fälle unterschieden. Für alle Neuronen der Ausgabeschicht gilt:

$$\frac{\delta E_p}{\delta o_{pj}} = -(t_{pj} - o_{pj}) .$$

Somit ergibt sich für das Fehlersignal aller Neuronen der Ausgabeschicht:

$$\delta_{pj} = (t_{pj} - o_{pj})\varphi'(net_{pj}) .$$

Für die Neuronen aller anderen Schichten errechnet sich ihr Fehlersignal aus den Fehlersignalen der *k* nachfolgenden Neuronen, dies bedeutet für die Herleitung des ersten Faktors:

$$\frac{\delta E_p}{\delta o_{pj}} = \sum_k \frac{\delta E_p}{\delta net_{pk}} \frac{\delta net_{pk}}{\delta o_{pj}} = \sum_k \frac{\delta E_p}{\delta net_{pk}} w_{kj} = -\sum_k \delta_{pk} w_{kj} .$$

In Folge dessen ergibt sich für diese Neuronen das Fehlersignal:

$$\delta_{pj} = \varphi'(net_{pj}) \sum_k \delta_{pk} w_{kj} .$$

3.3.4 Lernalgorithmus

Das folgende Unterkapitel orientiert sich weitgehend an den Ausführungen von Rumelhart, Williams und Hinton (vgl. [Rum86], S. 327 – 329).

Die Ausgangslage, um ein neuronales Netz im Sinne seiner Funktion zu trainieren, ist eine Menge *P* an Trainingsmustern, die dem Netz präsentiert werden. Der Backpropagation of Error *Lernalgorithmus*, auch „Backward-Pass“ genannt (siehe [Lip05], S. 91), und die damit verbundenen Gewichtungsmodifikationen, kann in drei Phasen unterteilt werden. In der ersten Phase wird ein zufällig ausgewähltes Trainingsmuster *p* aus der Menge *P* dem Netz präsentiert, d. h. die Neuronen der Eingabeschicht übernehmen die Werte des Trainingsmusters. Diese Werte werden dann im gesamten Netz, wie in Abschnitt 3.3.2 beschrieben, bis zur Ausgabeschicht propagiert. Stimmen die Aktivierungen der Ausgabeneuronen nicht mit denen des gewünschten Zielmusters überein, wird die zweite Phase begonnen, in der die Fehlersignale für jedes Neuron rekursiv und rückwärts

gerichtet berechnet werden. Im ersten Schritt dieser Phase werden zunächst die Fehlersignale δ_{pj} für jedes Neuron *j* der Ausgabeschicht nach folgender Regel berechnet:

$$\delta_{pj} = (t_{pj} - o_{pj})\varphi'(net_{pj})$$

Für jedes Ausgabeneuron wird das Produkt aus der Differenz zwischen gewünschter und tatsächlicher Ausgabe und der Ableitung seiner Aktivierungsfunktion als Fehlersignal festgelegt. Die Aktivierungsfunktion eines Ausgabeneurons kann sowohl die Identitätsfunktion, deren Ableitung 1 ist, als auch die sigmoide Funktion, deren Ableitung $\varphi' = \varphi(1-\varphi)$ ist, sein. Sind alle Fehlersignale der Neuronen der Ausgabeschicht berechnet, werden die Fehlersignale der Neuronen der verborgenen Vorgängerschichten nach folgender Regel berechnet:

$$\delta_{pj} = \varphi'(net_{pj})\sum_{k}\delta_{pk}w_{kj}$$

Um das Fehlersignal des *j*-ten Neurons einer verborgenen Schicht zu berechnen, müssen zunächst alle Fehlersignale δ_{pk} aller mit Neuron *j* verbundenen Neuronen k, multipliziert mit der entsprechenden Verbindungsgewichtung, aufsummiert werden. Neuron *j* erhält das Produkt aus dieser Summe und der Ableitung seiner Aktivierungsfunktion, die Ableitung der sigmoiden Funktion, als Fehlersignal δ_{pj}. Für die Neuronen der Eingabeschicht wird keinerlei Fehlersignal berechnet. Sind alle Fehlersignale berechnet, beginnt die dritte Phase, die Gewichtungsmodifikation nach der generalisierten Delta-Regel:

$$\Delta_p w_{ji} = \eta\delta_{pj}o_{pi}$$

Ausgehend von den Verbindungen aller Neuronen *j* der Ausgabeschicht zu allen Neuronen *i* der letzten verborgenen Schicht werden alle Gewichtungen um $\Delta_p w_{ji}$ geändert. Die Lernrate η sollte dabei als eine möglichst große reelle Zahl zwischen 0.0 und 1.0 gewählt werden, um ein schnelles Lernen des Netzes zu gewährleisten. Es folgt somit für die neuen Verbindungsgewichtungen:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta_p w_{ji}(t)$$

Wenn diese Gewichtungsmodifikation rückwärts gerichtet auf alle Verbindungen einschließlich der Verbindungen zu den Neuronen der Eingabeschicht durchgeführt wurde, gelten die dritte Phase und ein Durchlauf des Algorithmus als beendet. Dieser Lernalgorithmus wird nun für alle Trainingsmuster einmal durchgeführt.

3.3.5 Momentum-Version

Das Ziel der Momentum-Version[13], einer Modifikation des Backpropagation of Error Verfahrens, ist es, das Gradientenabstiegsverfahren so zu beeinflussen, dass sich auf flachen Ebenen der Fehleroberfläche die *Schrittweite* η des Abstiegs erhöht bzw. sie sich in den Tälern der Oberfläche reduziert. Dieses Verhalten wird dadurch erreicht, dass die in der Vergangenheit erfolgten Veränderungen der Verbindungsgewichtungen einen Einfluss auf die aktuelle Gewichtungsveränderung haben (vgl. [Lip05], S. 109f). Demnach wird im Simulationsschritt t jede Gewichtung $w_{ji}(t)$ des Netzes folgendermaßen modifiziert:

$$w_{ji}(t+1) = w_{ji}(t) + \Delta_p w_{ji}(t)$$
$$\Delta_p w_{ji}(t) = (1-\alpha)\,\eta\delta_{pj}o_{pi} + \alpha\,\Delta_p w_{ji}(t-1)$$

Dabei ist η die Lernrate und $\alpha \in [0,1[$ das *Momentum*. Der Term $\Delta_p w_{ji}(t-1)$ beinhaltet die zuletzt durchgeführte Gewichtungsmodifikation und wird mit dem Faktor α in die aktuell durchgeführte Modifikation eingerechnet. Setzt man das *Trägheitsmoment* $\alpha = 0$, so erfolgt die Gewichtungsmodifikation wieder gemäß der in Abschnitt 3.3.4 erläuterten generalisierten Delta-Regel. Idealerweise wird für das Momentum ein Wert von 0,9 gewählt, um den Gradientenabstieg auf den meist flachen Fehleroberflächen zu beschleunigen. Auf sehr stark gekrümmten Oberflächen ist ein zu groß gewähltes Momentum für das Backpropagation of Error Verfahren eher hinderlich.

3.4 Vorhandene Modelle

3.4.1 Das Modell von Hutchins und Hazlehurst

In diesem Abschnitt wird das von Edwin Hutchins und Brian Hazlehurst erstellte Modell zur Emergenz eines Lexikons innerhalb einer Gemeinschaft von Agenten vorgestellt, das als Basis für das in dieser Arbeit erstellte Modell angesehen werden kann und welches Bestandteil der vorangegangenen Diplomarbeit mit dem Titel „*LexLearn – Emergenz eines gemeinsam genutzten Lexikons*“ ([FK07]) ist. Es wird hierbei sowohl genauer auf die Struktur eines Lexikons als auch auf bestimmte Eigenschaften der mit neuronalen Netzen ausgestatteten Agenten und deren Gemeinschaft eingegangen.

[13] Diese Version des Backpropagation of Error Verfahrens geht auf Hinton und Williams zurück und wurde erstmals in [Rum86], S. 327 erwähnt. „Konjugierter Gradientenabstieg“ wird als synonyme Bezeichnung für dieses Verfahren verwendet (vgl. [Lip05], S. 109).

Bedingungen an ein gemeinsam genutztes Lexikon

Die Erläuterungen im folgenden Unterkapitel orientieren sich an den Ausführungen von Hutchins und Hazlehurst in [HH95], S. 161–165.

Das Erfinden eines gemeinsam genutzten Lexikons beinhaltet das zentrale Problem einer exakten Darstellung der Zuordnung einer visuellen Szene der Umwelt zu einem Symbol, welches jeder Agent für sich erstellt.

Ausgehend von zwei Agenten einer Gemeinschaft, A und B, und einer Menge von visuellen Szenen einer Welt, nummeriert mit 1,2,3,4,….*m*, kann die Konkatenation des Buchstabens des Agenten mit der Ziffer einer Szene als mögliche Darstellung der oben genannten Zuordnung gesehen werden. So beinhaltet „A*3*“ zum Beispiel das Symbol, welches Agent A zur Darstellung der dritten Szene einer Umwelt nutzt.

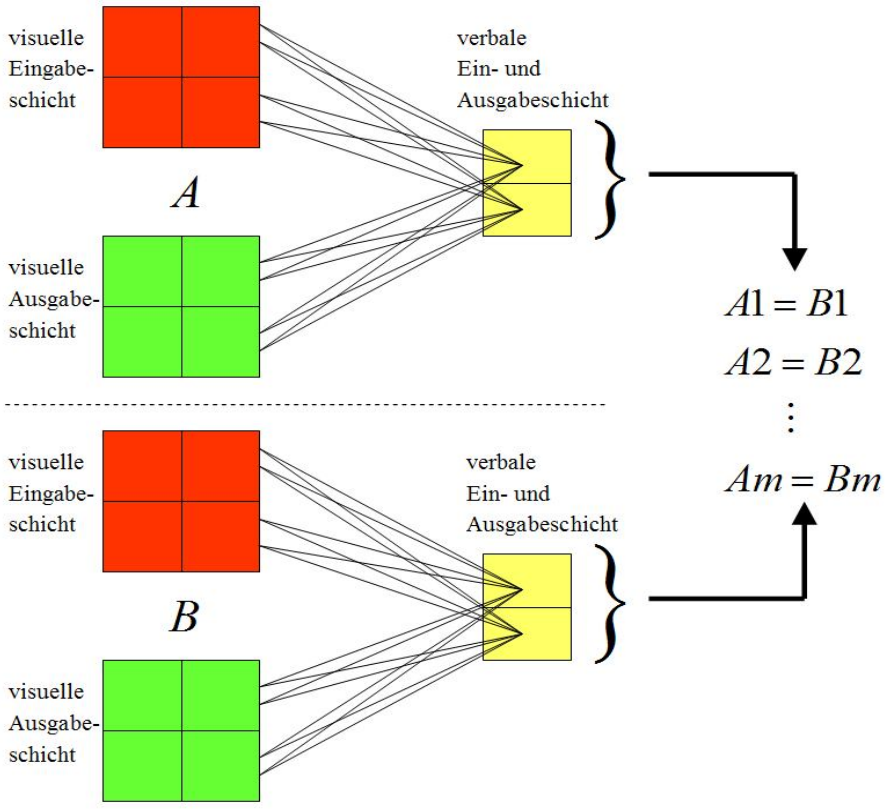


Abb. 30 Schema der Konsensbildung
Quelle: Neuzeichnung [HH95], S. 166, Figure 9.4

Soll ein neu entstandenes Lexikon von beiden Agenten A und B genutzt werden, muss es zwei grundlegende Bedingungen erfüllen:

1. Das Wort, welches A für eine spezielle Szene nutzt, muss mit dem von B benutzten Wort übereinstimmen. Allgemein gesehen bedeutet dies: A1 = B1, A2 = B2, A3 = B3, ….., A*m* = B*m* .
2. Die Wörter, die ein Agent A und B für eine bestimmte Menge Szenen 1,2,3….*m* erstellt, müssen sich voneinander unterscheiden. Es muss somit gelten: A1 ≠ A2 ≠ A3 ≠ … A*m* und B1 ≠ B2 ≠ B3 ≠ … B*m* .

Ein neu entwickeltes Lexikon kann dementsprechend als Konsens der Einhaltung dieser beiden Bedingungen gesehen werden.

Jedes neuronale Netz aller Agenten ist darauf ausgelegt, diese Bedingungen einzuhalten. Die Netze der Agenten des Modells von Hutchins und Hazlehurst besitzen eine *visuelle* Eingabe- und Ausgabeschicht, d. h. ihre Fähigkeiten beschränken sich auf das Sehen und Abzeichnen eines von der Umwelt vorgegebenen Musters. Die letzte verborgene Schicht repräsentiert das *Wort* oder *Symbol* eines Agenten für die aktuell angelegte visuelle Szene der Umwelt und wird als *verbale* Ein- und Ausgabeschicht definiert. Da der Inhalt dieser Schicht für andere Agenten sichtbar ist, handelt es sich nicht um eine verborgene Schicht.

Das Erfüllen der *ersten Bedingung* besteht darin, dass zwei oder mehrere Agenten das gleiche Wort für eine bestimmte Szene der Umwelt benutzen. Abbildung 30 skizziert dabei die Konsensbildung zweier Agenten *A* und *B*.

Neuronale Netze erfüllen diese Bedingung, indem sie als Gemeinschaft auftreten und nicht nur individuell betrachtet werden. Im Falle der oben genannten zwei Agenten A und B und deren neuronalen Netzen A und B sehen beide Netze das jeweils andere als Lehrer an und versuchen, dessen verbale Ausgabe als verbales Zielmuster zu übernehmen. Für diesen Fall wird es nach mehrfachen Durchläufen des Lernalgorithmus zu einem Konsens auf der Wortebene zu einer bestimmten Szene der Umwelt kommen.

Das Erfüllen der *zweiten Bedingung*, kein gleiches Wort für zwei oder mehrere Szenen der Umwelt zu benutzen, wird durch die neuronalen Autoassociator Netze automatisch erfüllt. Rumelhart, Hinton and Williams (vgl. [HH95], S. 165) haben gezeigt, dass unter bestimmten Umständen vollkommen trainierte Netze die Aktivierungen der Neuronen der verborgenen Schichten die Eingabemuster der Umwelt effizient und überschneidungsfrei verschlüsseln. Werden zum Beispiel an ein Autoassociator Netz mit vier Neuronen in der Eingabeschicht, zwei Neuronen in der verborgenen Schicht und vier Neuronen in der Ausgabeschicht vier orthogonale Eingabemuster angelegt, so sollten die Aktivierungen der Neuronen der verborgenen Schicht zu {(0,0),(0,1),(1,0),(1,1)} konvergieren.

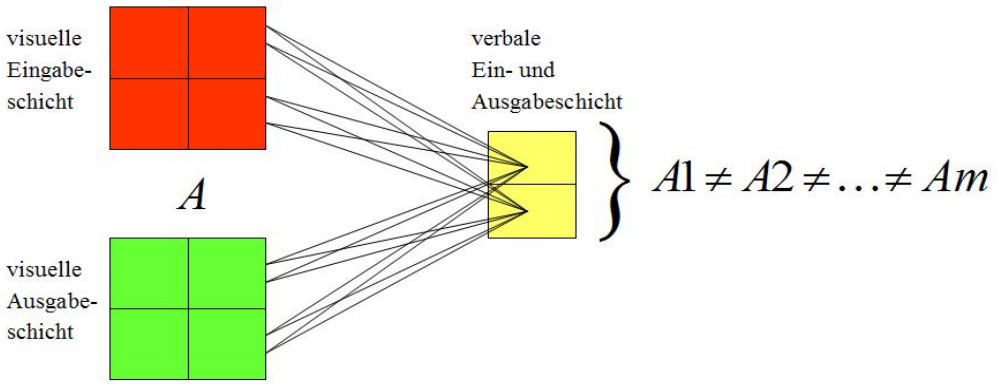


Abb. 31 Schema der Wortdifferenz
Quelle: Neuzeichnung [HH95], S. 165, Figure 9.3

Die Eigenschaften Avg1 und Avg2

Die Ausführungen in diesem Abschnitt orientieren sich an den Erläuterungen von Hutchins und Hazlehurst [HH95], S. 185f.

Innerhalb einer Gemeinschaft von *n* Agenten, die in einer Welt mit *m* visuellen Szenen leben, kann die Gemeinschaftssprache L_t, welche die Wörter aller Agenten *n* für alle Szenen *m* beinhaltet, in Form der Matrix

$$L_t = \begin{matrix} s_{1,1}(t) & s_{1,2}(t) & \dots & \dots & s_{1,j}(t) \\ s_{2,1}(t) & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ s_{i,1}(t) & s_{i,2}(t) & \dots & \dots & s_{i,j}(t) \end{matrix}$$

notiert werden. Die Sequenz der Matrizen $\{L_0, L_1, ..., L_\varphi\}$ stellt die Evolution der Sprache über den Zeitraum t = 0 bis t = φ dar. $s_{i,j}(t)$ steht dabei für das Wort des Agenten *j* für die visuelle Szene *i* zum Zeitpunkt *t*.

Zu jedem Zeitpunkt *t* kann der Grad der Entwicklung der Sprache L_t durch zwei Eigenschaften ausgedrückt werden:

- Avg1, die durchschnittliche Differenz zwischen allen Wortpaaren eines Agenten für alle Agenten und Szenen;
- Avg2, die durchschnittliche Differenz zwischen allen Wortpaaren der gleichen Szene für alle Agenten und Szenen.

Avg1 bietet demnach ein Maß der *Fähigkeit eines Agenten*, Wörter zu kreieren, die zwischen den einzelnen Szenen unterscheiden, während Avg2 ein Maß der *Fähigkeit einer Gemeinschaft* ist, sich auf gleiche Wörter für eine bestimmte Szene zu einigen.

Formal gesehen sind $s_{i,j}(t)$ und $s_{q,p}(t)$ Vektoren mit reellen Werten der Länge γ, wobei *i* und *q* zwei einander verschiedene Elemente der Menge der visuellen Szenen bzw. *j* und *p* zwei unterschiedliche Agenten einer Gemeinschaft sind.

Als Distanzmetrik *d* wird definiert:

$$d(s_{i,j}(t), s_{q,p}(t)) = \sqrt{\frac{\sum_{k=1}^{\gamma} (r_{i,j}^{k} - r_{q,p}^{k})^2}{\gamma}}$$

Daraus folgt für die Eigenschaften Avg1 und Avg2:

$$Avg\ 1(t) = \frac{\sum_{j=1}^{n} \left(\frac{\sum_{(i_1, i_2) \in P_2(m)}^{\frac{m^2 - m}{2}} d\left(s_{i_1, j}(t), s_{i_2 j}(t)\right)}{\frac{m^2 - m}{2}} \right)}{n}$$

$P_2(m)$ ist die Menge aller Paare ganzzahliger, positiver Zahlen von 1 bis *m*, $\frac{m^2 - m}{2}$ gibt die Länge dieser Menge $P_2(m)$ an.

Auf die gleiche Weise gilt:

$$Avg\ 2(t) = \frac{\sum_{i=1}^{m} \left(\frac{\sum_{(j_1, j_2) \in P_2(n)}^{\frac{n^2 - n}{2}} d\left(s_{i, j_1}(t), s_{i, j_2}(t)\right)}{\frac{n^2 - n}{2}} \right)}{m}$$

Implementierung des Modells

Die folgende Erläuterung einer Implementierung des in den vorherigen Abschnitten beschrieben Modells gibt die Vorstellungen von Hutchins und Hazlehurst wieder [HH95], S. 166f.

Die Simulation erfolgt über *Interaktionen* innerhalb einer *Simulationsumgebung*, welche aus einer *Umwelt* mit *m* visuellen Szenen und einer *Gemeinschaft* aus n Agenten besteht. Genau eine Interaktion ist mit einem Zeitschritt innerhalb der Simulation gleichzusetzen und beinhaltet, dass zwei ausgewählten Agenten der Gemeinschaft, einem *Sprecher A* und einem *Zuhörer B*, eine der *m* visuellen Szenen präsentiert wird. Die Auswahl der Szene und der beiden Individuen erfolgt über das *Interaktionsprotokoll* und wird typischerweise durch eine einfache, *zufällige* Auswahl implementiert. Sprecher *A* reagiert auf die Präsentation einer Szene *m*, indem er über sein Feedforward Netz die Aktivierungen innerhalb seiner verbalen Ausgabeschicht berechnet. Hörer *B* reagiert in gleicher Weise, nutzt aber das Wort von *A* als Ziel, um seine eigenen Aktivierungen innerhalb der verbalen Ausgabeschicht zu korrigieren. Des Weiteren möchte *B* die aktuelle Szene erlernen und nutzt sein erzeugtes Wort, neben dem Abgleich zum Wort von A, dazu, durch Propagierung der Aktivierungen auf der visuellen Schicht eine Ausgabe zu erzeugen. Dieses Verhalten von B hat zwei wesentliche Effekte zur Folge; zum einen gleicht er sein Wort für die aktuell präsentierte Szene an das Wort von A an, zum anderen versucht er mit seinem Wort die aktuelle Szene in seiner Ausgabeschicht korrekt wiederzugeben.

Über die Zeit wird durch die zufällige Auswahl der interagierenden Agenten und der präsentierten Szenen garantiert, dass jedes Individuum einer Gemeinschaft die Chance bekommt, sowohl in sprechender als auch in hörender Rolle mit allen anderen Individuen der Gemeinschaft über alle visuellen Szenen der Umwelt zu kommunizieren.

Hutchins und Hazlehurst untersuchen den Ablauf der Simulation anhand zweier verschiedener Simulationen, einer ersten Simulation mit einer komplexen Netzstruktur innerhalb der Agenten und einer großen Menge visueller Szenen, die das Verhalten innerhalb der Simulation eher qualitativ veranschaulichen soll und einer zweiten Simulation mit einer einfachen Menge visueller Szenen (vier orthogonale Vektoren {(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)}) und einer einfacheren Netzstruktur innerhalb der Agenten, welche eher einer analytischen Untersuchung der Simulation dient.

Beobachtungen

Bei beiden Simulationen ist eine klare Differenz im Verhalten der Fehlerindikatoren Avg1 und Avg2 zu beobachten. Aufgrund der zufällig

gewählten Verbindungsgewichtungen besteht zu Beginn der Simulationen keine große Differenz zwischen den einzelnen Wörtern der Agenten, was einem niedrigen Avg1-Wert entspricht. Im Gegensatz dazu stimmen die Wörter der Agenten für eine einzelne visuelle Szene nahezu überein, welches den geringen Avg2-Wert zur Folge hat. Beginnen die Agenten ihre Wörter zu differenzieren, so steigen Avg1 und Avg2 an. Die Wörter der Agenten für eine visuelle Szene differieren nun stärker.

Ist die größtmögliche Differenz zwischen den Wörtern eines Agenten erreicht, beginnen die Agenten sich auf ein gemeinsames Lexikon zu einigen, welches ein Absinken des Avg2-Wertes zur Folge hat. Dies ist sehr deutlich in Abbildung 32 zu erkennen, die den Verlauf von Avg1 und Avg2 in der zweiten genannten Simulation unter Verwendung des Werkzeugs LexLearn aus [FK07] zeigt. Der Wert für den Avg1 steigt hier auf einen Wert von ca. 0.75 an, während der Avg2 gegen 0.0 konvergiert. Die Population besteht in diesem Beispiel aus fünf Agenten, die mit einem simplen neuronalen Netz mit einer Ein- und Ausgabeschicht mit vier Neuronen und einer verborgenen Schicht bestehend aus zwei Neuronen ausgestattet sind.

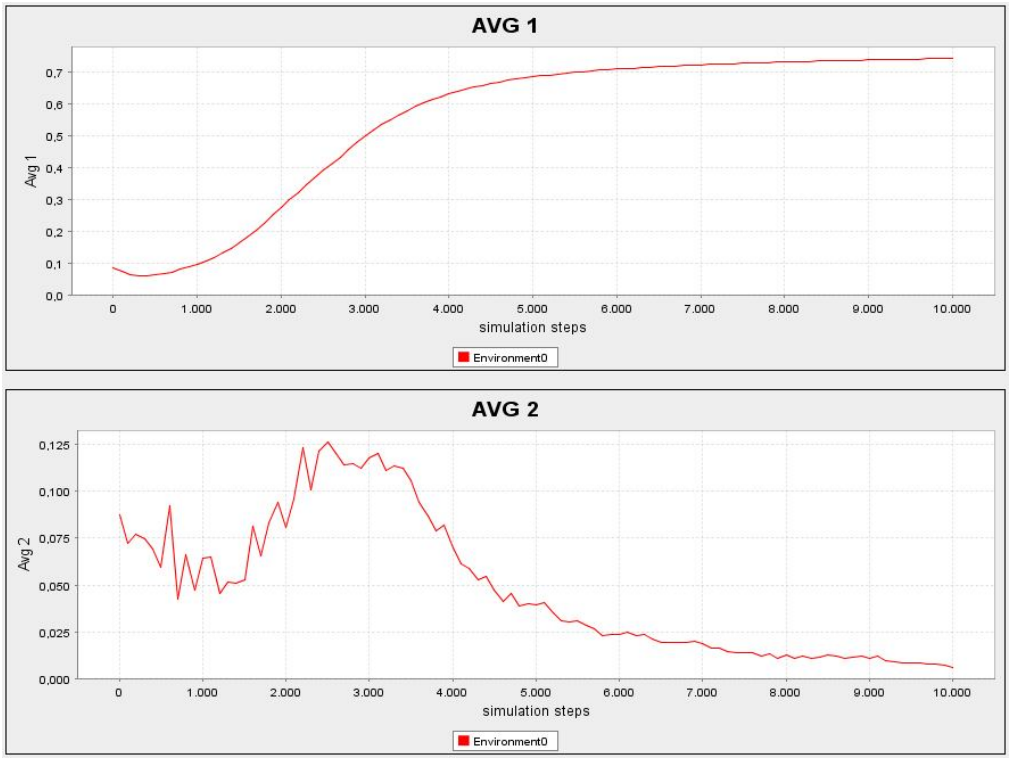


Abb. 32 Verlauf von Avg1 und Avg2 unter der Verwendung von LexLearn

Die Ergebnisse der komplexeren Simulation sind ähnlich, jedoch sind die für das Fehlermaß Avg1 erreichten Werte etwas schwächer, was sich auf eine größere Anzahl visueller Szenen zurückführen lässt. Fuchs und Klein zeigen in ihrer

Reimplementierung des Modells (siehe [FK07]), dass sich die Größe der verborgenen Schichten ebenfalls auf beide Fehlerindikatoren auswirkt. Bei zunehmender Größe der verborgenen Schichten verschlechtern sich die Eigenschaften hinsichtlich des Avg1, während die Werte bei Avg2 stets besser werden.

Die Entstehung von Dialekten

Gelegentlich gelingt es einer Gemeinschaft von Agenten nicht, ein gemeinsam genutztes Lexikon auszubilden. Dieser Effekt, der sowohl in der Implementierung von Hutchins und Hazlehurst als auch in LexLearn zu beobachten ist, tritt auf, weil es von Zeit zu Zeit vorkommt, dass zufällige Initialgewichtungen mehrerer Agenten, gepaart mit einer „unglücklichen“ Auswahl derer innerhalb des Interaktionsprotokolls zu einer Divergenz der verbalen Repräsentationen der Agenten einer Gemeinschaft führt, die nicht wieder überwunden werden kann. In diesem Fall ist die Sprache, die eine Teilmenge der Agenten erlernen wird, inkompatibel zu der Sprache anderer Agenten. Beide Untergruppen werden sich folglich niemals auf ein gemeinsames Lexikon einigen können.

Der Tatsache entsprechend, dass einige Initialgewichtungen zu einigen kompatibler sind als zu anderen, ermöglichten es Hutchins und Hazlehurst in einer Erweiterung ihres Modells den Agenten, sich ihren Konversationspartner selbst auszuwählen. Die Agenten wurden derart erweitert, dass sie sich die vergangenen Äußerungen anderer Agenten merken konnten und bei der Auswahl ihrer Gesprächspartner Agenten bevorzugten, deren vergangene Äußerungen den eigenen Wörtern ähnlich waren. Das Ergebnis dieser Veränderung des Interaktionsprotokolls war die Entstehung von Dialekten bzw. eine Partitionierung der Agentengemeinschaft in Gruppen von Agenten, die jeweils für sich ein kohärentes Lexikon ausbildeten.

3.4.2 Das Modell von Livingstone

Daniel Livingstone entwickelte im Rahmen seiner Arbeit ebenfalls ein Modell für Multi-Agenten-Systeme, in dem Agenten eine Sprache erlernen. Dieses Modell, das er unter anderem in [Liv02] und [LF98] vorstellt, ist die Basis für die Ausführungen dieses Abschnitts.

Der Aufbau der Agenten

Obwohl die Agenten ebenfalls durch neuronale Netze interpretiert werden, erscheinen sie simpler, da sie lediglich ein zweischichtiges Feedforward Netz besitzen. Jede Schicht besteht dabei aus drei Neuronen, wobei die äußere Schicht,

die die Signalschicht darstellt, im Gegensatz zur inneren Schicht, die den internen Zustand der Agenten speichert, zusätzlich mit einem Bias-Neuron ausgestattet ist.

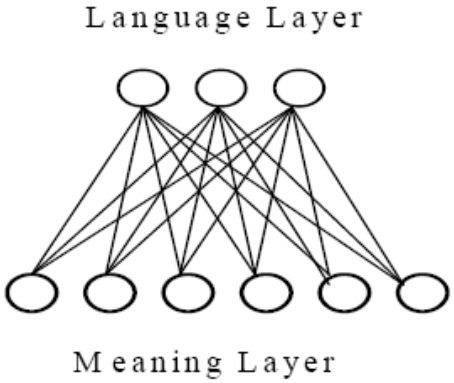


Abb. 33 Das neuronale Netz der Agenten. Quelle: [LF98]

Die internen Signale werden dabei derart kodiert, dass ein internes Neuron mit dem Wert +1 belegt wird, während die anderen beiden auf -1 gesetzt werden. Die Neuronen der Signalschicht können unabhängig voneinander die Werte +1 oder -1 annehmen, weitere Einschränkungen sind nicht vorgesehen. Dadurch ergeben sich drei mögliche interne Zustände und acht verschiedene Signale. Die dadurch erzeugte Redundanz ist für die in dem Modell gewünschte Entstehung von Dialekten unabdingbar.

Bei der Signalerzeugung wird die interne Schicht mit den Werten der Bedeutung belegt und die Werte der Signalschicht durch Aufsummieren der eingehenden Kantenbelegungen errechnet. Positive Werte ergeben dabei den Aktivierungswert +1, während negative Werte zu einem Wert von -1 führen. Die Interpretation der Signale anderer Agenten erfolgt in entgegengesetzter Richtung. Um jedoch interne Zustände mit genau einer Wertbelegung von +1 zu erhalten, greift hier eine zweite Bedingung. Das Neuron mit den höchsten aufsummierten Kantenwerten erhält die Belegung +1, alle anderen werden mit -1 belegt.

Die Kommunikation

Der Ablauf einer Kommunikation vollzieht sich dabei ähnlich wie im Modell von Hutchins und Hazlehurst. (siehe Abschnitt 3.4.1) Ein Agent wird in einem Simulationsschritt als Lehrer, ein anderer als Schüler ausgewählt. Der Lehrer produziert ein Signal anhand einer ihm gezeigten Bedeutung. Der Schüler vernimmt das erzeugte Signal und generiert seine interne Bedeutung. Die Abweichung zwischen der ursprünglichen Bedeutung *x* und der vom Schüler berechneten Bedeutung *x'* wird wie in der nachfolgenden Gleichung dazu benutzt, um die Gewichte des Schülers neu zu berechnen.

$$\Delta w_{ij} = \eta\left(x_i - x_j^{'}\right)y_j$$

Wie sich aus der Gleichung ergibt, erfolgt ein Lernen nur dann, wenn der Schüler das Signal falsch versteht.

Die Simulationsumgebung

In Analogie zum simplen Aufbau der Agenten ist die Simulationsumgebung ebenfalls sehr einfach gestaltet. Die Agenten werden stets in einer einzigen Reihe angeordnet, deren Enden nicht verbunden sind. Die Agenten können in dieser Reihe nur mit Agenten kommunizieren, die sich in einem bestimmten Abstand, der Nachbarschaftsgröße, voneinander befinden. Durch diese Anordnung entsteht eine Bevölkerung, in der keine expliziten Gruppengrenzen existieren und dennoch eine lokalisierte Kommunikation vorherrscht.

Die Agenten durchlaufen in ihrem Lebenszyklus zwei Lebensphasen, Kindheit und Erwachsenenstadium. Beide Bevölkerungsgruppen bevölkern jeweils eine Reihe mit identischer Größe. Kinder lernen nur von Erwachsenen, die Positionen in jeder Reihe bestimmen die Nachbarschaftsverhältnisse, so befindet sich beispielsweise der Agent in der Reihe der Kinder an der Position *x* in der gleichen Nachbarschaft wie der Agent an Position *x* der erwachsenen Agenten. Nach einem Training werden die bestehenden Erwachsenen entfernt, die Agenten der Reihe der Kinder werden zu Erwachsenen und neue Kinder werden erzeugt. Die Auswirkungen der Änderung folgender Faktoren werden im Rahmen verschiedener Simulationsdurchläufe untersucht:

Nachbarschaftsgröße
Die Nachbarschaftsgröße kann beliebig zwischen den beiden Extremen der ausschließlichen Kommunikationsfähigkeit eines Agenten mit seinen direkten Nachbarn und der Möglichkeit der Kommunikation mit der kompletten Bevölkerung variiert werden.

Initialzustand
Da die erste Generation eines Simulationsdurchlaufs keine Lehrer haben kann, müssen die initialen Gewichtungen der neuronalen Netze bestimmt werden. Hierbei können entweder zufällig gewählte Werte benutzt werden oder aber die Agenten werden derart initialisiert, dass sie alle das gleiche Signalisierungsschema besitzen.

Rauschen
Die Kommunikation kann entweder rauschfrei ablaufen oder aber es wird ein Mechanismus gewählt, der mit einer kleinen Wahrscheinlichkeit ein individuelles Bit eines Worts invertieren kann.

Zusätzlich können ebenfalls die Bevölkerungsgröße, die Anzahl der Bedeutungen und Signale und die Lernrate verändert werden.

Beobachtungen

Im Gegensatz zum Modell von Hutchins und Hazlehurst ist bei nahezu allen Simulationsdurchläufen eine Entstehung von Dialekten zu beobachten. Hierbei sind jedoch keine starren Dialektgrenzen zu beobachten, sondern eine Verschiebung der Dialektgrenzen über die Generationen hinweg. In einigen wenigen Fällen ist auch eine Konvergenz hin zu einem gemeinsam genutzten Lexikon erkennbar, wie es im Modell von Hutchins und Hazlehurst in den meisten Fällen auftritt. Dieser Effekt, der stets nur nach einer großen Anzahl von Simulationsschritten auftritt, ist endgültig, die Entstehung „neuer" Dialekte danach nicht mehr möglich. Gleichfalls ist eine Entstehung von Dialekten nur dann zu beobachten, wenn die Agenten nicht mit gleichen Initialzuständen einen Simulationsdurchlauf beginnen.

Durch die Aktivierung von Rauschen ist eine Veränderung zu beobachten. Gruppen von Agenten mit einem gleichen Dialekt spalten sich dadurch wieder in verschiedene neue Gruppen auf. Bei zunehmender Stärke des Rauschens wird eine Konvergenz zunehmend schwieriger zu erreichen. Die Nachbarschaftsgröße steht ebenfalls im Zusammenhang mit der Anzahl der entstehenden Dialekte. Mit zunehmender Größe der Nachbarschaft reduziert sich die Anzahl verschiedener Dialekte bis hin zu einem globalen Dialekt bei einer globalen Nachbarschaft. Die Anzahl der Bedeutungen und Signale wurde in den Simulationsdurchläufen nicht verändert, da dadurch eine Visualisierung erschwert wird. (vgl. [Liv02], S.109)

3.4.3 Das Modell von de Boer

Bart de Boer entwickelte ebenfalls ein Modell, das mit Hilfe einer Multi-Agenten-Simulation einen Aspekt der Sprachentwicklung simulieren kann. In seinem Modell, das in [deB97], [deB99] und [deB02] detailliert erläutert wird, besteht die Aufgabe der Agenten darin, eine Menge von gemeinsam genutzten Vokalen in einer Gemeinschaft auszubilden.

Der Aufbau der Agenten

Jeder Agent besitzt eine individuelle Vokalliste, die zu Beginn einer Simulation leer ist. Ein Vokal wird durch drei Parameter definiert, die Zungenposition, Zungenhöhe und Lippenstellung, die jeweils durch Werte zwischen 0 und 1 kodiert werden.

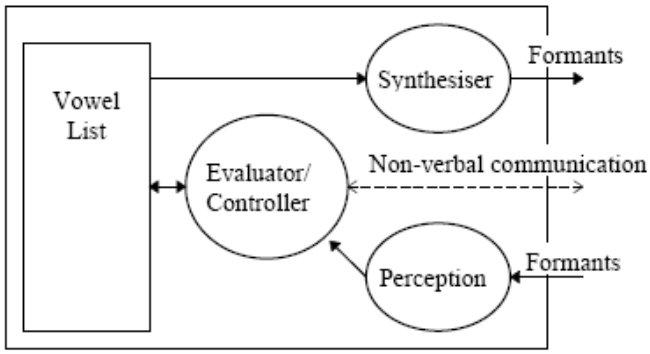


Abb. 34 Der Aufbau der Agenten. Quelle: [deB97]

Die so gespeicherten Vokale werden mit Hilfe eines Synthesizers in die Frequenzen der ersten vier Formanten des zugehörigen Vokals umgewandelt. In einigen Simulationsdurchläufen wird erzeugten Frequenzen ein Rauschen hinzugefügt. Dies geschieht durch eine Multiplikation der Erzeugungsfrequenzen mit dem Faktor $1 \pm U(a)$, wobei *U(a)* als ein zufällig gleichverteilter Wert aus dem Intervall [*-a, a*] gewählt wird. *a* variiert bei verschiedenen Experimenten. Die Verwendung von Rauschen passt die Simulation an natürliche Gegebenheiten an. Bei einer menschlichen Kommunikation kann ebenfalls nicht angenommen werden, dass alle Lautäußerungen absolut exakt verstanden werden können und ein Zuhörer somit ein Phonem akkurat kopieren kann.

Für jedes Phonem, das ein Agent kreiert, generiert er eine ideale, rauschfreie entsprechende Artikulationsform, die Prototyp-Vektor genannt wird und mit der zugehörigen Kodierung in Parameterform gespeichert wird. Jedes Mal, wenn ein Agent einen Lautvektor vernimmt, ist er in der Lage, den Abstand zwischen dem gehörten Ton und den ihm bekannten Prototyp-Vektoren zu berechnen. Der Prototyp-Vektor mit der geringsten Differenz zu dem vernommenen Phonem wird dabei als das gehörte Phonem angesehen. Dieser Prozess ist komplett mit Hilfe von neuronalen Netzen implementiert, um dem Prozess eine biologische Plausibilität zuzufügen. Der genaue Aufbau der Netze wird jedoch in keiner der angegebenen Quellen erläutert.

Die Kommunikation

Der Simulationsumgebung ist simpel gehalten. Eine Positionierung der Agenten findet nicht statt, die Wahrscheinlichkeit für eine Kommunikation zwischen zwei Agenten ist somit stets gleich.

Zur Kommunikation werden zwei Agenten zufällig ausgewählt, ein Sprecher und ein Zuhörer, oder wie de Boer sie nennt, ein Initiator und ein Imitator. Der Sprecher wählt eines seiner Phoneme zufällig aus und erzeugt eine verbale

Repräsentation, die der Zuhörer hört. Der Zuhörer interpretiert dieses, möglicherweise verrauschte Soundpattern, kalkuliert die Distanz zu den ihm bekannten Phonemen und produziert das dem gehörten ähnlichste Phonem. Der Sprecher verarbeitet den soeben gehörten Laut auf die gleiche Weise und berechnet ebenfalls das ähnlichste ihm bekannte Phonem. Ist das vom Sprecher anfangs ausgewählte Phonem identisch mit dem zuletzt vernommenen, so ist die Kommunikation erfolgreich verlaufen. Dies wird über eine nonverbale Kommunikationsform Initiator und Imitator verständlich gemacht.

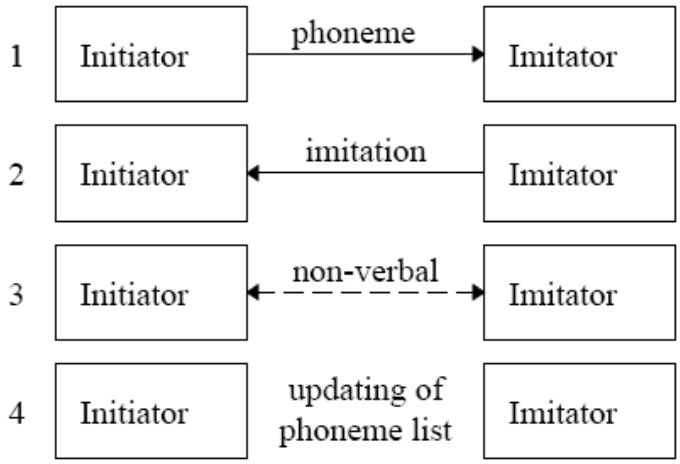


Abb. 35 Die Kommunikation im Modell von deBoer. Quelle: [deB97]

In den Listen beider Kommunikationsteilnehmer wird für jedes Phonem vermerkt, wie oft es benutzt und wie oft es erfolgreich benutzt wurde. So können Agenten zwischen Phonemen mit hoher und niedriger Qualität unterscheiden.

Da die Phonemlisten zu Beginn eines Simulationsdurchlaufs leer sind und ohne eine zufällige Kreierung eines Phonems nie Phonemmengen ausgebildet würden, sind die Agenten ebenfalls mit einem Mechanismus ausgestattet, der zufällig Phoneme erzeugen kann. Ein Zuhörer, der einen ihm unbekannten Laut vernimmt, kann diesen Laut sich auch selbst antrainieren, in dem er ihn für sich selbst zu reproduzieren versucht. Ist die Imitation gut genug, so fügt er dieses neue Phonem seiner Liste hinzu. Besitzt ein Sprecher bei einer Kommunikation bereits eine ausgeprägte Lautliste, so wählt er zur Kommunikation einen der bekannten Laute, oder aber, mit einer kleinen Wahrscheinlichkeit, kreiert er ein neues zufälliges Phonem. Der Zuhörer versucht nach einer erfolgreichen Imitation seinen imitierten Laut dem des Sprechers weiter anzupassen.

Bei einer fehlgeschlagenen Kommunikation und wenn die Qualität des Phonems schlecht war, so wird das Phonem ebenfalls an das gehörte angepasst, damit der Agent sein eigenes Repertoire verbessert. Bei einer hohen Qualität eines Phonems wird es nicht verändert, auch nicht, wenn es das Ähnlichste zu einem neu

vernommenen Laut ist, da es offensichtlich bei der Nachahmung eines anderen Phonems sehr erfolgreich ist. In diesem Fall erzeugt der Agent ein neues Phonem und fügt es, nachdem er es sich selbst antrainiert hat, seiner Lautliste hinzu.

Zwei weitere Prozesse laufen während der Simulation ab. Phoneme, die über eine lange Zeit eine schlechte Qualität aufweisen, werden aus der Lautliste gelöscht, dazu werden Laute, deren Kommunikationserfolgsquotient unter einem Grenzwert liegt, mit einer bestimmten Wahrscheinlichkeit gelöscht. Weiterhin werden Phoneme, die zu dicht beieinander liegen, zu einem Phonem zusammengefügt, da sie aufgrund des Rauschens sehr leicht verwechselt werden können.
Zu diesem bisher beschriebenen Grundmodell existieren noch drei kleinere Abänderungen, die besonders in [deB99] erläutert werden:

1. Die Möglichkeit der Simulation einer offenen Population wurde später hinzugefügt. Die Agenten sterben in einem Simulationsschritt mit einer bestimmten Wahrscheinlichkeit, werden damit aus der Simulation gelöscht und neue werden mit der gleichen Wahrscheinlichkeit geboren und somit der Gemeinschaft hinzugefügt.

2. Des Weiteren führte de Boer ein Agentenalter ein, das es jüngeren Agenten ermöglicht, schneller zu lernen, während der Mechanismus bei betagten Agenten das Lernen abbremst.

3. Im Grundmodell konnte ein Agent alleine unbegrenzt lange versuchen ein bestimmtes Phonem zu imitieren. Diese Methode wird später begrenzt, sodass er nun maximal zehn Versuche besitzt um ein Phonem abzubilden.

Beobachtungen

In mehreren Versuchsanordnungen zeigt de Boer, dass die Agenten in diesem Modell ein Vokalrepertoire ausbilden können. Nach 1000 Simulationsschritten tritt im Standardmodell (Rauschen: 0.1 bei 5 Agenten) bereits der Fall ein, dass die korrespondierenden Phoneme der verschiedenen Agenten sich einander sehr ähnlich sind, während sie innerhalb eines Agenten weit auseinander liegen. (vgl. Abb. 36)

Dies entspricht in etwa den Beobachtungen aus den Simulationen im Modell von Hutchins und Hazlehurst, wo ein möglichst großer Wert für das Ähnlichkeitsmaß Avg1 angestrebt wird, der die Differenz der verbalen Repräsentationen der verschiedenen Wortmuster anzeigt, während der Wert des Ähnlichkeitsindexes Avg2, der die Differenz eines entsprechenden Wortmusters über alle Agenten berechnet, möglichst klein werden soll.

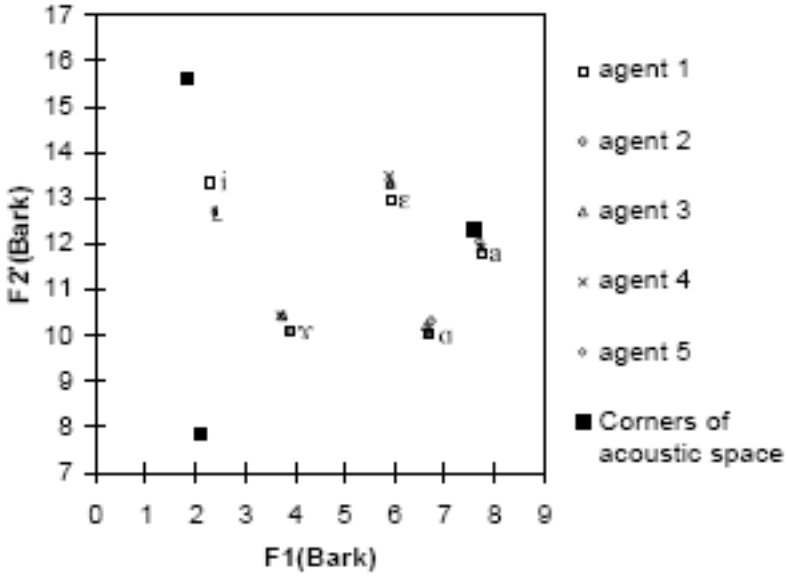


Abb. 36 Die Vokale der Agenten. Quelle: [deB97]

Der Quotient zur Angabe des Kommunikationserfolgs pendelt sich nachdem er mit dem Maximalwert 1.0 zu Beginn der Simulation startet und danach auf 0.7 stark abfällt bei einem Wert um 0.9 ein. (vgl. Abb. 37)

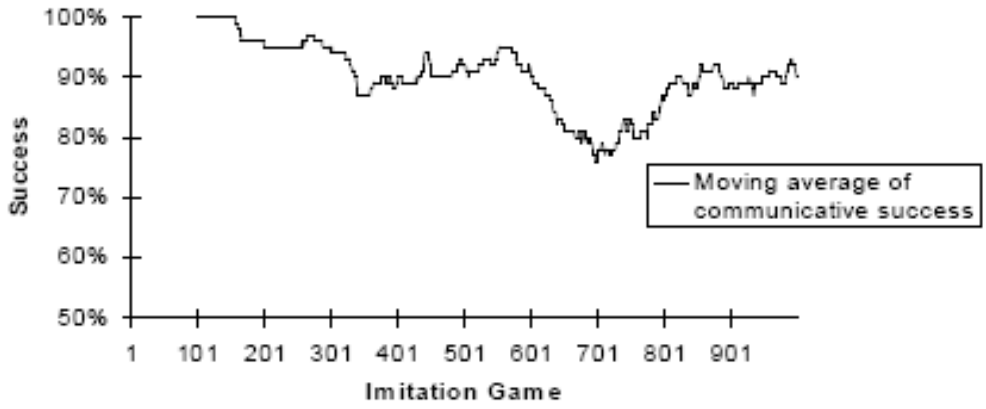


Abb. 37 Der Kommunikationserfolg. Quelle: [deB97]

Dieser Verlauf ist dadurch zu erklären, dass zu Beginn der Simulation die Agenten mit einem leeren Phonemrepertoire initialisiert werden. Da zuhörende Agenten in dieser Phase nur den gehörten Laut reproduzieren müssen und es keinerlei Verwechslungsgefahr mit anderen Phonemen gibt, ist eine erfolgreiche Kommunikation vorherbestimmt. Dies ändert sich jedoch in der nachfolgenden Phase, wenn neue Phoneme entstehen und der Quotient abfällt. Im Laufe der

zunehmenden Konvergenz steigt dieser Wert wieder an und verharrt später bei Werten um 0.9.

Weitere Simulationsdurchläufe mit größeren Populationen erzeugen ähnliche Ergebnisse, wobei die Klarheit der Phonemverteilungen wie erwartet abnimmt. Die Einführung einer offenen Population sowie die Verstärkung des Rauschens verlangsamen die Konvergenz einer Phonemverteilung. Die Einführung einer Altersstruktur bringt die bereits bekannten Resultate hervor.

3.4.4 Das Modell von Smith

Kenny Smith stellte ebenfalls ein Modell der kulturellen Evolution von Kommunikation in einer Gemeinschaft von neuronalen Netzen vor. Das Untersuchungsobjekt dieses Modells ist im Gegensatz zu den vorher genannten nicht eine Ausbildung von Dialekten, sondern es untersucht eine begrenzte Menge von Lernregeln darauf, ob sie zu einem funktionierenden Kommunikationssystem führen können. Da ein solches funktionierendes System die Grundlage einer späteren möglichen Dialektbildung ist, ist die Realisierung für diese Arbeit ebenfalls interessant. Die Ausführungen dieses Abschnitts basieren auf dem Artikel [Smi02].

Der Aufbau der Agenten

Die Agenten werden durch zweischichtige neuronale Netze repräsentiert. Eine Schicht, im Modell N_M genannt, kodiert dabei die Bedeutung einzelner Objekte der Simulationsumgebung während die zweite Schicht, N_S, die Signalschicht darstellt. Beide Schichten sind miteinander durch bidirektionale Verbindungen derartig verknüpft, dass jeder Knoten aus einer Schicht mit jedem Knoten der anderen Schicht verbunden ist. Die Verbindungen eines Agenten werden in der Menge ${}^{o}W$ zusammengefasst.

Innerhalb beider Schichten ist es jeweils nur einem Neuron gestattet aktiv zu sein, sprich einen Aktivierungswert von 1 zu tragen, alle anderen werden mit 0 belegt. Daraus ergibt sich eine Menge von $|N_M|$ möglichen orthogonalen Bedeutungsdarstellungen und $|N_S|$ möglichen Signaldarstellungen.

Wenn G_i der i-te Knoten der Menge N_G und dessen Aktivierungswert a_{G_i} ist, dann entspricht die Bedeutung m_i einem Eingabemuster von N_M bei dem $a_{M_i} = 1$ und $a_{M_{(j \neq i)}} = 0$ ist.

Dies gilt analog für die Signale s_i der Eingabemuster der Menge N_S, wo $a_{S_i} = 1$ und $a_{S_{(j \neq i)}} = 0$.

Die Berechnung der Ausgabemuster innerhalb der neuronalen Netze erfolgt durch die „the winner takes it all"-Strategie, die besagt, dass das Neuron mit dem höchsten Eingabewert den Aktivierungswert 1 erhält, während alle anderen Neuronen einen Aktivierungswert von 0 propagieren. Die Berechnung des Eingabewerts eines jeden Neurons erfolgt durch die nachfolgende Formel

$$q_{Si} = \sum_{j=1}^{j=|N_M|} a_{Mj} w_{Mj,Si}$$

in der $w_{a,b} \in {}^{o}W$ die Verbindungsgewichtung zwischen den Neuronen *a* und *b* ist. Sollten mehrere Neuronen den gleichen errechneten Eingabewert besitzen, so wird zufällig einer der Kandidaten als Gewinner ausgewählt. Abbildung 38 zeigt ein Beispiel eines der hier beschriebenen neuronalen Netze.

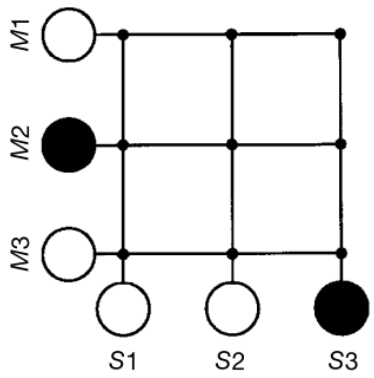


Abb. 38 Das neuronale Netz eines Agenten. Quelle: [Smi02]

Das Lernen innerhalb der neuronalen Netze erfolgt durch eine Regel *W*, die durch ein 4-Tupel $(\alpha, \beta, \chi, \delta)$ definiert werden kann. α spezifiziert dabei wie eine Verbindungsgewichtung w_{ij} konfiguriert werden soll wenn $a_i = a_j = 1$, β wenn $a_i = 1$ und $a_j = 0$, χ wenn $a_i = 0$ und $a_j = 1$ und analog δ wenn $a_i = 0$ und $a_j = 0$.

Die Verbindungsgewichtungen liegen stets im ganzzahligen Intervall [-1;1], sodass sich $3^4 = 81$ verschiedene Lernregeln herleiten lassen.

Die Kommunikation

Eine Kommunikation findet in diesem Modell nur indirekt statt. Innerhalb eines Simulationsdurchlaufs werden alle Agenten mit der gleichen der 81 möglichen Lernregeln initialisiert. Beim Beginn eines Simulationsdurchlaufs werden 100 Agenten initialisiert.

In jedem Simulationsschritt wird ein Agent zufällig entfernt. Darauf generiert jedes Mitglied der Agentenpopulation die Menge der möglichen Bedeutung-Signal-Darstellungen mit Hilfe seines neuronalen Netzes. Ein Rauschen wird dabei jedem Bedeutungs-Signal-Paar mit der Wahrscheinlichkeit $p_n = 0.05$ zugefügt. Als nächstes wir ein neuer ungelernter Agent mit leere Verbindungsgewichtungen initialisiert. Er erhält die Bedeutungs-Signal-Paare dreier Agenten um damit sein eigenes neuronales Netz zu optimieren. In der Folge tritt er als gleichberechtigtes Mitglied der Population bei. Ein Simulationsschritt ist hiermit beendet.

Es ist ersichtlich, dass in diesem Modell keine direkte Kommunikation zwischen zwei Agenten stattfindet. Vielmehr lernen junge Agenten durch Beobachten erfahrener Agenten ein simples Kommunikationssystem. Dies wird in Abbildung 39 verdeutlicht.

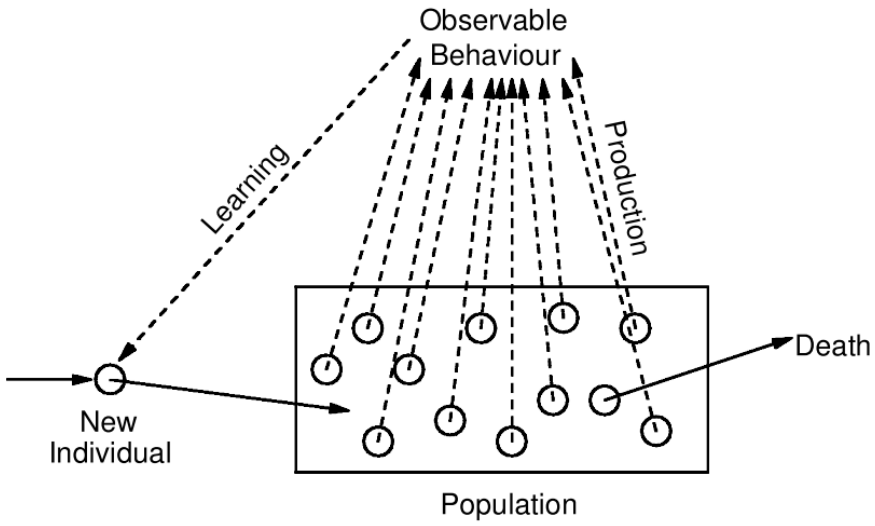


Abb. 39 Kommunikation im Modell von Smith. Quelle: [Smi02]

Die Simulationsumgebung

Die Simulationsumgebung ist in diesem Modell ebenfalls sehr simpel gehalten, da auch hier der Fokus nicht auf einer detailliert gestalteten Simulationsumgebung, sondern auch der Fähigkeit einer Agentenpopulation ein funktionierendes

Kommunikationsmodell zu erlernen liegt. Die Objekte bzw. in diesem Modell die Bedeutungen existieren in der Simulationsumgebung und sind für jeden Agenten in jedem Simulationsschritt ersichtlich. Eine räumliche Anordnung der Agenten findet nicht statt.

Beobachtungen

Die Erfolgsaussichten einer jeden der 81 Lernregeln werden in diesem Modell in drei Stufen unterteilt. In einem ersten Schritt wird jeweils anhand eines Agenten getestet, ob er in der Lage ist, die notwendige Menge von Bedeutungs-Signal-Paaren zu erlernen. Es zeigt sich, dass 50 der möglichen 81 Lernregeln nicht in der Lage sind eine funktionierende Konfiguration der neuronalen Netze herzustellen. Diese Lernregeln werden mit dem Prädikat „-learner“ gekennzeichnet. Die restlichen 31 Lernregeln sind in der Lage innerhalb eines Agenten die notwendige Menge von Bedeutungs-Signal-Paaren zu kodieren. Daher erhalten die das Prädikat „+learner“.

In einem weiteren Schritt wird untersucht, ob und wie sich die vorhandene Einteilung auf Agentenpopulationen auswirkt. Es zeigt sich, dass alle 50 Lernregeln, die auch bei einem einzelnen Agenten nicht zu einem erfolgreichen Lernen führen, ebenfalls kein funktionierendes Kommunikationssystem innerhalb einer Population ausbilden können. Die restlichen 31 unterscheiden sich jedoch stark.

Nur neun der 31 Lernregeln, die bei einem einzelnen Agenten zu einem erfolgreichen Lernen führen sind in einer Gemeinschaft in der Lage ein funktionierendes Kommunikationssystem auszubilden. Diese werden als „+constructur“ deklariert, alle anderen als „-constructor“. Neun der restlichen 22 Lernregeln, die einen einzelnen Agenten zu einem erfolgreichen Lernen verhalten, sind dabei in der Lage, ein durch eine „+constructur“-Lernregel aufgebautes Kommunikationssystem aufrechtzuerhalten, diese sind „+maintainer“-Lernregeln, 13 der 31 Lernregeln sind dazu nicht befähigt, also „-maintainer“. Daraus ergibt sich eine Einteilung der 81 Lernregeln in die vier Untergruppen, die in der nachfolgenden Abbildung ersichtlich ist.

Classification	Number
[– learner, – maintainer, – constructor]	50
[+ learner, – maintainer, – constructor]	13
[+ learner, + maintainer, – constructor]	9
[+ learner, + maintainer, + constructor]	9

Abb. 40 Die Klassifizierung der Lernregeln. Quelle:[Smi02]

3.4.5 Das Modell von Galantucci

Im folgenden Abschnitt wird ein Modell von Galantucci vorgestellt, das als Schnittstelle zwischen einem rein menschenbasierten System und einer Multi-Agenten-Simulation gesehen werden kann. Die Agenten in seinem Modell werden von realen Personen gesteuert, die durch eine räumliche Abschottung jedoch nur sehr begrenzt miteinander kommunizieren können. Es kann daher als empirisches Korrelat zu den hier vorgestellten Modellen angesehen werden. Die Ausführungen dieses Abschnitts basieren auf dem Artikel [Gal04].

Der Aufbau der Agenten

Die Agenten in diesem Modell werden von Menschen gesteuert. Über eine Benutzerschnittstelle sind die Personen in der Lage die Position „ihres“ Agenten in der Simulationsumgebung zu sehen und diese durch Steuertasten zu verändern.
Des Weiteren sind die Agenten, bzw. die steuernden Personen in der Lage eine simple Form der grafischen Kommunikation durchzuführen. Diese wird jedoch im folgenden Unterabschnitt genauer durchleuchtet.

Die Kommunikation

Die teilnehmenden Menschen eines Simulationsdurchlaufs sind räumlich und auch akustisch voneinander abgeschirmt, um nicht in natürlicher Sprache oder durch Gestik und Mimik miteinander kommunizieren zu können. Die einzig mögliche Art der Kommunikation erfolgt über ein Grafiktablett, das mit einem passenden Stift bedient wird. Die Eingaben einer Person sind für alle Agenten innerhalb der Simulationsumgebung sichtbar.

Um das Benutzen bekannter Kommunikationssymbole auszuschließen, wird die Kommunikation durch drei weitere Faktoren erschwert. Zum einen erlischt die Zeichnung bereits sehr kurz nach der Eingabe, des Weiteren bewegt sich das Grafiktablett unter dem vertikal fixierten Eingabestift hinweg, sodass nur eine horizontale Veränderung der Position des Stifts möglich ist. Auf diese Weise ist das Zeichnen bekannter Symbole unmöglich. Abbildung 41 zeigt eine solche Kommunikationsvorrichtung und Beispiele der damit möglichen Kommunikationsmuster.

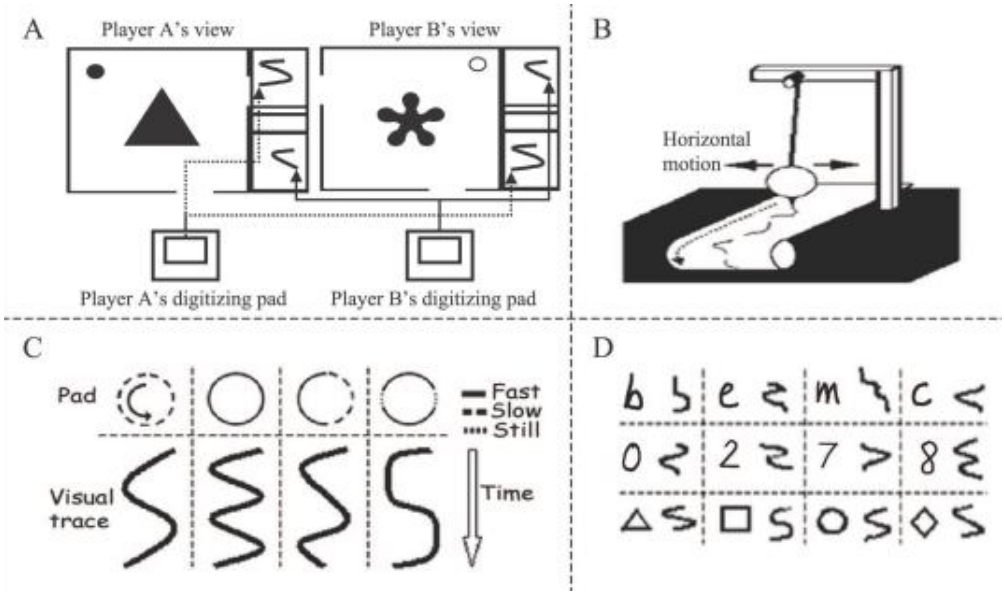


Abb. 41 Die Kommunikationsvorrichtung und mögliche Symbole Quelle:[Gal04]

Die Simulationsumgebung

Die Simulationsumgebung in diesem Modell besteht aus einer Anzahl von verschiedenen Räumen, die durch ein Symbol in der Mitte des Raumes gekennzeichnet werden. Im Basismodell existieren vier Räume, die quadratisch angeordnet sind. Erweiterungen des Modells setzen auf größere Umgebungen, sodass neun oder gar 16 verschiedene Räume existieren. In jedem Raum existieren Türen zu den jeweils benachbarten Räumen, sodass im Basismodell jeder Raum zwei Türen, in den Erweiterungen bis zu vier Türen besitzt.

Die Agenten können jeweils nur die Geschehnisse innerhalb des Raumes visuell wahrnehmen, in dem sie sich gerade befinden. Eine Kommunikation über das Grafiktablett ist im Gegensatz dazu jederzeit möglich. Abbildung 42 zeigt den Aufbau einer Simulation mit den jeweils beschränkten Sichtweisen beider Agenten.

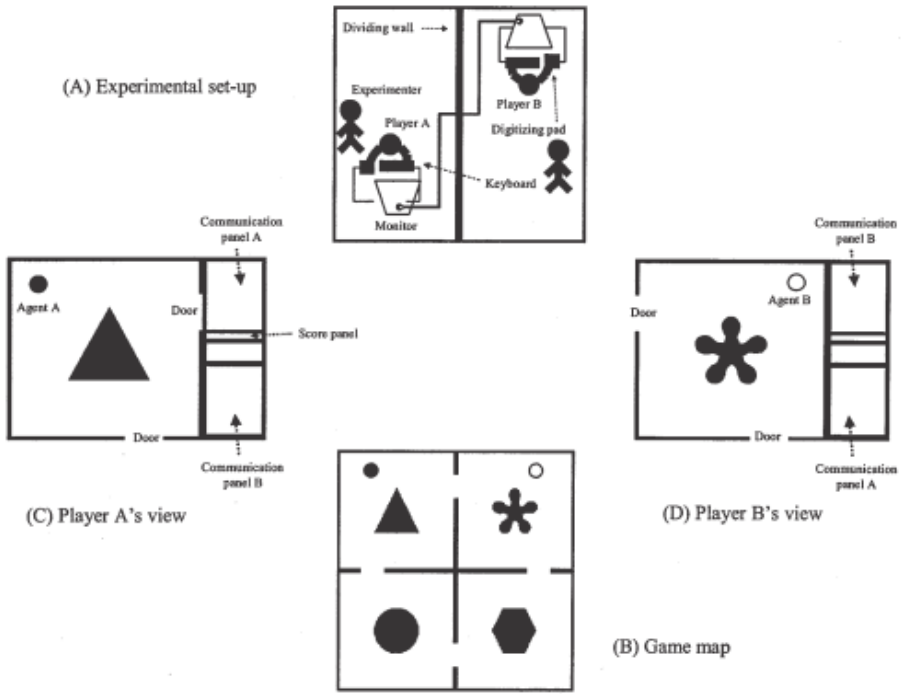


Abb. 42 Der Aufbau der Simulation. Quelle:[Gal04]

Beobachtungen

Zu Beginn eines jeden Simulationsschritts werden zwei Agenten in unterschiedlichen Räumen zufällig positioniert. In dem hier betrachteten Basismodell ist jedem Agenten nur ein Raumwechsel gestattet. Ziel eines Simulationsschritts ist es, dass sich beide Teilnehmer nach den maximal zwei möglichen Raumwechseln innerhalb desselben Raums befinden. Das Ende einer Simulationsrunde wird den Agenten durch das Erscheinen von vier Quadraten innerhalb des Raumes, in dem sie sich befinden, signalisiert. Betreten beide Agenten eines dieser Quadrate gilt der Simulationsschritt als abgeschlossen.

Die Simulation wird mit zehn Versuchspaaren durchgeführt, die so lange spielen, bis sich ein stabiles zuverlässiges Kommunikationsmedium etabliert hat und dadurch jeder Simulationsschritt erfolgreich abgeschlossen wird. Abbildung 43 zeigt den Verlauf des Kommunikationserfolgs der zehn Paare.

Es zeigt sich, dass die Konvergenz zu einer gemeinsamen Sprache keine triviale Aufgabe darstellt. Selbst nach über 160 Minuten Simulationsdauer ist Paar 5 nicht in der Lage erfolgreich zu kommunizieren. Da eine der Personen Zeichen zunehmender Frustration zeigte wird der Versuch abgebrochen. Paar 6 gelingt eine Kommunikation erst nach 152 Minuten und auch nur dank der ausführlichen Hilfe einer Person, die die Simulation beaufsichtigt. Aus diesem Grunde wird auch dieses Paar als nicht erfolgreich angesehen.

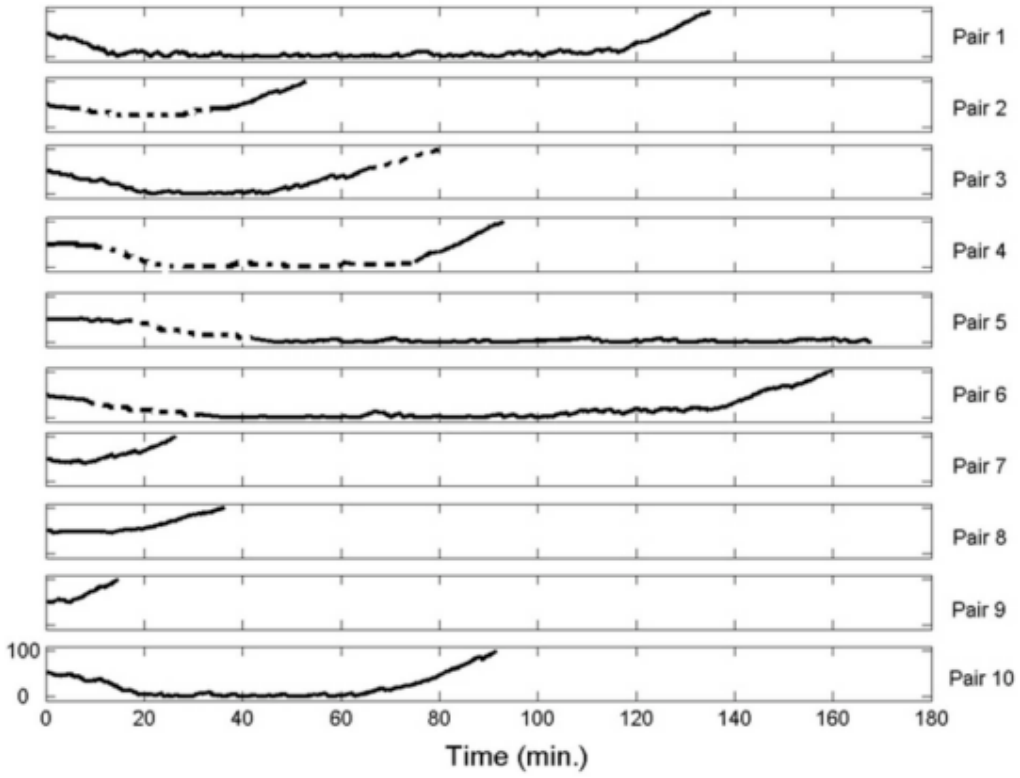


Abb. 43 Der Kommunikationserfolg der zehn Paare. Quelle:[Gal04]

Innerhalb der Analyse der Simulationsdurchläufe stellt sich zuerst die Frage, welche Strategie die Teilnehmer bei der Ausbildung eines Kommunikationssystems hatten. Dabei kann zwischen zwei verschiedenen Strategien unterschieden werden: Sechs der zehn Paare (1,3-6,10) versuchen das Kommunikationssystem durch Versuch und Irrtum zu erlernen. Ein unbekanntes Symbol wird dabei zu Beginn zufällig einem Raum zugeordnet. Treffen sich beide Agenten in einem Raum einigen sie sich im Laufe der Zeit darauf, diesen Raum mit dem Symbol zu bezeichnen. Eine Bezeichnung für einen Raum etabliert sich.

Die zweite und bei Weitem schnellere Möglichkeit eine Menge von Symbol-Raum-Paaren zu erzeugen wird von den restlichen Paaren (2,7-9) angewendet. Diese verzichten zu Beginn der Simulation auf eine schnelle Abfolge von Simulationsschritten. Sie bewegen sich, nachdem sie sich zufällig treffen, durch die Simulationsumgebung und benennen die Räume gemeinsam. Dies wird beispielsweise dadurch gezeigt, dass sich ein Agent, während er auf dem Symbol in der Mitte eines Raumes steht, fortwährend ein Symbol zeichnet, bis seinem Partner klar ist, dass dieses Symbol mit dem Raum assoziiert werden soll. Dies wiederholt sich in allen Räumen. Diese sogenannte „Naming-Tour“ ermöglicht es viel schneller ein erfolgreiches Kommunikationssystem zu etablieren.

Im letzten Schritt werden die Zeichensysteme der einzelnen Paare untersucht. Die nachfolgende Abbildung zeigt dazu die Kommunikationssymbole der an der Simulation teilnehmenden Paare.

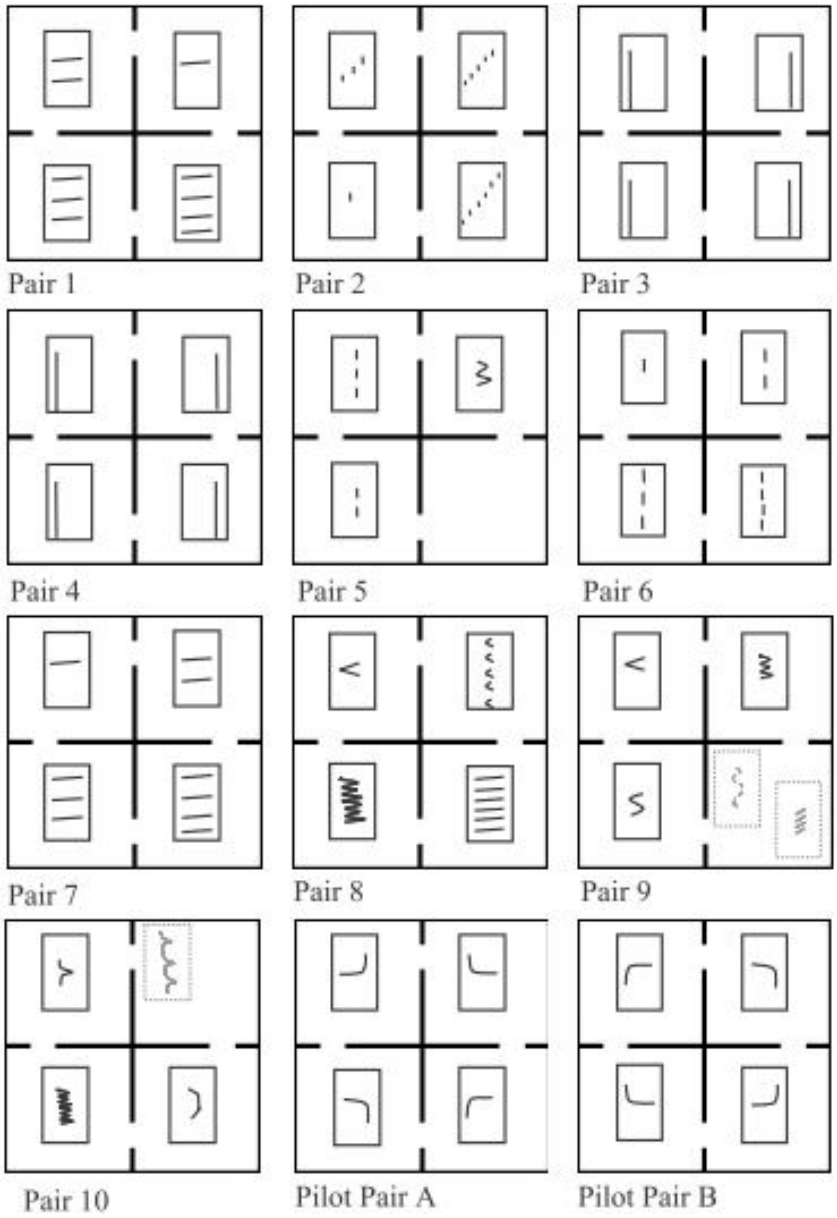


Abb. 44 Die Kommunikationssymbole der Paare. Quelle:[Gal04]

Es lassen sich drei verschiedene Systeme unterscheiden. Drei der zehn Paare (1,6,7) nutzen ein Nummerierungssystem, in dem eine unterschiedliche Anzahl von Strichen, entweder diagonal oder vertikal angeordnet, einen Raum kodieren. Vier Paare (2,8-10) benutzen ein symbolbasiertes System, in dem die Kommunikationssymbole meist den Raumsymbolen nachempfunden sind. Zwei

Paare (3,4) kreieren ein kartenbasiertes System, in dem lediglich die Position der Agenten über längengradähnliche Striche kodiert wird. Somit existieren stets zwei mögliche Aufenthaltsorte eines Agenten. Dies ist aber bei nur vier Räumen weniger problematisch, da, sollten beide Agenten direkt oder nachdem bereits ein Agent den Raum gewechselt hat, den gleichen Längengrad angeben, der zweite Agent immer noch in den Raum des Längengrades gehen kann, in dem er sich gerade nicht befindet und so ein Zusammentreffen beider Agenten immer noch mit maximal zwei Raumwechseln möglich ist.

3.5 Das Modell in DiaLex

In diesem Unterkapitel werden der Weg zu dem in dieser Arbeit genutzten Modell gezeigt, die Gründe für die Erweiterung des Modells von Hutchins und Hazlehurst dargelegt, die erwarteten Ziele dieser Erweiterung und schließlich der Weg zum nachfolgenden Kapitel, das Anforderungsdefinition, Architektur und Entwurf des Modells beinhaltet, bereitet.

3.5.1 Vergleich zu vorherigen Modellen

Die Entwicklung des zugehörigen Modells dieser Arbeit gestaltete sich, wie es bereits aus den bisherigen Ausführungen zu erwarten war, als äußerst schwierig. Nimmt man die menschliche Sprache als Vorbild, so müsste es weit komplexer gestaltet sein, als es alle vorherigen Modelle jemals waren. Alleine die Anforderungen an ein relativ menschennahes Modell scheinen aus heutiger Sichtweise in naher Zukunft unerreichbar. Die Realisierung eines Modells, das eine Kommunikationsmöglichkeit mit der Mächtigkeit von Tiersprachen erzeugt scheint daher eher möglich.

Unbestritten ist, dass Gehirne von Lebewesen, rein biologisch betrachtet, wie riesige neuronale Netze funktionieren. Jedoch erscheinen alleine die Anforderungen an diese unerreichbar. Modernen Schätzungen zufolge besteht ein menschliches Gehirn aus zwischen 100 Milliarden und einer Billion Nervenzellen, die durch 100 Billionen bis zu einer Billiarde Synapsen miteinander verbunden sind. (vgl. [Tho01], S. 13) Ein solches Netz ist selbst mit heutigen Hochleistungsrechnern nicht in angemessener Zeit zu realisieren. Jedoch damit nicht genug: Es müssten Tausende bis Millionen dieser, sich permanent verändernder Netze simuliert werden, die in einer nahezu realistischen Umgebung miteinander interagieren können.

Betrachtet man die möglichen sprachlichen Evolutionsstufen, die in den Abschnitten 2.3.2 und 2.3.3 dieser Arbeit dargestellt werden, so scheinen einzelne Bestandteile der menschlichen Sprache separat künstlich darstellbar. Dies gilt

besonders für das sogenannte „Namengebungsspiel", das Luc Steels als Basisstufe seiner Einteilung sieht. Es existieren zwar auch einfache Modelle der nachfolgenden Stufen, die jedoch alle den Nachteil haben, dass sie nicht auf neuronalen Netzen basieren. Hinzu kommt, dass sie, genau so wie nahezu alle außer den in Unterkapitel 3.4 genannten Modellen, zwar die Ausbildung eines Sprachbestandteils zeigen, dieser jedoch, nachdem er sich ausgebildet hat, unveränderlich existiert. Erzeugte Wörter bleiben so im Laufe der Zeit identisch oder aber eine entstandene Syntax verändert sich nicht. Dies entspricht, wie in den Unterkapiteln 2.4 und 2.5 gezeigt wird, nicht der Realität. Sprache verändert sich im Laufe der Zeit, dieser Effekt kann und darf bei linguistischen Modellen nicht vernachlässigt werden.

Grenzt man die vorhandenen Modelle unter den beiden Gesichtspunkten Sprachveränderung und Einsatz von neuronalen Netzen zur Darstellung der Agenten ein, so verbleiben die in Unterkapitel 3.4 genannten Modelle. Hutchins und Hazlehurst liefern dabei das mächtigste Modell. Die Kapazitäten der neuronalen Netze der Agenten liegen weit über den Möglichkeiten der anderen Modelle. Abstriche müssen jedoch bei der Simulationsumgebung gemacht werden, da Distanzen zwischen den Agenten nicht über eine räumliche Darstellung kreiert werden. Vielmehr werden in einer Erweiterung des Modells die Agenten derart implementiert, dass sie sich den Gesprächspartner mit dem ähnlichsten Wortschatz aussuchen, was in diesem Modell zur einzigen „Dialektbildung" führt. Es ist jedoch fraglich, ob in diesem Fall von Dialekten gesprochen werden kann. Vielmehr erlernen in dieser Variante mehrere, bereits nach den ersten Simulationsschritten separierte Gruppen unabhängig voneinander jeweils einen Wortschatz. Dialekte sollten jedoch viel mehr durch das Weiterentwickeln und Verändern einer ursprünglich gemeinsamen Sprache in räumlich abgegrenzten Gruppen entstehen. Hutchins und Hazlehurst hatten sich vielmehr, vielleicht ohne es zu wollen, auf Soziolekte fokussiert, die in der realen Welt eine weit weniger starke Abgrenzung der Gruppensprachen erwirken.

Das Modell von de Boer liefert ohne Rauschen sehr ähnliche Ergebnisse. Vokale werden ausgebildet, die über die Zeit konstant bleiben. Fügt man dem Modell jedoch ein Rauschen hinzu, so verändern sich die Lautäußerungen der Individuen, eine Dialektbildung ist demnach zumindest vorstellbar. Defizite gibt es hier jedoch auch bei der Darstellung der Distanzen zwischen den einzelnen Agenten. Livingstone und Smith nutzen in ihren Modellen ebenfalls neuronale Netze. Sie sind jedoch weit simpler als die von Hutchins und Hazlehurst verwendeten Exemplare. Der zweischichtige Aufbau und das simple Kodieren der Signale in ihnen macht das Trainieren der Netze anhand komplexerer Algorithmen unnötig. Hinzu kommt, dass eine Ähnlichkeit zwischen verschieden Wortdarstellungen nicht sinnvoll darstellbar ist. Stets wird exakt ein Neuron mit dem Wert +1 belegt, während alle anderen auf -1 bzw. 0 gesetzt werden. Alle Worte sind sich somit zu jeder Zeit gleich ähnlich. Die Möglichkeit einer stetigen Wortveränderung über einen Zeitraum ist somit nicht gegeben.

Die Simulationsumgebung mit ihren Attributen „Nachbarschaftsgröße“ und „Rauschen“ ist bei Livingstone weit besser realisiert als es in den vorherigen Modellen der Fall war. Kritisch zu sehen ist jedoch die Tatsache, dass die Agenten nur in einer relativ simplen Reihe angeordnet sind, eine wirkliche Separierung von einzelnen Gruppen ist somit nicht möglich, obwohl dies der Realität recht nahe käme. Eine Gruppenbildung würde temporäre Lücken in der Reihe der Agenten bedeuten, was jedoch im Modell nicht vorhergesehen ist. Das permanente Verändern der Dialektgrenzen in diesem Simulationsmodell ist ein weiterer zu bemerkender Nachteil. Interessanter wäre es zu untersuchen, was mit dem Wortschatz einzelner Agentenuntergruppen mit fortlaufender Aufteilung geschieht, wobei dies, auch bedingt durch die simplen neuronalen Netze, nicht viele Ergebnisse liefern könnte. Die Modelle von Smith und de Boer verzichten vollends auf eine Positionierung der Agenten.

Das Modell von Galantucci unterscheidet sich stark von den anderen genannten Modellen. Die Agenten in seinem Modell werden direkt von Menschen gesteuert, die jedoch in ihrer Wahrnehmung und der Möglichkeit der Kommunikation stark eingeschränkt sind. Daher ist dieses Modell eher als Schnittstelle zwischen rein computerbasierten Modellen und menschlichen oder tierischen Sprachen zu sehen und findet als empirisches Korrelat eine Berechtigung in der Liste der Simulationsmodelle.

Ordnet man alle Modelle in die aus den Abschnitten 2.3.2 und 2.3.3 dieser Arbeit bekannten sprachlichen Evolutionsstufen ein, so zeigt sich, dass Modelle, die auf neuronalen Netzen basieren oder eine Dialektbildung erwirken sollen, noch nicht über das sogenannte „Namengebungsspiel“ hinausgehen. Die Stärken im Modell von Hutchins und Hazlehurst liegen dabei in der Architektur der Agenten und der Erzeugung des Wortschatzes der Agenten während im Modell von Livingstone und de Boer ein Störfaktor, das Rauschen, in die Kommunikation Einzug hält. Als weiterer Störfaktor im Modell von Livingstone kann die Einführung einer offenen Population gesehen werden. Als „Kinder“ erzeugte neue Agenten wirken störend auf den vorhandenen Wortschatz, da sie möglicherweise ohne einen komplett ausgebildeten Wortschatz als „Erwachsene“ andere „Kinder“ unterrichten. Weitere Störfaktoren wären jedoch in allen Modellen wünschenswert.

Im Bezug auf die räumliche Anordnung der Agenten liegt das Modell von Livingstone vorne, wobei in dieser Hinsicht auch hier noch eine Verbesserung zu erhoffen wäre. Zusätzliche Erweiterungen wie eine Einführung einer relativen Häufigkeit von Objekten der Simulationsumgebung oder mögliche Veränderungen der Agentenanordnung während der Laufzeit sind ebenfalls wünschenswert, um reale Situationen der Sprachentwicklung des Menschen nachzustellen.

Basierend auf dieser Ausgangslage erscheint es zweckmäßiger, ein weiteres Modell eines „Namengebungsspiels“ mit den Stärken der vorhandenen Modelle zu erstellen und weitere wünschenswerte Erweiterungen zu implementieren. Erkenntnisse zur Dialektbildung auf höheren sprachlichen Kommunikationsstufen

können erst sinnvoll gewonnen und verstanden werden, wenn sie auf der einfachsten, der lexikalischen Ebene vollzogen wurden.

3.5.2 Agenten und Sprache

Die Agenten des Modells dieser Arbeit orientieren sich am Aufbau der Agenten im Modell von Hutchins und Hazlehurst in [HH95]. Wie bereits erläutert, erscheint der Nutzen eines Autoassociator Netzes (siehe Abschnitt 3.3.1) in diesem Zusammenhang am sinnvollsten, da das Netz auf diese Weise durch den Backpropagation of Error Algorithmus (vgl. Abschnitt 3.3.4) im Bezug auf das Erlernen von Symbol-Wort-Paaren optimiert werden kann. Gerade im Bezug auf eine Wortveränderung und eine einhergehende Dialektbildung sollte der Streckfaktor der sigmoiden Funktion, welche zur Berechnung der Aktivierungswerte der Netzneuronen benötigt wird, mit $T = 1$ gewählt werden.

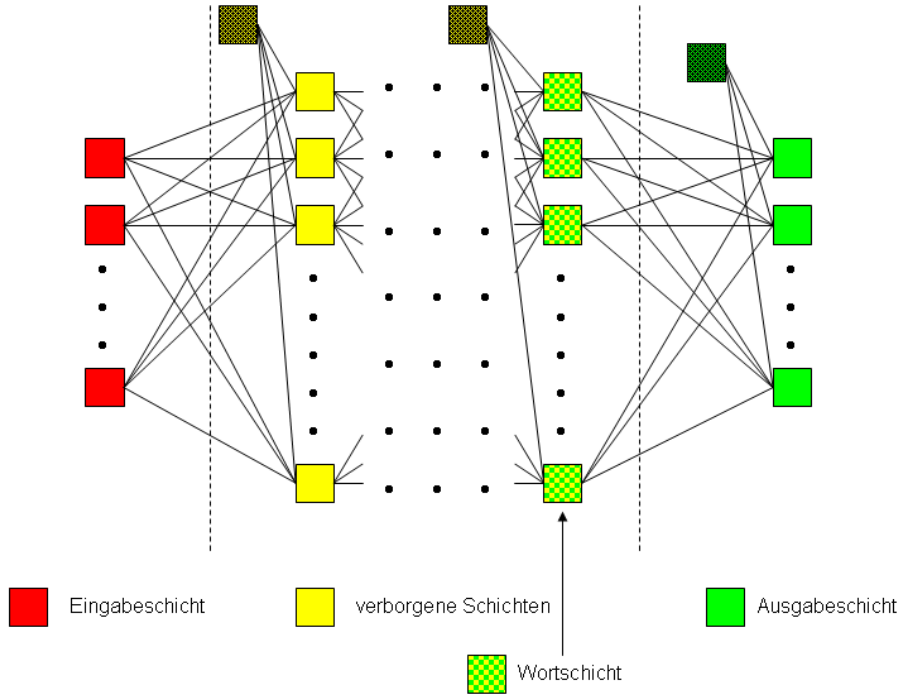


Abb. 45 Das neuronale Netz der Agenten

Durch eine zu steile Funktion wären die Übergänge der einzelnen Aktivierungswerte zu abrupt und würden demnach einer stetigen Dialektbildung entgegenwirken. Neben den beiden gleich großen Ein- und Ausgabeschichten sollen die neuronalen Netze des Modells aus zwischen 1 und n verborgenen Schichten bestehen, wobei erfahrungsgemäß mehr als zwei verborgene Schichten keine Veränderung mehr mit sich bringen. Die hinterste verborgene Schicht repräsentiert dabei die Wortschicht, während die Eingabeschicht das reale Abbild

eines Objekts im Auge und die Ausgabeschicht das gedankliche Abbild des gesehenen Objekts darstellen. Sie sind somit die beiden Ausprägungen von Ferdinand de Saussures Signifikat, während die Wortschicht den im Hirn kodierten Signifikanten repräsentiert. Zu Beginn eines Simulationsdurchlaufs können die Verbindungsgewichtungen der neuronalen Netze entweder mit zufälligen Werten im Intervall [-0.5; 0.5] oder aber mit Nullwerten initialisiert werden. In der nachfolgenden Abbildung wird der Aufbau der neuronalen Netze der Agenten illustriert.

Die Aktivierungswerte der Wortschicht des neuronalen Netzes eines Agenten (ein zweidimensionales Feld reeller Zahlen im Intervall [0.0; 1.0]) nach der Durchführung des Feedforward Algorithmus mit einer visuellen Szene als Eingabe ist dabei die verbale Repräsentation eines Wortes. Das Format der verbalen Repräsentation ist wegen der Vollvermaschung zweier benachbarter Schichten des neuronalen Netzes dabei beliebig, die Fähigkeiten eines Netzes werden alleine durch die Anzahl der Neuronen einer Schicht ausgedrückt.

Auf ein Matching hin zu Buchstaben oder Phonemen während der Agentenkommunikation wird bewusst verzichtet, da dies, aufgrund des unstetigen, nicht linear abbildbaren Werteraums, zu einem hemmenden Faktor in der Sprachveränderung werden könnte.[14]

Vielmehr aber soll die verbale Repräsentation als ein Ortsvektor in einem n-dimensionalen Wörter-Raum verstanden werden, der die Lage eines Wortes definiert. Durch eine Visualisierung des Wortvektors, in dem jede reelle Zahl eines Wortes als ein Grauwert dargestellt wird, kann ein Wort ebenfalls als ein „phonetisches Schriftzeichen“ aufgefasst werden, ähnlich den Schriftzeichen der chinesischen Sprache, die eine Wortsilbe kodieren. Abbildung 46 zeigt vier Beispiele für visualisierte Formen von Wortvektoren.

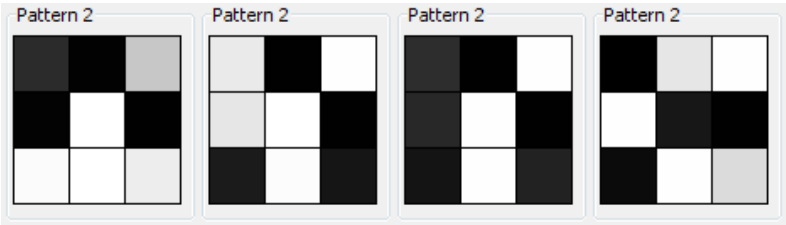


Abb. 46 Visualisierte Wortvektoren

[14] Um während der Auswertung der Simulationsdurchläufe eine benutzerfreundliche Darstellungsweise der verbalen Repräsentationen innerhalb DiaLex zu ermöglichen, sind neben den originalen reellen Zahlenwerten ebenfalls verschiedene grafische und eine literale Darstellungsform wählbar. (vgl. Abschnitt B.2.3) Diese haben jedoch keinen Einfluss auf die Kommunikation der Agenten.

Der Übergang der Kodierung von einem gesamten Wort hin zu einer Wortsilbe wäre durch die Nutzung einer konstanten Anzahl von Silben zu erreichen. In der chinesischen Sprache beispielsweise bestehen die meisten Wörter aus exakt zwei Schriftzeichen. Eine variable Wortlänge im Bezug auf Phoneme erscheint also im Bezug auf die Ausbildung einer Hochsprache von untergeordneter Bedeutung zu sein. Das Werteintervall einer reellen Zahl der Wortschicht könnte in diesem Falle als eine Art Sonoritätsintervall (vgl. Abbildung 9) aufgefasst werden, in dem, bei stetiger Veränderung eines Werts innerhalb eines Simulationsdurchlaufs, eine vereinfachte lineare Veränderung eines Lauts ähnlich der Sonoritätshierarchie dargestellt wird.

3.5.3 Die Simulationsumgebung

Die Simulationsumgebung soll aus einem beliebig großen zweidimensionalen Feld bestehen, auf dem die Agenten frei positioniert werden können. Auf diese Weise können eine räumliche Abgrenzung und Bewegung von Agenten oder Agentengruppen simuliert werden, da die räumliche Distanz zweifelsohne eine starke Rolle bei der Sprachentwicklung spielt. Eine große Distanz in Verbindung mit der fehlenden Möglichkeit der Kommunikation führt über kurz oder lang stets zu einer Sprachspaltung, während das plötzliche räumliche Zusammentreffen zweier verschiedener Sprachen oft zu so genannten Misch- oder Kreolsprachen führt.

In dieser zweidimensionalen Welt der Agenten existiert eine Menge visueller Szenen, deren relative Häufigkeit definiert werden kann. Abbildung 47 zeigt als Beispiel vier simple visuelle Szenen. Die Ergebnisse der Untersuchungen in [PAM07] haben gezeigt, dass sich Wörter mit einer hohen Verwendungsrate in der realen Welt als stabiler erweisen als weniger häufig gebrauchte Wörter. Dieser Effekt soll in dem Modell dieser Arbeit ebenfalls untersucht werden können. Jede visuelle Szene kann dabei an jeder beliebigen Stelle der Simulationsumgebung auftauchen und die Agenten zu einem Kommunikationsversuch verleiten.

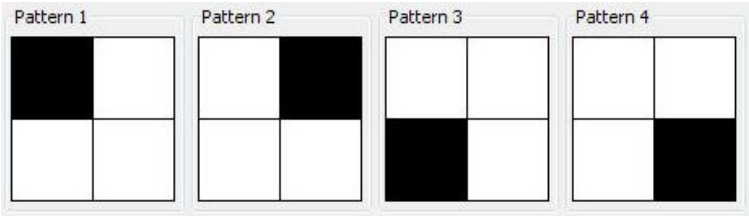


Abb. 47 Visuelle Szenen der Simulationsumgebung

3.5.4 Die Kommunikation

Die Kommunikation innerhalb eines Simulationsschritts läuft folgendermaßen ab: Es werden exakt zwei Agenten der Gemeinschaft ausgewählt, ein Sprecher und ein Zuhörer. Der Sprecher wird entweder zufällig oder aber in chronologischer Reihenfolge ausgewählt, der Zuhörer entweder zufällig oder aber durch eine, auf entweder euklidischer oder aber Manhattan-Distanz basierender Wahrscheinlichkeit des Zusammentreffens. Wechsel dieses Verfahrens sollen auch innerhalb einer laufenden Simulation möglich sein. Durch die Umstellung von zufälliger Zuhörerauswahl auf ein distanzbasiertes Verfahren in Verbindung mit den entsprechenden Koordinaten der Agenten kann beispielsweise eine Aufteilung der Agentengemeinschaft in Untergruppen realisiert werden. Um größere Agentengruppen so zu behandeln, als ob sie direkte Nachbarn wären, soll eine Nachbarschaftsgröße definiert werden. Dadurch besitzen alle Agenten, die sich innerhalb dieses Radius befinden, eine gleich hohe Wahrscheinlichkeit des Aufeinandertreffens wie direkte Nachbarn.

Ebenfalls wird genau eine visuelle Szene zufällig oder zufällig basierend auf ihrer relativen Häufigkeit, ausgewählt. Diese visuelle Szene dient beiden Agenten als Stimulus auf ihrer Eingabeschicht. Sie führen mit ihren Eingabewerten die Forward Pass Berechnung durch und kreieren so ihre individuellen, der Szene zugehörigen, Wörter. Basierend auf den Unterschieden der Werte auf der Ein- und Ausgabeschicht und der verbalen Repräsentation des Sprechers optimiert der Zuhörer in der Folge sein neuronales Netz mit Hilfe des Backpropagation of Error Verfahrens, sodass er seine Fähigkeit, das Eingabemuster zu reproduzieren, verbessert und gleichzeitig seine Wortrepräsentation der des Sprechers annähert.

In einer weiteren Variante soll es in dem Modell auch möglich sein, dass beide ausgewählten Agenten sowohl Sprecher als auch Zuhörer sind und somit beide nicht nur die Werte der Ausgabeschicht der Eingabeschicht angleichen, sondern auch ihre verbale Repräsentation der des jeweils anderen anpassen.

3.5.5 Die Störfaktoren

Die bei Weitem wichtigste Neuerung des Modells ist die Einführung von fünf Störfaktoren, die eine Dialektbildung auslösen oder beeinflussen können. Sie orientieren sich dabei an den biologischen Eigenschaften menschlicher Individuen und an den Gegebenheiten der realen Welt. Die Simulationsergebnisse sollen Erkenntnisse darüber bringen, welche der hier genannten Faktoren eine Dialektbildung der Sprachen hervorrufen kann, in welchem Intervall dies geschieht und welche Wechselwirkungen die Störfaktoren miteinander haben. Dies wäre besonders dann interessant, wenn sich der ein oder andere Faktor stabilisierend auf einen anderen auswirken könnte.

Als erster Faktor wird eine Sterbewahrscheinlichkeit der Agenten eingeführt, der die Wahrscheinlichkeit angibt, mit der ein Agent in einem jeden Simulationsschritt „stirbt“ und dabei durch einen untrainierten neuen Agenten ersetzt wird. Eine konstante Agentenanzahl ist dabei bewusst vorgegeben, damit keine Effekte durch eine schwankende Populationsgröße ausgelöst werden können. Dieser Faktor entspricht dem biologischen Prozess des Sterbens bzw. der Geburt eines Individuums.

Der zweite Faktor ist das Abschwächen bzw. das Verwischen der Erinnerung eines Agenten. Dieser Störfaktor setzt direkt am neuronalen Netz der Agenten an und multipliziert jede Verbindungsgewichtung mit einem Faktor *a*, der durch einen zufällig erzeugten Wert der maximalen Abschwächung *m* bestimmt wird. a liegt dabei im Intervall [1-*m*; 1+*m*]. Eine fortdauernde Veränderung der Verbindungsgewichtungen mit diesem Faktor führt im Laufe der Zeit zu einem fortlaufenden Vergessen der gespeicherten Informationen eines neuronalen Netzes, ähnlich dem des menschlichen Vergessens. Dieser Faktor ist damit der Gegenspieler zum Backpropagation of Error Lernalgorithmus.

Als dritter Störfaktor ist ein Rauschen während der Perzeption der visuellen Szene vorgesehen. Ein Wahrscheinlichkeitswert *p* bestimmt mit welcher Wahrscheinlichkeit ein Feld des Eingabemusters mit einem zufälligen Wert im Intervall [0; 1] anstelle seines Ursprungswertes belegt wird. Dieser Faktor entspricht der realen Welt derart, dass auch die visuelle Wahrnehmung niemals störungsfrei stattfinden kann. Entfernung, Betrachtungswinkel oder andere, das Objekt überlappende Gegenstände, sind hier als Beispiele zu nennen.

Der vierte Faktor ist das Äquivalent des vorangegangenen Faktors während der Kommunikation der Agenten. Hier wird auf gleiche Weise ein Wert des Wortmusters mit einem Zufallswert überschrieben und somit die Kommunikation gestört. Dies entspricht sämtlichen akustischen Störsignalen bei der menschlichen Kommunikation in der realen Welt.

Der letzte Störfaktor entspricht einer Komprimierung des Wortmusters basierend auf einer Umwandlung des Musters mit Hilfe der *diskreten Kosinustransformation* gefolgt von einer linearen *Quantisierung*. Im Anschluss daran werden Faktoren, die unter einem zu deklarierenden Schwellenwert liegen, herausgefiltert, da sie einen zu geringen Einfluss auf die Wortbildung haben. Ähnliche Kompressionsverfahren werden ebenfalls bei der Umrechnung von Audiosignalen in ein mp3 - Format oder dem jpg - Format bei Bilddaten angewandt.[15] Dieser Störfaktor lässt sich mit der variablen Wortlänge der menschlichen Sprache vergleichen. Es ist sowohl denkbar, dass er eine Dialektbildung durch ein Rauschen hervorruft als auch, dass er eine

[15] Das mp3-Format setzt dabei jedoch auf die ähnliche Fouriertransformation anstatt der Kosinustransformation. Bei der Umrechnung einer Bilddatei in das jpg –Format sind weitere Schritte vorgesehen, wie beispielsweise eine Farbumrechnung und eine Tiefpassfilterung. Dies ist in dieser Arbeit nicht notwendig.

Stabilisierung des Lexikons bewirkt, da er sich wie eine verkleinerte Wortschicht der Agenten auswirken könnte und somit nicht genug Raum für Innovationen lassen könnte. Die Funktionsweise des Störfaktors, bestehend aus einer zweidimensionalen diskreten Kosinustransformation und einer nachfolgenden Quantisierung wird im Folgenden genauer beschrieben:

Das zweidimensionale Muster eines Wortes wird mit der diskreten Kosinustransformation derart umgerechnet, dass sein Muster als Linearkombination mehrerer Kosinusfunktionen dargestellt wird. Die Anzahl der einzelnen Funktionen entspricht der Menge der Bildpunkte und ihre Frequenzen leiten sich dabei von den Dimensionen des Ausgangsmusters ab. Bei einem Wortmuster mit einer horizontalen Ausdehnung *I* und einer vertikalen Ausdehnung *J* leiten sich die Frequenzen wie folgt ab:

$$F_{x,y}(hor,ver) = \left(\frac{1}{i},\frac{1}{j}\right)$$

Die Werte der Matrix nach der Berechnung geben den Streckfaktor bzw. die Höhe der einzelnen Kosinusfunktionen an. Sie werden mit folgender Formel errechnet, die eine Umrechnung eines Musters mit den Ausdehnungen *I* und *J* in die gleich große Darstellung der Streckfaktoren der Kosinusfunktionen, deren Ausdehnung der besseren Unterscheidungsmöglichkeit halber als *X* und *Y* bezeichnet werden. Der Funktionswert an der Stelle (*x,y)* errechnet sich dabei wie folgt:

$$F_{x,y} = \frac{2 * C(x) * C(y)}{I * J} * \sum_{i=o}^{I-1}\sum_{j=0}^{J-1} f_{i,j} * \cos\left(\frac{\pi}{I} * x * \left(i+\frac{1}{2}\right)\right) * \cos\left(\frac{\pi}{J} * y * \left(j+\frac{1}{2}\right)\right)$$

Die Funktion *C(n),* mit der die Kanten eines Musters abgeschwächt werden, ist dabei wie folgt definiert:

$$C(n) = \begin{cases} 1, n \neq 0 \\ \frac{1}{\sqrt{2}}, n = 0 \end{cases}$$

Diese Umformung ist bis an diese Stelle verlustfrei und reversibel. Da auf diese Weise jedoch keine Komprimierung eines Musters gelungen ist, wird die neu errechnete Matrix quantisiert. Während das jpg - Format hier auf eine Standardmatrix der Größe 8x8 setzt, erscheint der Einsatz hier nicht sinnvoll, da die meisten Wortmuster deutlich kleiner sein werden. Aus diesem Grund wird vor der Quantisierung eine Matrix errechnet, in der die Gewichtungswerte von der Stärke *1.0* an Position (*0,0)* in beide Richtung linear abfallen bis sie an der Koordinate (*X-1,Y-1*) eine Stärke von *0.0* erreichen.

Ihre Werte $f_{x,y}$ werden mit folgender Formel berechnet:

$$f_{x,y} = 1 - \frac{1}{2} * \left(x * \frac{1}{X} + y * \frac{1}{Y} \right)$$

Die vorher errechneten Werte der Kosinusfunktionen werden nun mit den entsprechenden Werten der Quantisierungsmatrix multipliziert, wobei Ergebnisse unter einem Schwellenwert S nicht mehr berücksichtigt und auf *0.0* gesetzt werden. Auf diese Weise fallen Faktoren, die einen kleinen Einfluss an dem Aussehen eines Musters haben, weg. Diese Umformung ist selbsterklärend verlustbehaftet und daher irreversibel.

Zuletzt wird die quantisierte Matrix wieder in ihr ursprüngliches Format umgewandelt. Dies geschieht mit Hilfe der Umkehrfunktion der diskreten Kosinustransformation, der *inversen diskreten Kosinustransformation*. Die Funktionswerte des Wortpatterns werden dabei wie folgt berechnet:

$$f_{i,j} = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \frac{2 * C(x) * C(y)}{X * Y} * F_{x,y} * \cos\left(\frac{\pi}{I} * x * \left(i + \frac{1}{2} \right) \right) * \cos\left(\frac{\pi}{J} * y * \left(j + \frac{1}{2} \right) \right)$$

Die so umgeformten Wortmuster werden trotz gleicher Endgröße aus weniger Informationen gebildet als die Ursprungsmuster und sind somit deren verlustbehaftete komprimierte Nachbildungen.

3.5.6 Messgrößen

Zur Analyse der Simulationsdurchläufe werden folgende Indikatoren herangezogen:

Der Fehlerquotient Avg1, der bereits aus dem Simulationsmodell von Hutchins und Hazlehurst bekannt ist, (vgl. Abschnitt 3.4.1) zeigt an, wie stark sich die verbalen Repräsentationen der Agenten für eine visuelle Szene von denen der anderen visuellen Szenen unterscheiden. Die errechneten Werte liegen im Intervall [0; 1], wobei der Wert 0 die exakt gleichen Wortmuster anzeigen würde, während der Wert 1 auf komplett gegensätzliche Wortrepräsentationen schließen lassen würde. Es ist also erstrebenswert, möglichst hohe Werte bei diesem Indikator zu erlangen, wobei bereits in [FK07] gezeigt wurde, dass bei größeren Simulationen Werte von 0.5 im Modell von Hutchins und Hazlehurst auch ohne Störfaktoren bereits als gute Werte angesehen werden müssen. Daher ist die Aussagekraft dieses Indikators weniger stark zu bewerten, als die der folgenden Messgrößen.

Der Quotient Avg2, der ebenfalls im Modell von Hutchins und Hazlehurst definiert wurde, indiziert den Unterschied aller verbalen Repräsentationen eines Agenten mit

den entsprechenden Wortmustern der anderen Agenten. Der Wertebereich ist ebenfalls das Intervall [0; 1]. Kleine Werte deuten auf eine hohe Ähnlichkeit der Agentenwortschätze hin, während große Werte auf starke Unterschiede hinweisen. Eine gewichtige Neuerung des Modells wird es sein, den Avg2-Wert, wie auch den Avg1-Wert, nicht nur über die komplette Agentengemeinschaft zu errechnen und in Form eines Graphen darstellbar zu machen, sondern ihn ebenfalls für Agenten-Untergruppen zu berechnen und darzustellen. Ein Verharren der Avg2-Werte der Untergruppen auf niedrigem Niveau während der Avg2-Wert der gesamten Agentengemeinschaft in einem Simulationsdurchlauf steigt, würde eine Sprachspaltung in Dialekte nachweisen. Eine weitere Neuerung wird die agentenbasierte Berechnung beider Indikatoren in Form einer Kreuztabelle sein, um einen detaillierten Agent-zu-Agent-Vergleich zu gewährleisten. So sollte eine Gruppenzugehörigkeit der Agenten gezeigt werden können.

Im Laufe der ersten Testläufe des Modells wurde ersichtlich, dass Avg1 und Avg2 selbst zusammen nur eine sehr begrenzte Aussagekraft über die Entwicklung der Wortschätze bzw. Lexika der einzelnen Agenten haben. Selbst kleinste Unterschiede zwischen den verbalen Repräsentationen einzelner visueller Szenen reichten den Agenten für eine Unterscheidung aus, während auch kleinste Unterschiede in den Wortschätzen der einzelnen Agenten nicht mehr zu einem Verständnis zu reichen schienen. Daher wird das Modell um den Indikator des gegenseitigen Verständnisses erweitert. Zur Berechnung des gegenseitigen Verständnisses muss die Formel der Differenzberechnung in Wortmustern aus Abschnitt 3.4.1 verallgemeinert werden, damit sie auf beliebige Muster wie Eingabeszenen, Wortmuster oder errechnete Ausgabemuster angewandt werden kann. Somit ergibt sich folgende, allgemeinere Definition der Differenzmetrik *d*, die die Differenz zwischen den beiden Mustern $p_1(t)$ und $p_2(t)$ errechnet. Die beiden Muster sind dabei Vektoren reeller Werte der Länge γ zum Zeitpunkt *t*.

$$d(p_1(t), p_2(t)) = \sqrt{\frac{\sum_{k=1}^{\gamma}(r_1^k - r_2^k)^2}{\gamma}}$$

Weiterhin sei $b_{j,s}(t)$ definiert als die errechnete Ausgabeschicht des Agenten *j* aus einem Wort *s* zum Zeitpunkt *t*. Somit ergibt sich, unter Berücksichtigung der weiteren Definitionen aus Abschnitt 3.4.1, dass

$$b_{j_2, s_{i, j_1}(t)}(t)$$

die errechnete Ausgabeschicht des Agenten j_2 zum Zeitpunkt *t* aus dem Wort ist, das Agent j_1 bei Betrachtung der visuellen Szene *i* ebenfalls an Zeitpunkt *t* in seiner Wortschicht berechnete.

Der Quotient des gegenseitigen Verständnisses ist dabei wie folgt definiert:

$$Understanding(t) = \frac{\sum_{i=1}^{m} \left(\frac{\sum_{(j_1, j_2) \in P_2(n)}^{n^2-n} f\left(i, b_{j_2, s_{i,j_1}(t)}(t)\right)}{(n^2 - n)} \right)}{m}$$

$P_2(n)$ ist die Menge aller geordneten Paare ganzzahliger, positiver Zahlen von 1 bis *n*, $n^2 - n$ gibt die Länge dieser Menge $P_2(n)$ an.

Die Definition der in *Understanding(t)* enthaltenen Funktion f lautet wie folgt:

$$f\left(i, b_{j_2, s_{i,j_1}(t)}(t)\right) = \begin{cases} 1 : d(i, b_{j_2, s_{i,j_1}(t)}(t)) < d(a, b_{j_2, s_{i,j_1}(t)}(t)) \forall a \in I : a \neq i \\ 0 : \exists a \in I : d(i, b_{j_2, s_{i,j_1}(t)}(t)) > d(a, b_{j_2, s_{i,j_1}(t)}(t) \end{cases}$$

Ihr Rückgabewert ist 1, falls die Differenz zwischen dem erzeugten Abbild und dem Eingabemuster *i* kleiner ist, als die Differenz zwischen dem Abbild und allen anderen Eingabemustern innerhalb der Menge *I* der Eingabemuster. Existiert ein Eingabemuster, das der errechneten Ausgabeschicht des Agenten ähnlicher ist, so gibt sie den Wert 0 zurück.

Bei der Berechnung wird der Kommunikationserfolg aller Agenten untereinander über alle visuellen Szenen errechnet. Ordnet ein Agent das aufgenommene Wort des anderen Agenten in einer Kommunikation der gleichen visuellen Szene zu, die der Gesprächspartner zur Erzeugung des Wortes genutzt hat, so ist die Kommunikation erfolgreich. Ebenso wie die Indikatoren Avg1 und Avg2 soll dieses Maß ebenfalls gruppenbasiert errechnet werden und in Form einer Kreuztabelle zum Auffinden von Gruppengemeinschaften verglichen werden.

Als letztes Maß wird das Lexikon einer Agentengemeinschaft gebildet, das die mittleren Wörter zu jeder visuellen Szene beinhaltet. Ein Vergleich dieser Wörter zu mehreren Zeitpunkten des Simulationsdurchlaufs lässt auf Rückschlüsse über die Geschwindigkeit der Wortveränderung schließen. Diese Daten sollen aus Gründen der Übersichtlichkeit nicht nur in simulationsschrittbasierten Wörterbüchern aufbereitet werden, sondern auch die jeweiligen Veränderungen von Messpunkt zu Messpunkt und die aufsummierten Veränderungen aller Wörter des Lexikons in einer Kreuztabelle mit den jeweiligen Simulationsschritten auf ihren Achsen.

3.5.7 Ereignisse

Die letzte Neuerung des Modells ist die Einführung von Ereignissen, die Simulationsparameter zur Laufzeit der Simulation ändern können. Sie bestehen aus einem Auslöser, der ein gewisses Attribut auf Echtheit überprüft und eine Anweisung, die bei Eintritt des Ereignisses ausgeführt wird. Als mögliche Auslöser sind das Erreichen eines bestimmten Simulationsschritts oder aber das Überschreiten vorher definierter Schwellenwerte der Indikatoren Avg1, Avg2 und des gegenseitigen Verständnisses vorgesehen. Die Anweisung soll variabel gehalten werden und alle vorher definierten Attribute einer Simulation manipulieren können.

Auf diese Weise soll beispielsweise ermöglicht werden, eine Agentengruppe einen gemeinsamen Wortschatz ausbilden zu lassen und bei einem ausreichend großen Kommunikationserfolg in mehrere Untergruppen aufzuteilen, um eine Dialektbildung zu forcieren. Wichtig ist hierbei, dass die Entscheidung über die Aufteilung der Agentengemeinschaft zur Laufzeit geschieht. Eine Einteilung der Agenten in Gruppen von Beginn des Simulationsdurchlaufs an würde die Ausbildung gruppeninterner Lexika bewirken, jedoch nicht die eines gemeinsamen Lexikons aller Agenten.

4. Simulationsdurchführungen und Auswertungen

In diesem Kapitel werden mit Hilfe des geschaffenen Werkzeugs DiaLex erstellte Simulationen thematisch hergeleitet, um darauf die in der Simulation gewonnenen Daten auszuwerten und zu interpretieren. In jedem der Unterkapitel werden die Attribute einer Simulation durch eine Instanziierung der folgenden Variablen und Funktionen festgelegt:

Visuelle Szenen:

- m = Anzahl der visuellen Szenen
- S = Menge der visuellen Szenen $\{ s_1, s_2, \ldots, s_m \}$
- F = Menge der Häufigkeit der visuellen Szenen

Simulationsumgebung:

- n = Anzahl der Agenten innerhalb einer Simulationsumgebung
- D = Ausdehnung der Simulationsumgebung $\{ d_l, d_w \}$
- f_{pos} = Funktion zur Positionierung der Agenten
- C = Koordinaten der Agenten

Agent:

- f_{arch} = Funktion der Netzstruktur eines Agenten
- W = Menge der initialen Verbindungsgewichtungen des neuronalen Netzes eines jeden Agenten
- w = Wortposition im neuronalen Netz
- μ = Lernrate
- ψ = Momentum

Kommunikation:

- f_{scene} = Funktion zur Auswahl einer visuellen Szene
- f_{sp} = Funktion zur Auswahl eines Sprechers
- f_{li} = Funktion zur Auswahl eines Zuhörers
- f_{com} = Funktion zum Ablauf der Kommunikation
- ς = Größe der Nachbarschaft

Störfaktoren:

- f_{death} = Funktion der Sterbewahrscheinlichkeit eines Agenten
- f_{dec} = Funktion der Abschwächung der Verbindungsgewichtungen
- f_{nip} = Funktion des Rauschens innerhalb der visuellen Szenen
- f_{ncom} = Funktion des Rauschens innerhalb der Kommunikation
- f_{dct} = Funktion der diskreten Kosinustransformation der Kommunikation

Ereignisse:

- E = Menge der Ereignisse

4.1 Simulation 1: kein Störfaktor

In diesem Unterkapitel werden aus Gründen der Vergleichbarkeit die Ergebnisse eines Simulationsdurchlaufs ohne Störfaktoren dargestellt. Die Gemeinschaft der Agenten soll innerhalb der kompletten Gemeinschaft ein Lexikon erlernen, bis ein Wert des gegenseitigen Verständnisses von 0.75 erreicht ist. Danach teilt sich die Agentengemeinschaft in vier Untergruppen auf.

Als Eingabepattern werden die vier Einheitsvektoren des vierdimensionalen Raumes gewählt, die zusammen die Orthonormalbasis bilden. (siehe Abb. 48) Da sie jeweils senkrecht zueinander stehen, unterscheiden sie sich jeweils zueinander um den gleichen Betrag. Dadurch ist gewährleistet, dass sie sich die verbalen Repräsentationen voneinander unabhängig entwickeln.

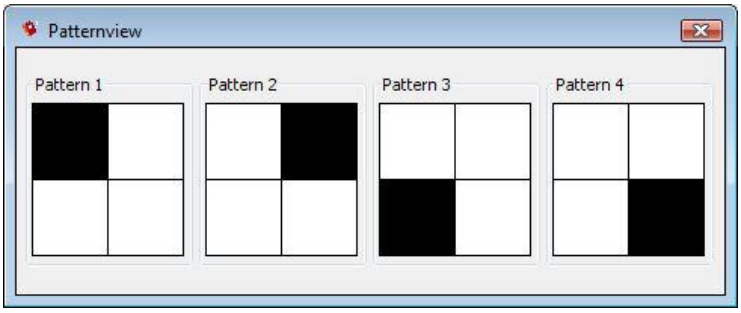


Abb. 48 Die Einheitsvektoren als Eingabemuster

Die Agenten besitzen eine verborgene Schicht, die die Wortschicht darstellt. Im Gegensatz zu den in [FK07] erläuterten Simulationsdurchläufen ist die verborgene Schicht hier und in den nachfolgenden Simulationen mit Störfaktoren bewusst größer gewählt, da eine Sprachspaltung sonst auf einem genauen Vertauschen zweier Symbol-Wort-Kombinationen basieren müsste. Neben der Tatsache, dass dieser Fall extrem unwahrscheinlich erscheint, entspricht er auch nicht der Realität. Im menschlichen Phonemsystem ist eine viel größere Anzahl von Phonemkombinationen möglich als tatsächlich benötigt werden. Viele mögliche Wortkreationen werden nicht genutzt und liegen daher brach.

In diesem und den nachfolgenden Simulationsdurchläufen wird eine Gruppe von 20 Agenten, die sich bereits auf ein gemeinsam genutztes Lexikon geeinigt haben, in vier voneinander getrennte Untergruppen aufgeteilt, zwischen denen ab diesem Zeitpunkt keine Kommunikation mehr möglich ist. Über die Zeit soll sich die Sprache der vier Untergruppen in verschiedene Richtungen entwickeln, sodass sie, mit zunehmender Zeit, zueinander immer unterschiedlicher werden.

4.1.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2
  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen
- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge S
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Es existiert keine Sterbewahrscheinlichkeit für die Agenten
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.1.2 Analyse

Die Agentenpopulation verhält sich vor der Trennung in Untergruppen exakt so, wie es bereits von Hutchins und Hazlehurst in [HH95] und Fuchs und Klein in [FK07] gezeigt worden ist. Die Werte des Avg1-Fehlermaßes steigen stetig an, während die Werte des Avg2-Indikators gegen 0.0 tendieren. Diese beiden Zeichen waren in den vorangegangenen Arbeiten stets das Zeichen für einen perfekten Simulationsdurchlauf mit einer eindeutigen Ausbildung eines gemeinsam genutzten Lexikons. Als Konsequenz dieser beiden Werte übersteigt somit auch der Indikator des gegenseitigen Verständnisses bereits in Simulationsschritt 15000 den Wert 0.75, der eine Aufteilung der Agentenpopulation in vier getrennte Untergruppen bewirkt.

Der Indikator Avg1 verhält sich danach genau so, wie es ohne eine Aufteilung in Untergruppen zu erwarten gewesen wäre. Er steigt langsam weiter an, bis er einen Wert von ca. 0.6 erreicht. Ab diesem Punkt gehen weitere Verbesserungen der Differenzausprägung zwischen den verbalen Repräsentationen extrem langsam vonstatten. Dieser Wert muss nicht zusätzlich in den einzelnen Untergruppen berechnet werden, da er agentenbezogen ist.

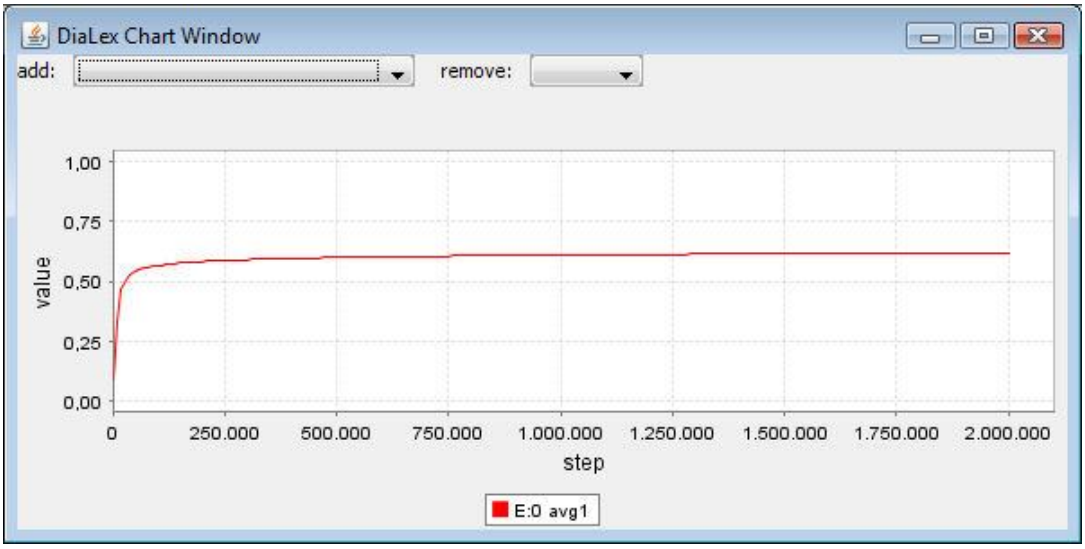


Abb. 49 Graph der Avg1-Werte

Die Abbildung des Verlaufs der Avg2-Werte zeichnet ein anderes Bild. (vgl. Abb. 50) Die gruppeninterne Differenz der verbalen Repräsentationen konvergiert gegen 0, genau so, wie es in den vorherigen Arbeiten gezeigt wurde. Die Wörter der Agenten verschiedener Gruppen unterscheiden sich jedoch leicht, der Avg2-Wert der gesamten Population stabilisiert sich rasch bei einem Wert von ca. 0,22.

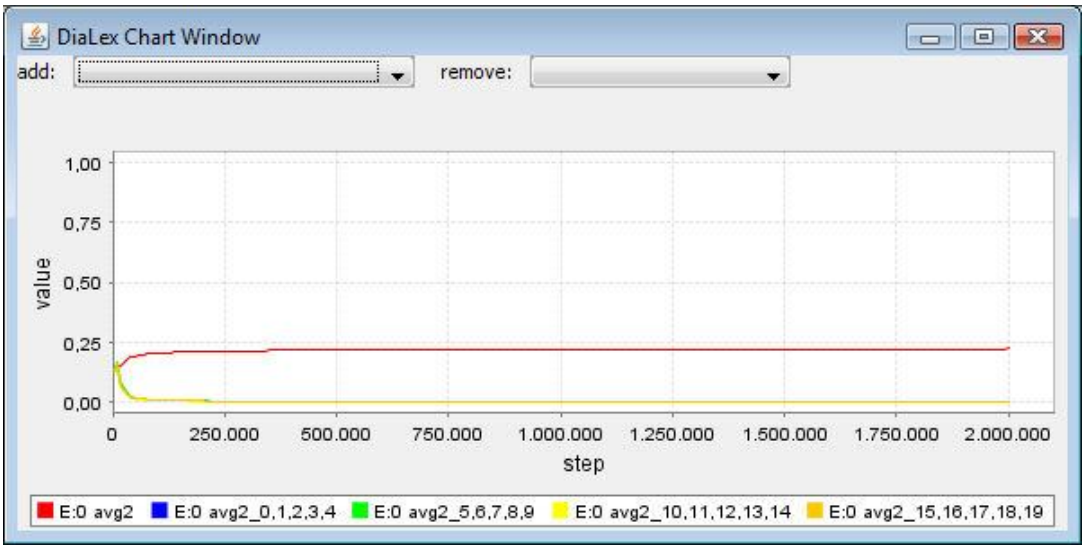


Abb. 50 Graph der Avg2-Werte

Trotz der um ein Fünftel bis ein Viertel von den gruppenexternen Repräsentationen verschiedener Worte erreicht die gegenseitige Verständlichkeit in Abb. 51) einen sehr hohen Wert von 0,85. Damit sind die meisten der

gruppenexternen Kommunikationsversuche erfolgreich. Dennoch liegt dieser Wert unter dem gruppenintern erreichten Idealwert von exakt 1.0.

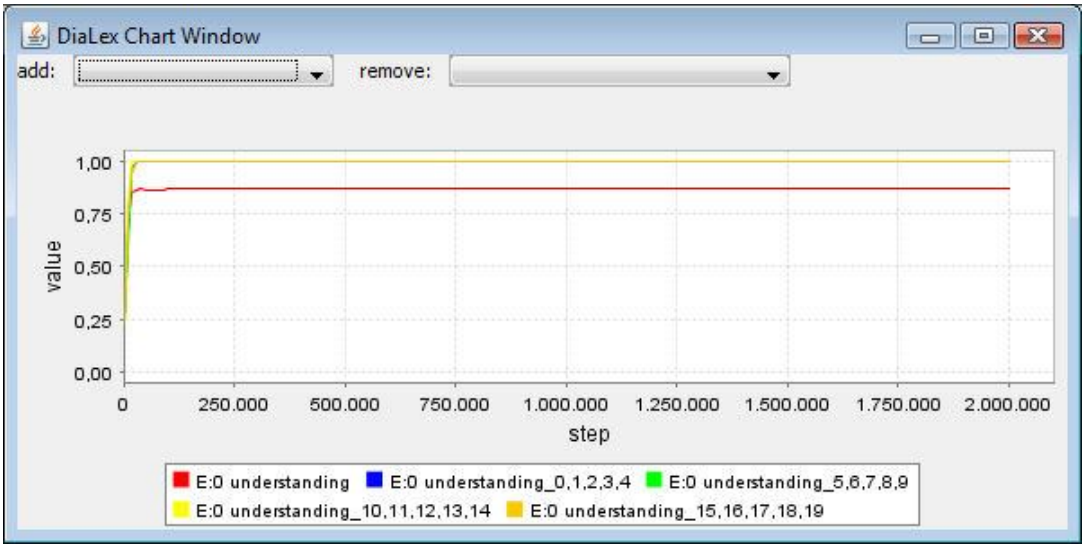


Abb. 51 Graph der Understanding-Werte

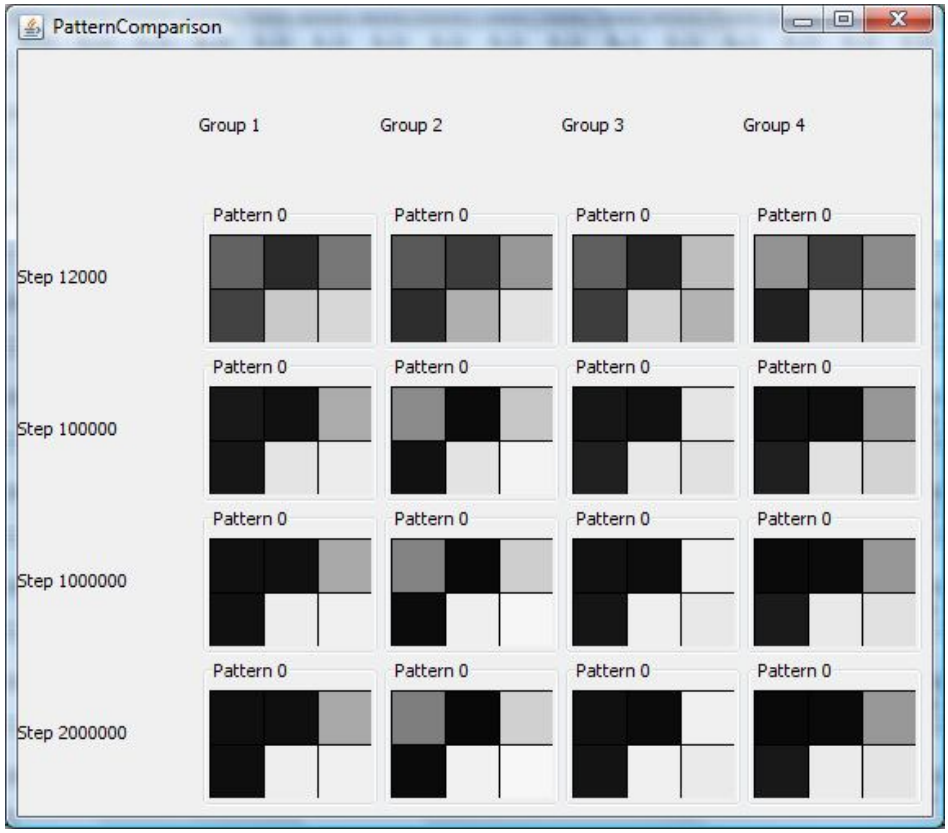
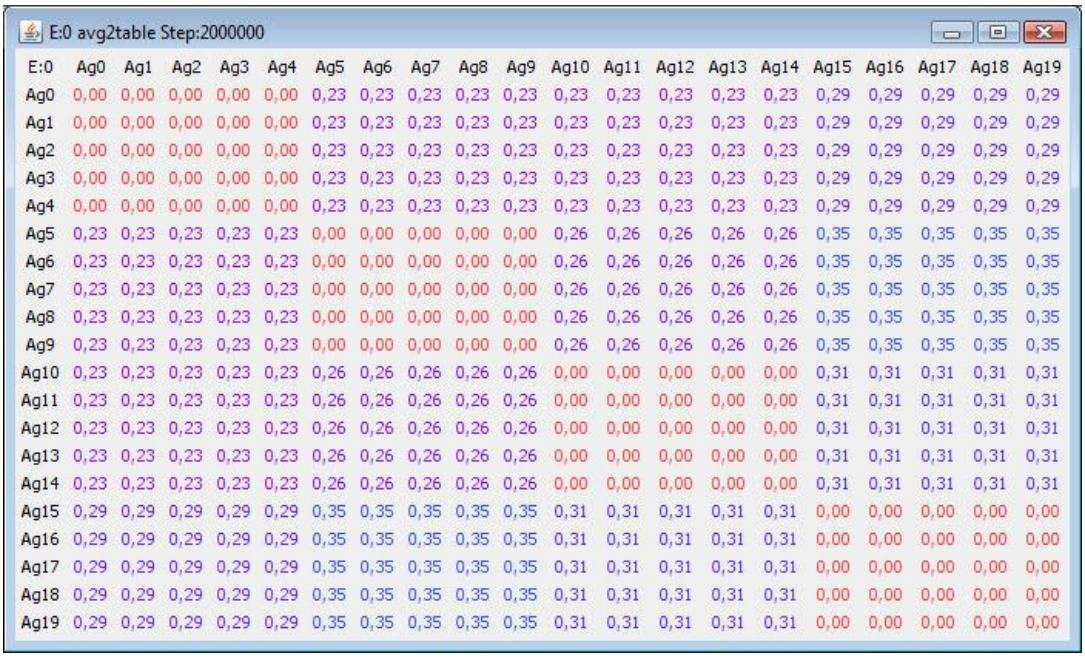


Abb. 52 Vergleich der Wortmuster

Eine starke Stabilität innerhalb des Simulationsdurchlaufs ist ebenfalls ersichtlich. Sämtliche Graphen verharren nach einer stetigen Zunahme bei einem Funktionswert. Daraus lassen sich stabile Lexika der einzelnen Agentengruppen ableiten. Dies zeigt sich ebenfalls in Abbildung 52, in der das Muster eines Wortes von vier Mitgliedern verschiedener Gruppen während verschiedener Simulationsschritte verglichen wird.

Die Avg2-Tabelle zum Ende der Simulation in Abb. 53 unterstreicht die nach Betrachtung der Graphen vermutete, klar ersichtliche Gruppeneinteilung. Die Worte aller Gruppenmitglieder unterscheiden sich nicht von den Worten der anderen Agenten der gleichen Gruppen. Dies zeigt sich klar in den roten, 5 mal 5 Felder großen, Rechtecken. Die gruppenexternen Wörter erreichen Unterschiede von bis zu 0.35, in diesem Fall die Wörter der Gruppen 2 (Agenten 5 bis 9) und 4 (Agenten 15-19).


E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,00	0,00	0,00	0,00	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,29	0,29	0,29	0,29	0,29
Ag1	0,00	0,00	0,00	0,00	0,00	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,29	0,29	0,29	0,29	0,29
Ag2	0,00	0,00	0,00	0,00	0,00	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,29	0,29	0,29	0,29	0,29
Ag3	0,00	0,00	0,00	0,00	0,00	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,29	0,29	0,29	0,29	0,29
Ag4	0,00	0,00	0,00	0,00	0,00	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,29	0,29	0,29	0,29	0,29
Ag5	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,35
Ag6	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,35
Ag7	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,35
Ag8	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,35
Ag9	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,35
Ag10	0,23	0,23	0,23	0,23	0,23	0,26	0,26	0,26	0,26	0,26	0,00	0,00	0,00	0,00	0,00	0,31	0,31	0,31	0,31	0,31
Ag11	0,23	0,23	0,23	0,23	0,23	0,26	0,26	0,26	0,26	0,26	0,00	0,00	0,00	0,00	0,00	0,31	0,31	0,31	0,31	0,31
Ag12	0,23	0,23	0,23	0,23	0,23	0,26	0,26	0,26	0,26	0,26	0,00	0,00	0,00	0,00	0,00	0,31	0,31	0,31	0,31	0,31
Ag13	0,23	0,23	0,23	0,23	0,23	0,26	0,26	0,26	0,26	0,26	0,00	0,00	0,00	0,00	0,00	0,31	0,31	0,31	0,31	0,31
Ag14	0,23	0,23	0,23	0,23	0,23	0,26	0,26	0,26	0,26	0,26	0,00	0,00	0,00	0,00	0,00	0,31	0,31	0,31	0,31	0,31
Ag15	0,29	0,29	0,29	0,29	0,29	0,35	0,35	0,35	0,35	0,35	0,31	0,31	0,31	0,31	0,31	0,00	0,00	0,00	0,00	0,00
Ag16	0,29	0,29	0,29	0,29	0,29	0,35	0,35	0,35	0,35	0,35	0,31	0,31	0,31	0,31	0,31	0,00	0,00	0,00	0,00	0,00
Ag17	0,29	0,29	0,29	0,29	0,29	0,35	0,35	0,35	0,35	0,35	0,31	0,31	0,31	0,31	0,31	0,00	0,00	0,00	0,00	0,00
Ag18	0,29	0,29	0,29	0,29	0,29	0,35	0,35	0,35	0,35	0,35	0,31	0,31	0,31	0,31	0,31	0,00	0,00	0,00	0,00	0,00
Ag19	0,29	0,29	0,29	0,29	0,29	0,35	0,35	0,35	0,35	0,35	0,31	0,31	0,31	0,31	0,31	0,00	0,00	0,00	0,00	0,00

Abb. 53 Avg2-Tabelle in Simulationsschritt 2000000

Die Tabelle des gegenseitigen Verständnisses in Abb. 54 zeigt ein weniger klares Bild der Gruppeneinteilung. Dies liegt vor allem daran, dass allgemein ein sehr hoher Grad der gegenseitigen Verständlichkeit in der Agentenpopulation vorherrscht.

Es ist jedoch erkennbar, dass die Kommunikation zwischen den Agenten 15 bis 19, die in der vierten Agentengruppe zusammengefasst wurden, und den Agenten 0 bis 9 der Gruppen 1 und 2 wenig erfolgreich abläuft. Offensichtlich unterscheiden sich die verbalen Repräsentationen der Gruppe 4 der Agentenpopulation daher stärker von denen der anderen Agenten.

Daraus lässt sich, aufgrund des Fehlens eines Störfaktors in diesem Simulationsdurchlauf, folgern, dass die Wörter der Gruppe 4 vor der Aufteilung der Agentenpopulation in vier Untergruppen den Wörtern der anderen Agenten nicht ausreichend ähnlich waren um eine Verständlichkeit aufrecht zu erhalten. Dies ist auf die zufällige Auswahl der Agenten zurückzuführen.

E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,50	0,62	0,50	0,50	0,62
Ag1	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,75	0,62	0,62	0,75
Ag2	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,75	0,62	0,62	0,75
Ag3	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,75	0,62	0,62	0,75
Ag4	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,75	0,62	0,62	0,75
Ag5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,75	1,00	0,62	0,75	0,62	0,62	0,75
Ag6	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,75	0,62	0,88	0,50	0,62	0,50	0,50	0,62
Ag7	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,75	1,00	0,62	0,75	0,62	0,62	0,75
Ag8	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,62	0,50	0,75	0,62	0,75	0,62	0,62	0,75
Ag9	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,75	0,62	0,88	0,62	0,75	0,62	0,62	0,75
Ag10	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75	0,88	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,75	0,88	0,62
Ag11	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75	0,88	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,75	0,88	0,62
Ag12	1,00	1,00	1,00	1,00	1,00	0,88	0,75	0,88	0,62	0,75	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75
Ag13	1,00	1,00	1,00	1,00	1,00	0,75	0,62	0,75	0,50	0,62	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75
Ag14	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	0,75
Ag15	0,50	0,62	0,62	0,62	0,62	0,62	0,50	0,62	0,62	0,62	0,88	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag16	0,62	0,75	0,75	0,75	0,75	0,75	0,62	0,75	0,75	0,75	0,88	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag17	0,50	0,62	0,62	0,62	0,62	0,62	0,50	0,62	0,62	0,62	0,75	0,75	0,88	0,88	0,88	1,00	1,00	1,00	1,00	1,00
Ag18	0,50	0,62	0,62	0,62	0,62	0,62	0,50	0,62	0,62	0,62	0,88	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag19	0,62	0,75	0,75	0,75	0,75	0,75	0,62	0,75	0,75	0,75	0,62	0,62	0,75	0,75	0,75	1,00	1,00	1,00	1,00	1,00

Abb. 54 Understanding-Tabelle in Simulationsschritt 2000000

4.2 Simulation 2: Rauschen während der Perzeption

In diesem Unterkapitel werden die Auswirkungen des Störfaktors Rauschen während der Perzeption der visuellen Eingabemuster auf die Simulationsergebnisse untersucht. Zusammen mit dem nachfolgenden Unterkapitel wird somit ein Vergleich der Auswirkungen der identischen Rauschfunktion an verschiedenen Stellen des neuronalen Netzes ermöglicht.

4.2.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2

  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen

- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5

- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge *S*
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Es existiert keine Sterbewahrscheinlichkeit für die Agenten
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert ein Rauschen innerhalb der visuellen Szenen der Stärken 0.001, 0.2 und 0.45.
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.2.2 Analyse

Bevor eine Sprachspaltung nachgewiesen werden kann, muss gezeigt werden, dass die Gemeinschaft der Agenten zu einem bestimmten Zeitpunkt eine gemeinsame Ursprache besessen hat, mit der sie sich untereinander nahezu fehlerfrei verständigen konnten. Als Maß hierfür bietet sich in erster Linie der Quotient des gegenseitigen Verständnisses an. Dieser Wert sollte sich in einem Bereich zwischen 0.75 und 1.0 befinden. Aufgrund des in dieser Simulation vorhandenen Störfaktors ist ein permanenter Wert von 1.0, der eine absolut fehlerfreie Verständigung signalisiert, nicht zu erwarten.

Ein weiterer, wenn auch weniger wichtiger Indikator, ist das Fehlermaß Avg2, das die Unterschiede der Wortschichten aller Agenten über alle visuellen Szenen berechnet. Dieser Wert sollte möglichst gegen 0.0 tendieren, wobei auch hier ein solch niedriger Wert aus vorher genannten Gründen nicht zu erwarten ist. Zur Sicherheit sollte auch der Avg1-Wert kontrolliert werden, da ein extrem niedriger Wert hier auf eine Unfähigkeit der neuronalen Netze, eine ausreichend große Anzahl verschiedener verbaler Repräsentationen bilden zu können, schließen ließe. Dies ist jedoch nicht zu erwarten, da die verborgene Schicht der Netze absichtlich zu groß gewählt wurde.

Die Simulationsergebnisse mit einem Rauschen in der Perzeption der Eingabemuster zeigen eindeutig, dass zwischen drei abgrenzbaren Intervallen im möglichen Wertebereich von 0.0 bis 1.0 unterschieden werden kann.

Rauschfaktor 0.001

Bei einem sehr kleinen Rauschfaktor im Bereich von 0.001 sind in der hier üblichen Simulationsdauer von 2000000 Schritten in den Indikatoren Avg2 und Understanding absolut keine Auswirkungen auf den Simulationsablauf erkennbar. Die Simulationsergebnisse zeigen dort das gleiche Muster wie der bereits bekannte Simulationsdurchlauf ohne jeglichen Störfaktor. (vgl. Unterkapitel 4.1) Dies wird in den nachfolgenden Abbildungen der Avg2-Werte (Abb. 55) sowie der Werte des gegenseitigen Verständnisses (Abb. 56) gezeigt.

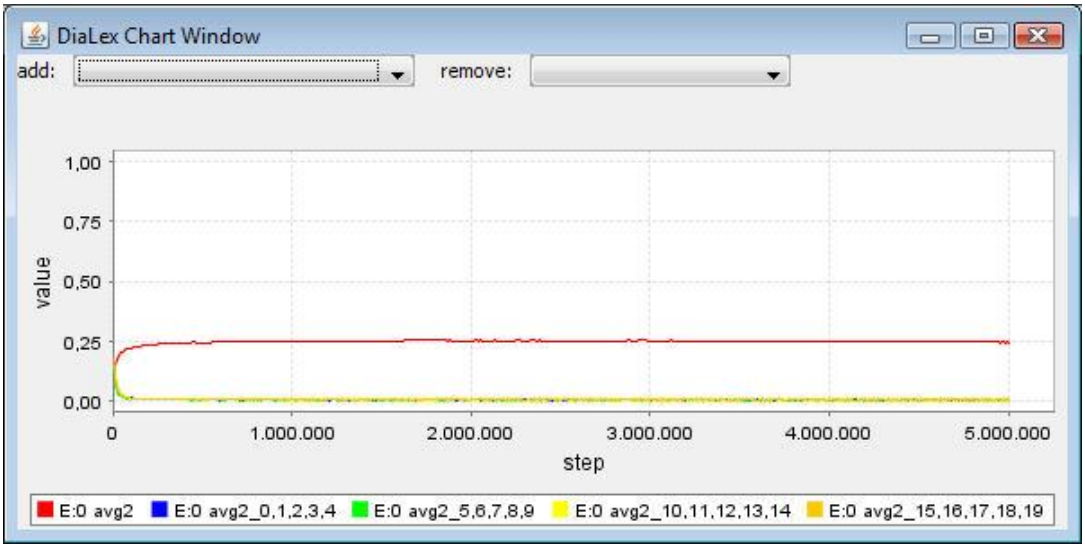


Abb. 55 Graph der Avg2-Werte

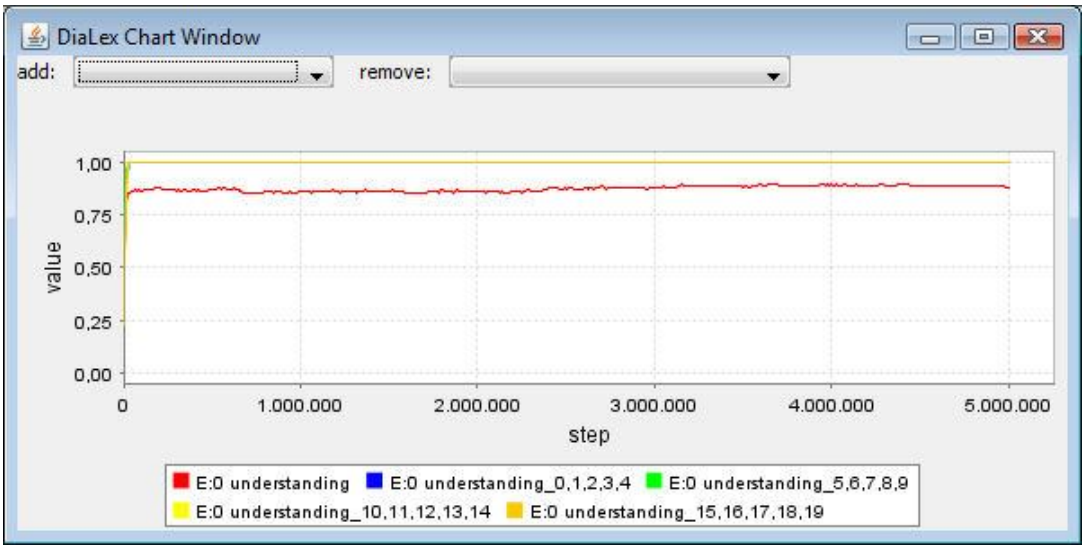


Abb. 56 Graph der Understanding-Werte

Die Tabellen der Indikatoren Avg2 und Understanding zeigen ebenfalls dieses Bild. Zwar unterscheiden sich die Wörter der einzelnen Agentengruppen um einen Wert zwischen 0.26 und 0.39, es sind dennoch keine sprachlichen Gruppenausprägungen beim gegenseitigen Verständnis innerhalb der Agentenpopulation erkennbar. (vgl. Abbs. 57 und 58) Ein Blick auf den Indikator Avg1 jedoch zeigt, dass sich die Veränderungen innerhalb des Lexikons bei derart kleinen Störfaktoren sehr langsam abspielen. So sinkt der durchschnittliche Avg1 Wert langsam aber stetig bei einer verlängerten Simulationsdauer von 5000000 Schritten von 0.55 auf einen Wert von 0.48. (vgl. Abb. 59) Dies ist die einzige messbare Abweichung zu den Ergebnissen einer Simulation ohne Störfaktoren.

Dieser Effekt ist aber offensichtlich immer noch zu klein, um eine Dialektbildung durch Rauschen innerhalb dieser Anzahl der Simulationsschritte zu ermöglichen.

E:0 avg2table Step:5000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,00	0,00	0,00	0,00	0,28	0,28	0,28	0,28	0,28	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,34
Ag1	0,00	0,00	0,00	0,00	0,00	0,27	0,27	0,27	0,27	0,27	0,26	0,26	0,26	0,26	0,26	0,34	0,34	0,34	0,34	0,34
Ag2	0,00	0,00	0,00	0,00	0,00	0,27	0,28	0,27	0,27	0,28	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,35	0,35	0,34
Ag3	0,00	0,00	0,00	0,00	0,00	0,28	0,28	0,28	0,28	0,28	0,26	0,26	0,26	0,26	0,26	0,35	0,35	0,34	0,35	0,34
Ag4	0,00	0,00	0,00	0,00	0,00	0,28	0,28	0,28	0,28	0,28	0,26	0,26	0,26	0,26	0,26	0,34	0,34	0,34	0,35	0,34
Ag5	0,28	0,27	0,27	0,28	0,28	0,00	0,00	0,00	0,00	0,00	0,27	0,28	0,27	0,28	0,28	0,29	0,29	0,29	0,29	0,29
Ag6	0,28	0,27	0,28	0,28	0,28	0,00	0,00	0,00	0,00	0,00	0,27	0,28	0,27	0,28	0,28	0,29	0,29	0,29	0,29	0,29
Ag7	0,28	0,27	0,27	0,28	0,28	0,00	0,00	0,00	0,00	0,00	0,27	0,28	0,27	0,28	0,28	0,29	0,29	0,29	0,29	0,29
Ag8	0,28	0,27	0,27	0,28	0,28	0,00	0,00	0,00	0,00	0,00	0,27	0,27	0,27	0,28	0,27	0,29	0,29	0,29	0,29	0,29
Ag9	0,28	0,27	0,28	0,28	0,28	0,00	0,00	0,00	0,00	0,00	0,27	0,27	0,27	0,28	0,28	0,29	0,29	0,29	0,29	0,29
Ag10	0,26	0,26	0,26	0,26	0,26	0,27	0,27	0,27	0,27	0,27	0,00	0,00	0,01	0,00	0,00	0,39	0,39	0,39	0,39	0,39
Ag11	0,26	0,26	0,26	0,26	0,26	0,28	0,28	0,28	0,27	0,27	0,00	0,00	0,01	0,00	0,00	0,39	0,39	0,39	0,39	0,39
Ag12	0,26	0,26	0,26	0,26	0,26	0,27	0,27	0,27	0,27	0,27	0,01	0,01	0,00	0,01	0,01	0,39	0,39	0,39	0,39	0,39
Ag13	0,26	0,26	0,26	0,26	0,26	0,28	0,28	0,28	0,28	0,28	0,00	0,00	0,01	0,00	0,00	0,39	0,39	0,39	0,39	0,39
Ag14	0,26	0,26	0,26	0,26	0,26	0,28	0,28	0,28	0,27	0,28	0,00	0,00	0,01	0,00	0,00	0,39	0,39	0,39	0,39	0,39
Ag15	0,35	0,34	0,35	0,35	0,34	0,29	0,29	0,29	0,29	0,29	0,39	0,39	0,39	0,39	0,39	0,00	0,00	0,00	0,00	0,00
Ag16	0,35	0,34	0,35	0,35	0,34	0,29	0,29	0,29	0,29	0,29	0,39	0,39	0,39	0,39	0,39	0,00	0,00	0,00	0,00	0,00
Ag17	0,35	0,34	0,35	0,34	0,34	0,29	0,29	0,29	0,29	0,29	0,39	0,39	0,39	0,39	0,39	0,00	0,00	0,00	0,00	0,00
Ag18	0,35	0,34	0,35	0,35	0,35	0,29	0,29	0,29	0,29	0,29	0,39	0,39	0,39	0,39	0,39	0,00	0,00	0,00	0,00	0,00
Ag19	0,34	0,34	0,34	0,34	0,34	0,29	0,29	0,29	0,29	0,29	0,39	0,39	0,39	0,39	0,39	0,00	0,00	0,00	0,00	0,00

Abb. 57 Avg2-Tabelle in Simulationsschritt 5000000

E:0 understandingtable Step:5000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,88	0,88	0,75
Ag1	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,75	0,75	0,62
Ag2	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,62	0,62	0,62	0,50
Ag3	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,75	0,75	0,62
Ag4	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,75	0,75	0,75	0,75	0,62
Ag5	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	0,88	0,88	0,88	0,88	0,75	0,88
Ag6	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	0,88	0,75	0,75	0,75	0,62	0,75
Ag7	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	0,88	0,88	0,88	0,88	0,75	0,88
Ag8	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	0,88	0,88	0,88	0,88	0,75	0,88
Ag9	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	0,88	0,75	0,75	0,75	0,62	0,75
Ag10	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,62	0,62	0,50	0,50	0,50
Ag11	1,00	1,00	1,00	1,00	0,88	0,88	0,88	0,88	0,88	0,88	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,62	0,62	0,62
Ag12	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,62	0,62	0,62
Ag13	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,75	0,62	0,62	0,62
Ag14	1,00	1,00	1,00	1,00	0,88	0,88	0,88	0,88	0,88	0,88	1,00	1,00	1,00	1,00	1,00	0,62	0,62	0,50	0,50	0,50
Ag15	0,88	0,75	0,62	0,75	0,75	0,88	0,75	0,88	0,88	0,75	0,62	0,75	0,75	0,75	0,62	1,00	1,00	1,00	1,00	1,00
Ag16	0,88	0,75	0,62	0,75	0,75	0,88	0,75	0,88	0,88	0,75	0,62	0,75	0,75	0,75	0,62	1,00	1,00	1,00	1,00	1,00
Ag17	0,88	0,75	0,62	0,75	0,75	0,88	0,75	0,88	0,88	0,75	0,50	0,62	0,62	0,62	0,50	1,00	1,00	1,00	1,00	1,00
Ag18	0,88	0,75	0,62	0,75	0,75	0,75	0,62	0,75	0,75	0,62	0,50	0,62	0,62	0,62	0,50	1,00	1,00	1,00	1,00	1,00
Ag19	0,75	0,62	0,50	0,62	0,62	0,88	0,75	0,88	0,88	0,75	0,50	0,62	0,62	0,62	0,50	1,00	1,00	1,00	1,00	1,00

Abb. 58 Understanding-Tabelle in Simulationsschritt 5000000

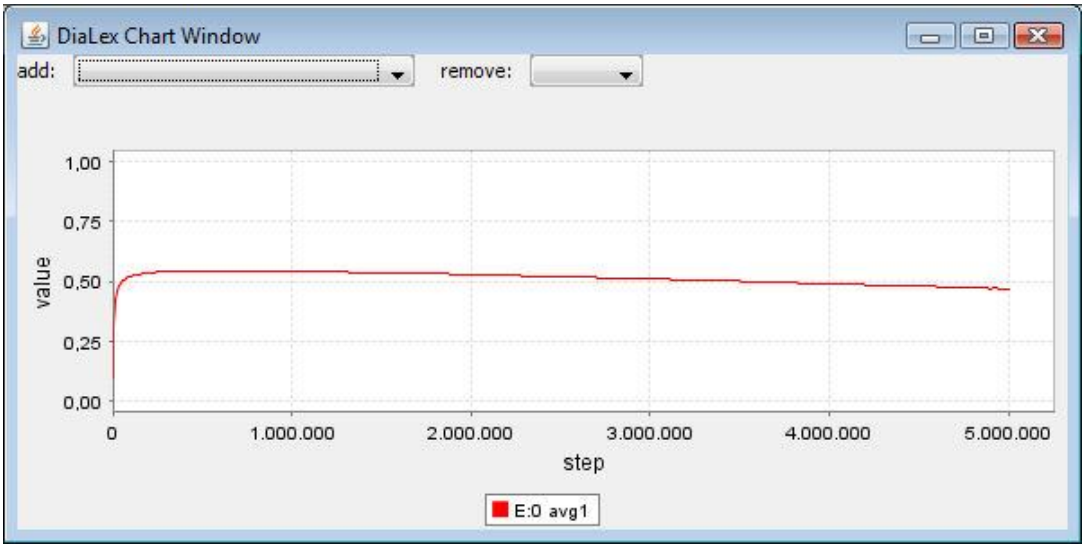


Abb. 59 Graph der Avg1-Werte

Rauschfaktor 0.2

Bei etwas größeren Werten dieses Störfaktors ändert sich das Verhalten der Agentenpopulation im Bezug auf das ausgebildete Lexikon. Die Werte des Indikators Avg1 steigen nur auf einen Wert von ungefähr 0.35 und sinken nach der Aufteilung der Agentenpopulation stark auf einen Wert von ungefähr 0.2-0.15 in dem sie sich dann relativ stabil verhalten, wenn auch mit leicht fallender Tendenz (vgl. Abb. 60), was eindeutig auf eine, durch das Rauschen bedingte, größere Gleichheit der verbalen Expressionen untereinander hindeutet.

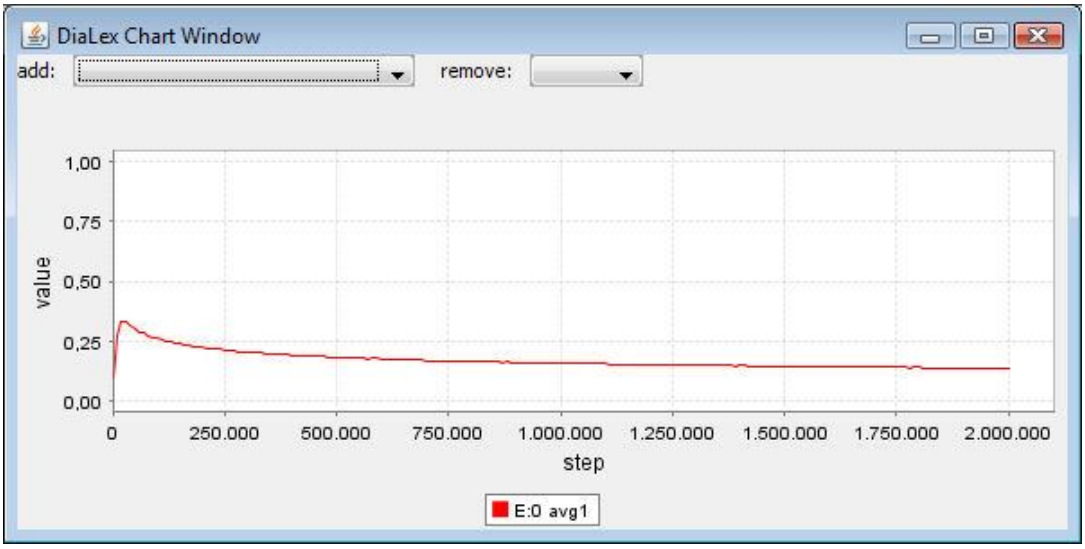


Abb. 60 Graph der Avg1-Werte

Avg2 und Understanding malen das erwünschte Bild einer nahezu perfekten Dialektbildung bzw. Sprachspaltung. (vgl. Abbs. 61 und 62) Mit dem Vollzug der Aufspaltung der Agentenpopulation in ihre räumlich definierten Untergruppen steigt der gemeinschaftliche Avg2-Wert stetig an, während sich die Avg2-Werte der Agentenuntergruppen auf gleichem Niveau halten. Die Werte des gegenseitigen Verständnisses demonstrieren dies in entgegengesetzter Weise. Die Fähigkeit der internen Gruppenkommunikation der Agenten wird nicht beeinträchtigt, während sich die Fähigkeit des Verständnisses gruppenfremder Agenten im Laufe der Zeit zurückbildet.

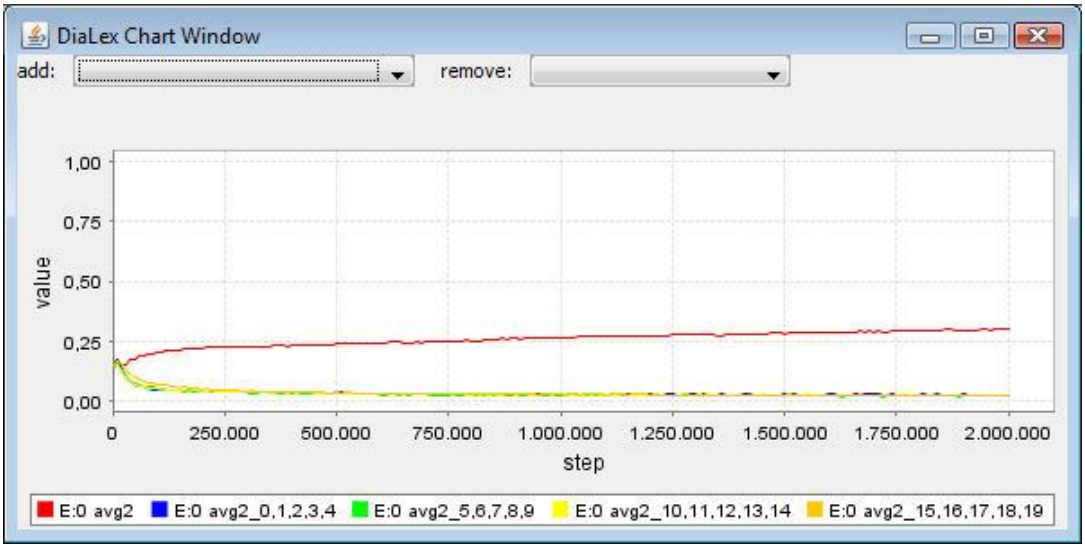


Abb. 61 Graph der Avg2-Werte

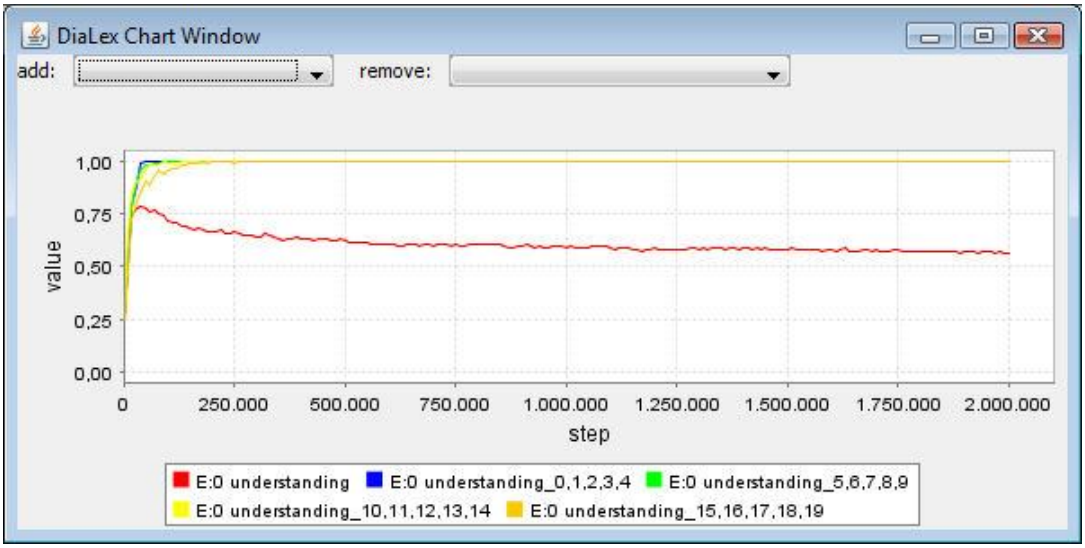


Abb. 62 Graph der Understanding-Werte

Der Vergleich der Wortmuster von Agenten verschiedener Gruppenzugehörigkeit über die Zeit zeigt ebenfalls eine stetige Entfernung der einzelnen Wörter.

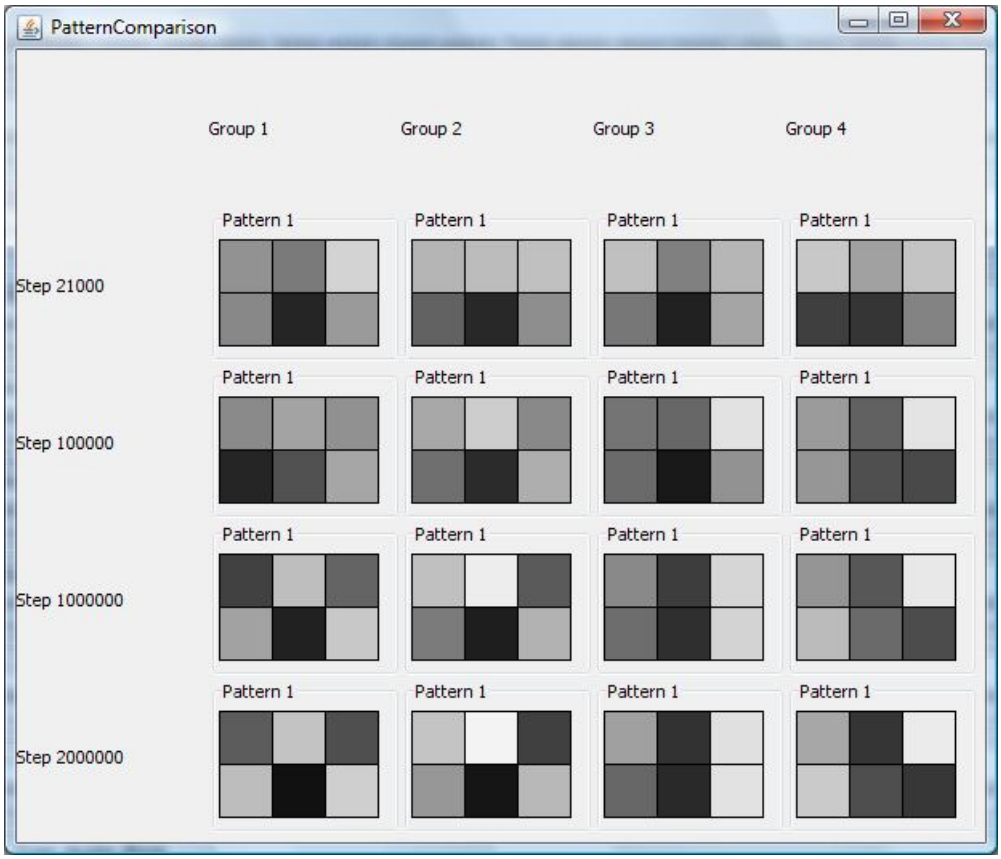


Abb. 63 Vergleich der Wortmuster

Die Tabellen zu Avg2 und Understanding zeigen nach 2000000 Schritten die erwarteten Gruppenbildungen (vgl. Abb. 64 und Abb. 65), die zum Zeitpunkt der Aufteilung in Untergruppen innerhalb des gleichen Simulationsdurchlaufs noch nicht ausgeprägt waren. (vgl. Abb. 66 und Abb. 67)

E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,02	0,03	0,03	0,02	0,19	0,20	0,19	0,20	0,20	0,41	0,39	0,39	0,39	0,39	0,44	0,45	0,45	0,44	0,44
Ag1	0,02	0,00	0,02	0,03	0,02	0,20	0,20	0,20	0,20	0,21	0,41	0,40	0,40	0,39	0,40	0,45	0,45	0,45	0,45	0,45
Ag2	0,03	0,02	0,00	0,03	0,02	0,19	0,20	0,20	0,20	0,20	0,41	0,40	0,40	0,39	0,40	0,46	0,46	0,46	0,46	0,46
Ag3	0,03	0,03	0,03	0,00	0,02	0,19	0,19	0,19	0,19	0,19	0,40	0,39	0,39	0,38	0,39	0,44	0,44	0,44	0,44	0,44
Ag4	0,02	0,02	0,02	0,02	0,00	0,19	0,19	0,19	0,19	0,19	0,40	0,39	0,39	0,38	0,39	0,44	0,45	0,45	0,44	0,44
Ag5	0,19	0,20	0,19	0,19	0,19	0,00	0,03	0,02	0,02	0,02	0,42	0,41	0,41	0,41	0,41	0,49	0,48	0,48	0,49	0,48
Ag6	0,20	0,20	0,20	0,19	0,19	0,03	0,00	0,02	0,02	0,02	0,42	0,41	0,41	0,41	0,41	0,49	0,48	0,48	0,48	0,48
Ag7	0,19	0,20	0,20	0,19	0,19	0,02	0,02	0,00	0,01	0,01	0,42	0,41	0,41	0,41	0,42	0,48	0,48	0,48	0,48	0,48
Ag8	0,20	0,20	0,20	0,19	0,19	0,02	0,02	0,01	0,00	0,01	0,42	0,41	0,41	0,41	0,41	0,48	0,48	0,48	0,48	0,48
Ag9	0,20	0,21	0,20	0,19	0,19	0,02	0,02	0,01	0,01	0,00	0,42	0,41	0,41	0,41	0,42	0,49	0,48	0,49	0,49	0,48
Ag10	0,41	0,41	0,41	0,40	0,40	0,42	0,42	0,42	0,42	0,42	0,00	0,02	0,03	0,04	0,04	0,32	0,32	0,33	0,32	0,32
Ag11	0,39	0,40	0,40	0,39	0,39	0,41	0,41	0,41	0,41	0,41	0,02	0,00	0,01	0,02	0,03	0,31	0,31	0,31	0,31	0,31
Ag12	0,39	0,40	0,40	0,39	0,39	0,41	0,41	0,41	0,41	0,41	0,03	0,01	0,00	0,02	0,02	0,30	0,31	0,31	0,31	0,30
Ag13	0,39	0,39	0,39	0,38	0,38	0,41	0,41	0,41	0,41	0,41	0,04	0,02	0,02	0,00	0,02	0,30	0,31	0,31	0,31	0,30
Ag14	0,39	0,40	0,40	0,39	0,39	0,41	0,41	0,42	0,41	0,42	0,04	0,03	0,02	0,02	0,00	0,29	0,30	0,30	0,30	0,30
Ag15	0,44	0,45	0,46	0,44	0,44	0,49	0,49	0,48	0,48	0,49	0,32	0,31	0,30	0,30	0,29	0,00	0,02	0,03	0,02	0,02
Ag16	0,45	0,45	0,46	0,44	0,45	0,48	0,48	0,48	0,48	0,48	0,32	0,31	0,31	0,31	0,30	0,02	0,00	0,02	0,02	0,02
Ag17	0,45	0,45	0,46	0,44	0,45	0,48	0,48	0,48	0,48	0,49	0,33	0,31	0,31	0,31	0,30	0,03	0,02	0,00	0,02	0,02
Ag18	0,44	0,45	0,46	0,44	0,44	0,49	0,48	0,48	0,48	0,49	0,32	0,31	0,31	0,31	0,30	0,02	0,02	0,02	0,00	0,02
Ag19	0,44	0,45	0,46	0,44	0,44	0,48	0,48	0,48	0,48	0,48	0,32	0,31	0,30	0,30	0,30	0,02	0,02	0,02	0,02	0,00

Abb. 64 Avg2-Tabelle in Simulationsschritt 2000000

E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38	0,38	0,38	0,50	0,38	0,38	0,38	0,38	0,50	0,50	0,62
Ag1	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38	0,38	0,38	0,50	0,38	0,38	0,38	0,38	0,50	0,50	0,62
Ag2	1,00	1,00	1,00	1,00	1,00	0,62	0,62	0,50	0,62	0,50	0,25	0,25	0,38	0,25	0,25	0,25	0,25	0,38	0,38	0,50
Ag3	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,38	0,38	0,38	0,38	0,38	0,38	0,25	0,50	0,50	0,50
Ag4	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,25	0,38	0,50	0,38	0,38	0,50	0,38	0,50	0,50	0,62
Ag5	0,50	0,50	0,62	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,25	0,25	0,50	0,38	0,62	0,50	0,50	0,62	0,88	0,50
Ag6	0,50	0,50	0,62	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38	0,25	0,38	0,25	0,62	0,25
Ag7	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,62	0,62	0,62	0,50	0,50	0,25	0,38	0,25	0,62	0,25
Ag8	0,50	0,50	0,62	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,50	0,62	0,62	0,62	0,62	0,25	0,38	0,25	0,62	0,25
Ag9	0,38	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,62	0,25	0,38	0,25	0,62	0,25
Ag10	0,38	0,38	0,25	0,38	0,25	0,25	0,50	0,62	0,50	0,50	1,00	1,00	1,00	1,00	1,00	0,50	0,38	0,50	0,38	0,38
Ag11	0,38	0,38	0,25	0,38	0,38	0,25	0,50	0,62	0,62	0,50	1,00	1,00	1,00	1,00	1,00	0,62	0,50	0,62	0,50	0,50
Ag12	0,50	0,50	0,38	0,38	0,50	0,50	0,50	0,62	0,62	0,50	1,00	1,00	1,00	1,00	1,00	0,38	0,25	0,38	0,25	0,25
Ag13	0,38	0,38	0,25	0,38	0,38	0,38	0,50	0,50	0,62	0,50	1,00	1,00	1,00	1,00	1,00	0,38	0,38	0,62	0,38	0,38
Ag14	0,38	0,38	0,25	0,38	0,38	0,62	0,38	0,50	0,62	0,62	1,00	1,00	1,00	1,00	1,00	0,62	0,50	0,62	0,50	0,50
Ag15	0,38	0,38	0,25	0,38	0,50	0,50	0,25	0,25	0,25	0,25	0,50	0,62	0,38	0,38	0,62	1,00	1,00	1,00	1,00	1,00
Ag16	0,38	0,38	0,25	0,25	0,38	0,50	0,38	0,38	0,38	0,38	0,38	0,50	0,25	0,38	0,50	1,00	1,00	1,00	1,00	1,00
Ag17	0,50	0,50	0,38	0,50	0,50	0,62	0,25	0,25	0,25	0,25	0,50	0,62	0,38	0,62	0,62	1,00	1,00	1,00	1,00	1,00
Ag18	0,50	0,50	0,38	0,50	0,50	0,88	0,62	0,62	0,62	0,62	0,38	0,50	0,25	0,38	0,50	1,00	1,00	1,00	1,00	1,00
Ag19	0,62	0,62	0,50	0,50	0,62	0,50	0,25	0,25	0,25	0,25	0,38	0,50	0,25	0,38	0,50	1,00	1,00	1,00	1,00	1,00

Abb. 65 Understanding-Tabelle in Simulationsschritt 2000000

E:0 avg2table Step:25000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,09	0,12	0,14	0,09	0,17	0,13	0,16	0,18	0,17	0,17	0,13	0,15	0,13	0,18	0,12	0,16	0,18	0,14	0,10
Ag1	0,09	0,00	0,11	0,10	0,06	0,16	0,10	0,11	0,14	0,15	0,17	0,14	0,15	0,13	0,17	0,10	0,18	0,21	0,16	0,11
Ag2	0,12	0,11	0,00	0,17	0,13	0,21	0,13	0,17	0,21	0,22	0,21	0,20	0,21	0,19	0,20	0,14	0,22	0,21	0,20	0,14
Ag3	0,14	0,10	0,17	0,00	0,11	0,18	0,12	0,11	0,15	0,15	0,16	0,16	0,15	0,14	0,15	0,13	0,20	0,21	0,17	0,14
Ag4	0,09	0,06	0,13	0,11	0,00	0,15	0,08	0,11	0,14	0,14	0,17	0,15	0,15	0,15	0,18	0,08	0,17	0,19	0,17	0,13
Ag5	0,17	0,16	0,21	0,18	0,15	0,00	0,16	0,14	0,08	0,07	0,19	0,18	0,16	0,18	0,15	0,18	0,17	0,15	0,16	0,18
Ag6	0,13	0,10	0,13	0,12	0,08	0,16	0,00	0,10	0,13	0,14	0,16	0,17	0,17	0,17	0,18	0,11	0,21	0,19	0,19	0,16
Ag7	0,16	0,11	0,17	0,11	0,11	0,14	0,10	0,00	0,12	0,11	0,19	0,19	0,17	0,17	0,17	0,14	0,22	0,19	0,20	0,18
Ag8	0,18	0,14	0,21	0,15	0,14	0,08	0,13	0,12	0,00	0,07	0,18	0,18	0,16	0,17	0,16	0,16	0,20	0,18	0,18	0,18
Ag9	0,17	0,15	0,22	0,15	0,14	0,07	0,14	0,11	0,07	0,00	0,16	0,16	0,14	0,16	0,14	0,16	0,17	0,16	0,16	0,17
Ag10	0,17	0,17	0,21	0,16	0,17	0,19	0,16	0,19	0,18	0,16	0,00	0,11	0,10	0,13	0,15	0,15	0,13	0,14	0,10	0,15
Ag11	0,13	0,14	0,20	0,16	0,15	0,18	0,17	0,19	0,18	0,16	0,11	0,00	0,05	0,05	0,14	0,15	0,13	0,16	0,09	0,12
Ag12	0,15	0,15	0,21	0,15	0,15	0,16	0,17	0,17	0,16	0,14	0,10	0,05	0,00	0,06	0,12	0,15	0,11	0,15	0,08	0,12
Ag13	0,13	0,13	0,19	0,14	0,15	0,18	0,17	0,17	0,17	0,16	0,13	0,05	0,06	0,00	0,14	0,15	0,15	0,18	0,11	0,11
Ag14	0,18	0,17	0,20	0,15	0,18	0,15	0,18	0,17	0,16	0,14	0,15	0,14	0,12	0,14	0,00	0,18	0,14	0,14	0,13	0,13
Ag15	0,12	0,10	0,14	0,13	0,08	0,18	0,11	0,14	0,16	0,16	0,15	0,15	0,15	0,15	0,18	0,00	0,16	0,17	0,16	0,13
Ag16	0,16	0,18	0,22	0,20	0,17	0,17	0,21	0,22	0,20	0,17	0,13	0,13	0,11	0,15	0,14	0,16	0,00	0,12	0,07	0,13
Ag17	0,18	0,21	0,21	0,21	0,19	0,15	0,19	0,19	0,18	0,16	0,14	0,16	0,15	0,18	0,14	0,17	0,12	0,00	0,13	0,17
Ag18	0,14	0,16	0,20	0,17	0,17	0,16	0,19	0,20	0,18	0,16	0,10	0,09	0,08	0,11	0,13	0,16	0,07	0,13	0,00	0,11
Ag19	0,10	0,11	0,14	0,14	0,13	0,18	0,16	0,18	0,18	0,17	0,15	0,12	0,12	0,11	0,13	0,13	0,13	0,17	0,11	0,00

Abb. 66 Avg2-Tabelle in Simulationsschritt 25000

E:0 understandingtable Step:25000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	0,75	1,00	0,50	1,00	0,75	0,50	0,75	0,75	1,00	1,00	1,00	0,75	0,88	1,00	0,75	1,00	1,00
Ag1	1,00	1,00	0,88	0,88	1,00	0,62	1,00	1,00	0,75	0,75	0,88	1,00	0,88	1,00	0,75	0,88	0,75	0,50	0,62	0,88
Ag2	1,00	0,88	1,00	0,38	0,88	0,50	0,88	0,62	0,50	0,50	0,12	0,38	0,25	0,25	0,50	0,75	0,12	0,12	0,38	0,88
Ag3	0,75	0,88	0,38	1,00	0,88	0,50	0,88	1,00	0,88	0,88	0,62	0,62	0,62	0,62	0,75	0,75	0,50	0,50	0,50	0,62
Ag4	1,00	1,00	0,88	0,88	1,00	0,75	1,00	1,00	0,88	1,00	0,75	1,00	0,88	0,88	0,75	1,00	0,88	0,62	1,00	0,75
Ag5	0,50	0,62	0,50	0,50	0,75	1,00	0,50	0,88	1,00	1,00	0,62	0,75	1,00	0,62	0,88	0,50	1,00	1,00	1,00	0,75
Ag6	1,00	1,00	0,88	0,88	1,00	0,50	1,00	1,00	0,62	0,75	1,00	0,62	0,88	0,50	0,62	1,00	0,25	0,50	0,50	0,75
Ag7	0,75	1,00	0,62	1,00	1,00	0,88	1,00	1,00	0,88	1,00	0,38	0,50	0,62	0,62	0,62	0,75	0,50	0,62	0,38	0,75
Ag8	0,50	0,75	0,50	0,88	0,88	1,00	0,62	0,88	1,00	1,00	0,62	0,75	0,75	0,75	0,75	0,75	0,75	0,62	0,75	0,75
Ag9	0,75	0,75	0,50	0,88	1,00	1,00	0,75	1,00	1,00	1,00	0,75	0,75	0,88	0,75	0,88	0,75	0,75	0,75	0,75	0,75
Ag10	0,75	0,88	0,12	0,62	0,75	0,62	1,00	0,38	0,62	0,75	1,00	1,00	1,00	1,00	0,88	0,50	1,00	0,75	1,00	0,75
Ag11	1,00	1,00	0,38	0,62	1,00	0,75	0,62	0,50	0,75	0,75	1,00	1,00	1,00	1,00	0,75	0,75	0,88	0,75	1,00	0,88
Ag12	1,00	0,88	0,25	0,62	0,88	1,00	0,88	0,62	0,75	0,88	1,00	1,00	1,00	1,00	0,75	0,75	1,00	0,88	1,00	0,75
Ag13	1,00	1,00	0,25	0,62	0,88	0,62	0,50	0,62	0,75	0,75	1,00	1,00	1,00	1,00	0,50	0,88	0,50	0,50	1,00	0,88
Ag14	0,75	0,75	0,50	0,75	0,75	0,88	0,62	0,62	0,75	0,88	0,88	0,75	0,75	0,50	1,00	0,62	0,62	1,00	0,88	0,75
Ag15	0,88	0,88	0,75	0,75	1,00	0,50	1,00	0,75	0,75	0,75	0,50	0,75	0,75	0,88	0,62	1,00	0,75	0,50	0,75	0,75
Ag16	1,00	0,75	0,12	0,50	0,88	1,00	0,25	0,50	0,75	0,75	1,00	0,88	1,00	0,50	0,62	0,75	1,00	1,00	1,00	0,88
Ag17	0,75	0,50	0,12	0,50	0,62	1,00	0,50	0,62	0,62	0,75	0,75	0,75	0,88	0,50	1,00	0,50	1,00	1,00	1,00	0,75
Ag18	1,00	0,62	0,38	0,50	1,00	1,00	0,50	0,38	0,75	0,75	1,00	1,00	1,00	1,00	0,88	0,75	1,00	1,00	1,00	0,88
Ag19	1,00	0,88	0,88	0,62	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,88	0,75	0,88	0,75	0,75	0,88	0,75	0,88	1,00

Abb. 67 Understanding-Tabelle in Simulationsschritt 25000

Rauschfaktor 0.45

Bei einem noch stärkeren Rauschen jedoch, ungefähr[16] bei Werten größer als 0.4, ändert sich das Simulationsverhalten ein weiteres Mal. Das Rauschen ist nunmehr zu stark, um überhaupt die Ausbildung eines gemeinsamen Lexikons zu ermöglichen. Dies zeigt sich beispielhaft in einem Simulationsdurchlauf mit einem Rauschen der Stärke 0.45 an den Graphen der Avg2- und Understanding-Indikatoren. (Abb. 68 und Abb. 69)

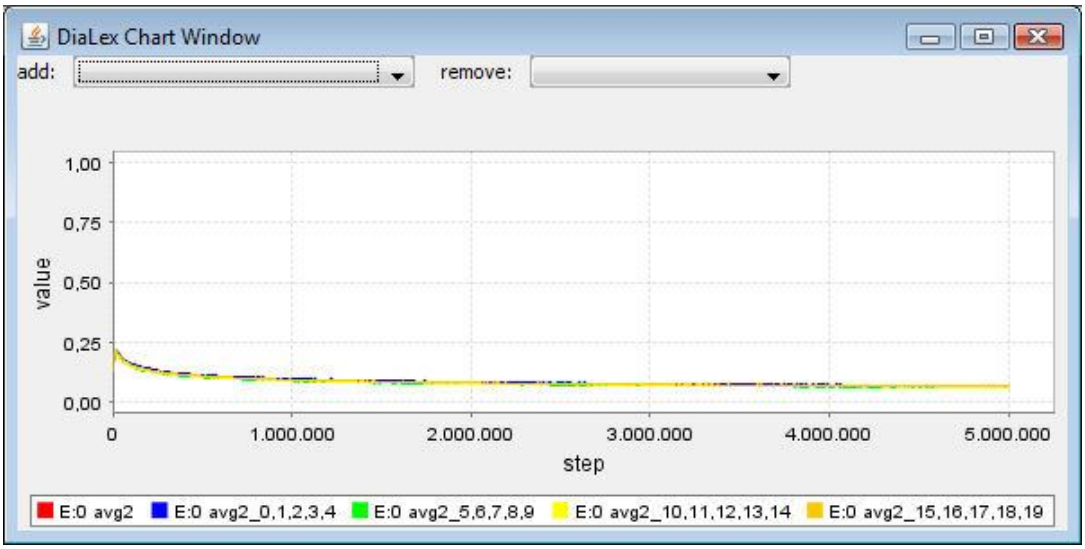


Abb. 68 Graph der Avg2-Werte

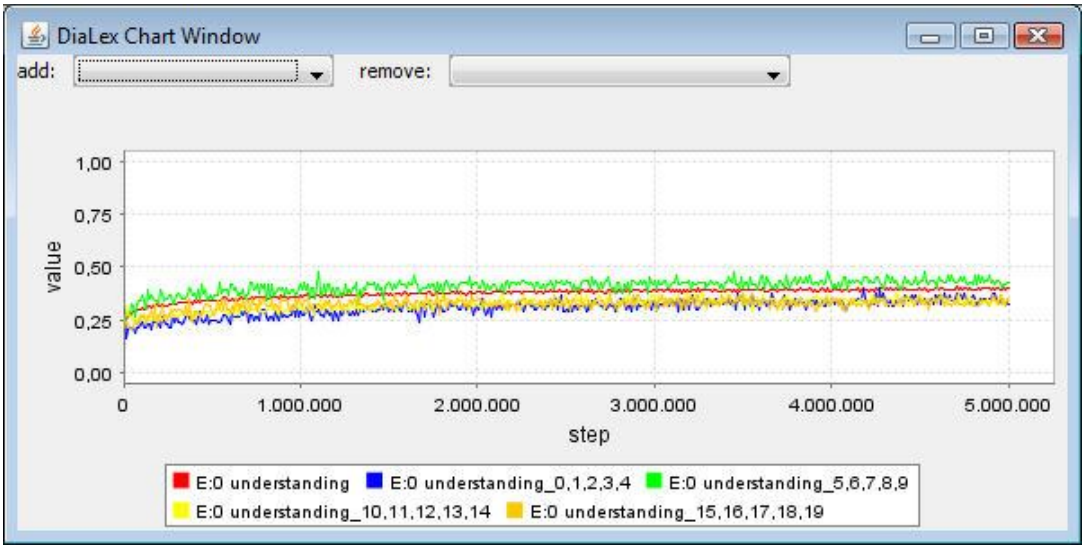


Abb. 69 Graph der Understanding-Werte

[16] Eine genaue Grenze zu ziehen ist nicht möglich, da selbst bei gleichen Werten in verschiedenen Simulationsdurchläufen konträre Ergebnisse erzielt werden.

Die verbalen Repräsentationen unterscheiden sich nicht stark genug, um von einem Wortschatz zu sprechen, was sich im Graphen zum Avg1-Indikator widerspiegelt. Vielmehr entsteht zu jedem beliebigen Eingabemuster ein verrauschtes „Einheitswort“.

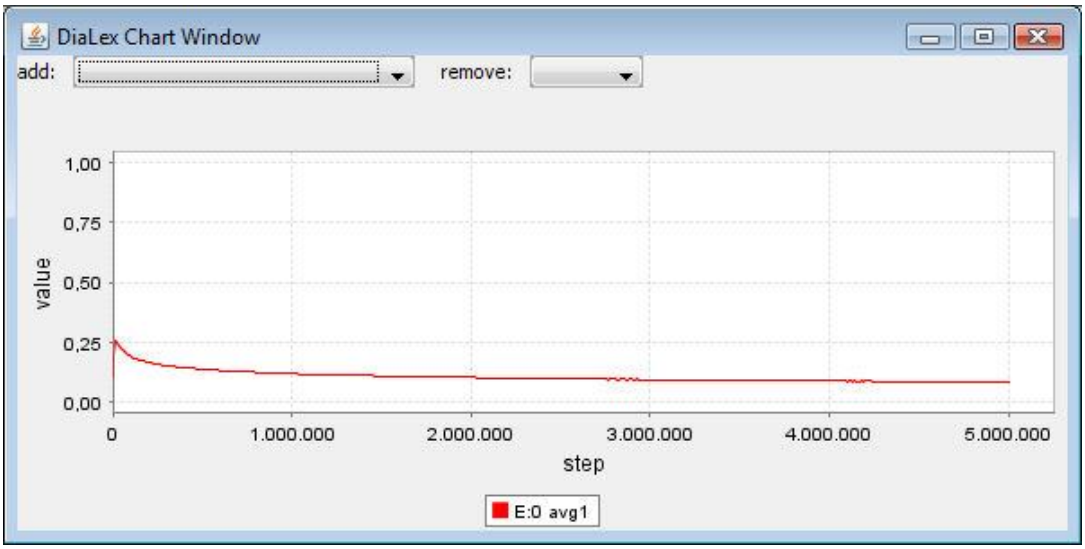


Abb. 70 Graph der Avg1-Werte

4.3 Simulation 3: Rauschen während der Kommunikation

In diesem Unterkapitel werden die Auswirkungen des Störfaktors Rauschen während der Agentenkommunikation auf die Simulationsergebnisse untersucht. Es ermöglicht somit, zusammen mit dem vorangegangenen Unterkapitel, einen Vergleich der Auswirkungen einer identischen Rauschfunktion an verschiedenen Stellen des neuronalen Netzes.

4.3.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }

- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2
  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen
- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge S
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Es existiert keine Sterbewahrscheinlichkeit für die Agenten
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert ein Rauschen innerhalb der Kommunikation der Stärke 0.62.
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.3.2 Analyse

Ein Rauschen während der Kommunikation zweier Agenten bewirkt entgegen aller Erwartungen aufgrund der Ähnlichkeit zu den Simulationsdurchläufen in Unterkapitel 4.2 in keinem Fall eine Ausbildung von Dialekten. Vielmehr teilt sich der Definitionsbereich der Rauschfunktion in nur zwei Intervalle mit verschiedenen Auswirkungen.

Rauschfaktor 0.62 - 1

Werte zwischen 0.0 und ungefähr 0.62 stören zwar die Kommunikation, was sich in einer, mit zunehmender Stärke des Störfaktors stets größer werdenden Dauer des Erreichens der Ereignisschwelle von Understanding > 0.75 niederschlägt, sie bewirken jedoch danach keine Auseinanderentwicklung der Sprachen der Agentenuntergruppen. Dies zeigt sich in den Graphen zu Avg2 und Understanding. (vgl. Abbs. 71 und 72) Der gruppeninterne Verständigungserfolg bleibt über die gesamte Simulationsdauer genau so groß wie der gruppenexterne Verständigungserfolg. Ein Zerfall in Dialektgruppen ist nicht erkennbar.

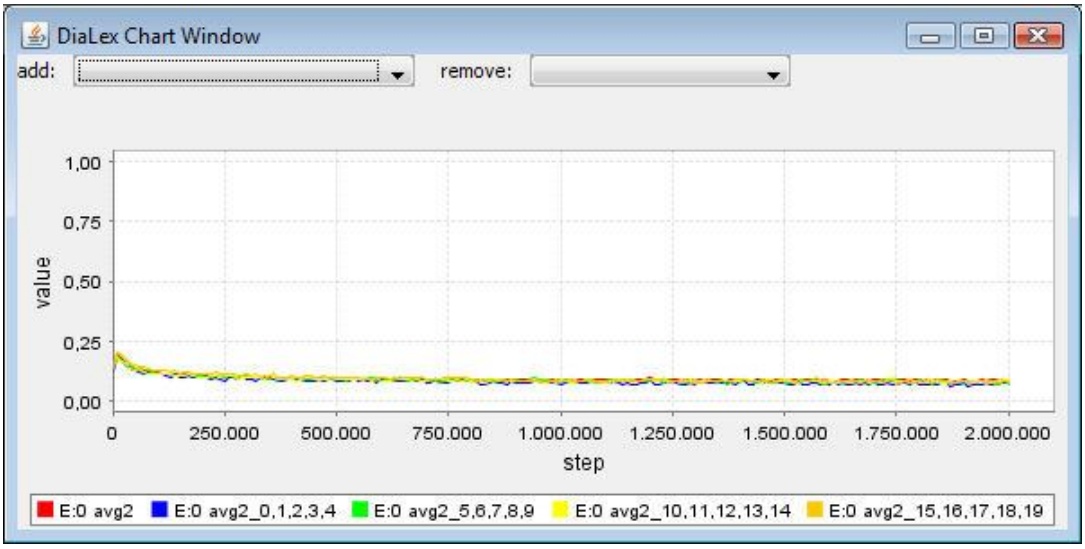


Abb. 71 Graph der Avg2-Werte

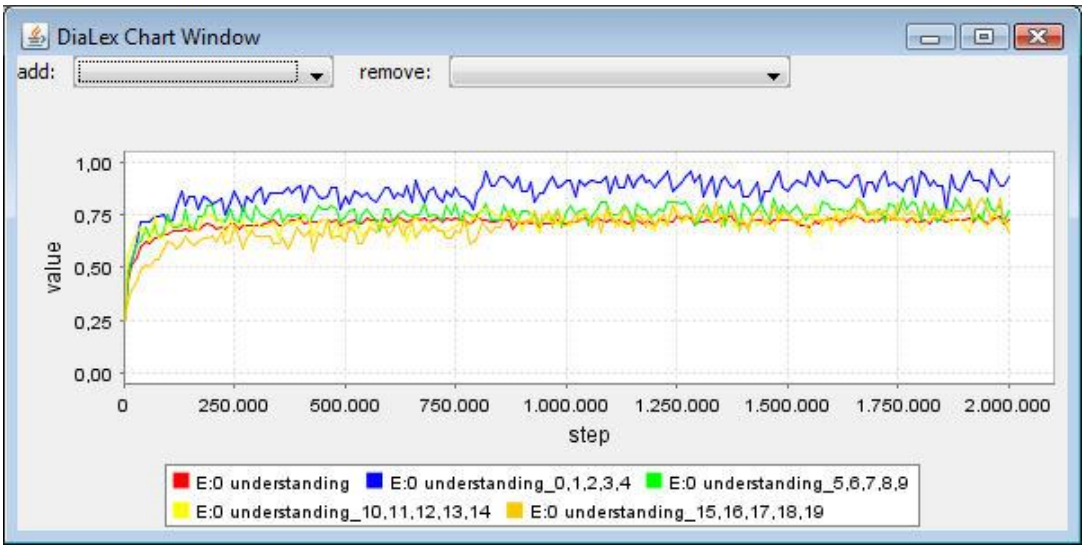


Abb. 72 Graph der Understanding-Werte

Rauschfaktor 0.62 – 2

Bei Simulationsdurchläufen mit Werten des Störfaktors im Intervall von ungefähr 0.62 bis 1.00 jedoch zeigt sich ein anderes Phänomen. Die Agentenpopulation ist wegen des starken Rauschens nicht mehr in der Lage, die Bedingung zum Eintritt des Ereignisses zu erreichen, womit sich die Agentenpopulation nicht in die vordefinierten Untergruppen aufspaltet. Die nachfolgenden Abbildungen zeigen die Avg2- und Understanding-Graphen eines anderen Simulationsdurchlaufs mit identischen Simulationsparametern.

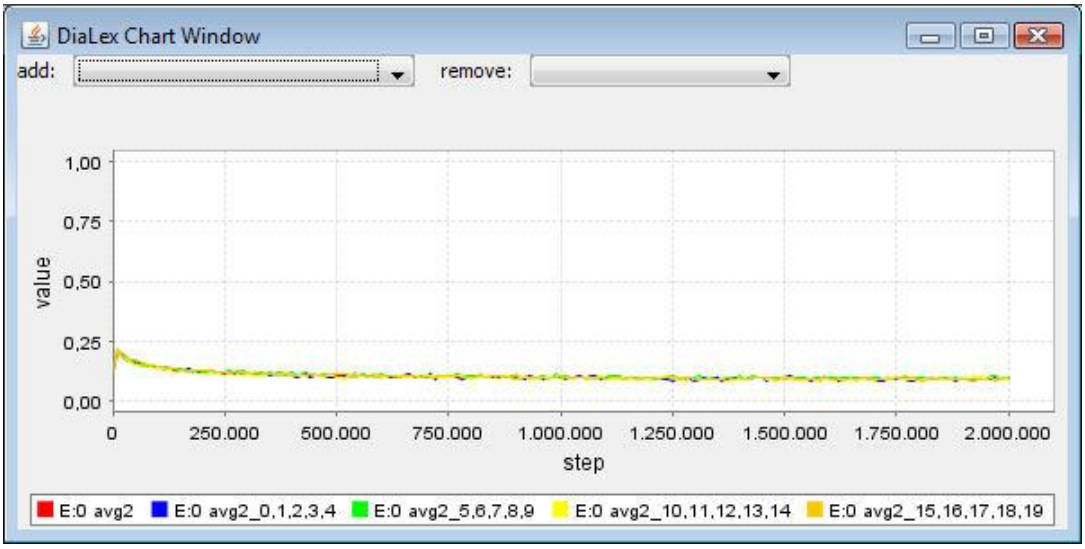


Abb. 73 Graph der Avg2-Werte

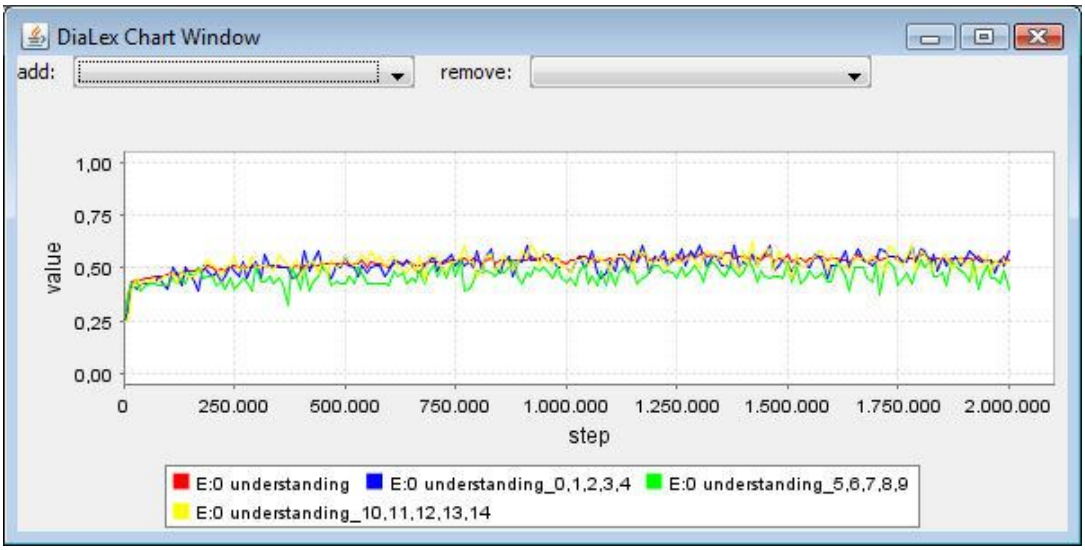


Abb. 74 Graph der Understanding-Werte

Deutlich zu erkennen sind die nahezu identischen Kurvenverläufe der beiden Avg2-Graphen. Alleine an ihnen wäre kein Unterschied im Simulationsverlauf erkennbar. Die Graphen des gegenseitigen Verständnisses jedoch unterscheiden sich stark: Erreicht der erste Simulationsdurchlauf zügig ein Verständnis von 0.75, was eine Aufteilung in Gruppen auslöst, ist dies im zweiten Simulationsdurchlauf während der gesamten 2000000 Schritte nicht gelungen. Vielmehr stagnieren die Werte in einem Intervall zwischen 0.4 und 0.55. Eine Dialektspaltung ist durch die nicht erfolgte Gruppenaufteilung ebenfalls nicht erfolgt. Auf eine Abbildung der entsprechenden Tabellen wird demnach verzichtet.

Die Analyse der beiden in diesem Unterkapitel aufgeführten Simulationsdurchläufe zeigt jedoch einen interessanten Effekt. Obwohl die Simulationsparameter absolut identisch sind, zeigt sich ein jeweils komplett anderes Verhalten. Dies lässt sich nur durch den Einfluss des Zufalls auf die Simulation erklären. Zufällig ist dabei in diesem Fall die Sprecherauswahl, die Zuhörerauswahl und die Auswahl und Stärke des Rauschens innerhalb des Sprachpatterns.

Ebenso zeigt sich, dass es keine absoluten Grenzen innerhalb eines, wenn auch nur teilweise, durch Zufall beeinflussten Werteintervalls geben kann. Vielmehr existiert eine Intervallgrenze, oberhalb derer der Eintritt eines Simulationsverlaufs wahrscheinlicher ist als bei Werten unterhalb dieser Grenze. In dem hier gezeigten Beispiel liegt diese Grenze auf Basis der Erfahrungen mehrerer Dutzend identischer Simulationsdurchläufe bei ungefähr 0.62.

4.4 Simulation 4: Sterblichkeit der Agenten

In diesem Unterkapitel werden die Ergebnisse einer Reihe von Simulationsdurchläufen mit einer Agentensterblichkeit erläutert. In dem Falle des Todes eines Agenten wird sein komplettes neuronales Netz entsprechend der zu Simulationsbeginn vorgegebenen Parameter neu initialisiert. Die Anzahl der Agenten bleibt somit konstant.

4.4.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert

- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2
  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen
- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge S
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Die Sterbewahrscheinlichkeit für die Agenten liegen bei 0.000005, 0.00001 und 0.0005.
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen

- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.4.2 Analyse

Genau wie in Unterkapitel 4.3 bildet dieser Störfaktor nur zwei verschiedenartige Werteintervalle aus. In diesem Falle existiert jedoch ein erfolgreicher Zerfall der Sprache in Dialekte, nach einer erfolgreichen Ausbildung eines gemeinsamen Wortschatzes, bei Störfaktoren im Intervall von Werten größer 0 bis zu einem Wert von ungefähr 0.0003. Ist der Störfaktor stärker, so ist die Agentenpopulation nicht in der Lage, eine gemeinsame Ursprache zu bilden, eine Aufteilung in Agentenuntergruppen findet daher nicht statt.

Sterbewahrscheinlichkeit 0.000005

Bei einer relativ kleinen Sterbewahrscheinlichkeit wie in diesem Simulationsdurchlauf spielt sich die Dialektspaltung innerhalb der Agentengemeinschaft sehr langsam ab. Dies ist in den beiden Abbildungen 76 und 77 ersichtlich, die die Graphen der Avg2 und Understanding Werte zeigen. So steigt der Avg2-Wert der Agentengemeinschaft im Laufe der Simulation nur minimal von 0.18 auf 0.23 an, während die Avg2-Werte der Untergruppen nahezu bei 0 verharren. Beim Graphen des gegenseitigen Verständnisses ist die Auseinanderentwicklung der verbalen Repräsentationen besser erkennbar. Hier sinkt der Understanding-Wert der kompletten Gemeinschaft von 0.95 zu Beginn des Simulationsdurchlaufs bis zum Schritt 2000000 auf 0.80. Das Verständnis innerhalb der Agentenuntergruppen verändert sich dabei ebenfalls nicht und bleibt konstant bei einem vollständigen Verständnis von 1.

Wichtig in diesem Zusammenhang ist ebenfalls der Graph der Avg1-Werte in Abbildung 75. Der Avg1 bleibt konstant in einem Intervall von 0.5 bis 0.6 und zeigt damit eine ausreichend große Wortdifferenz.

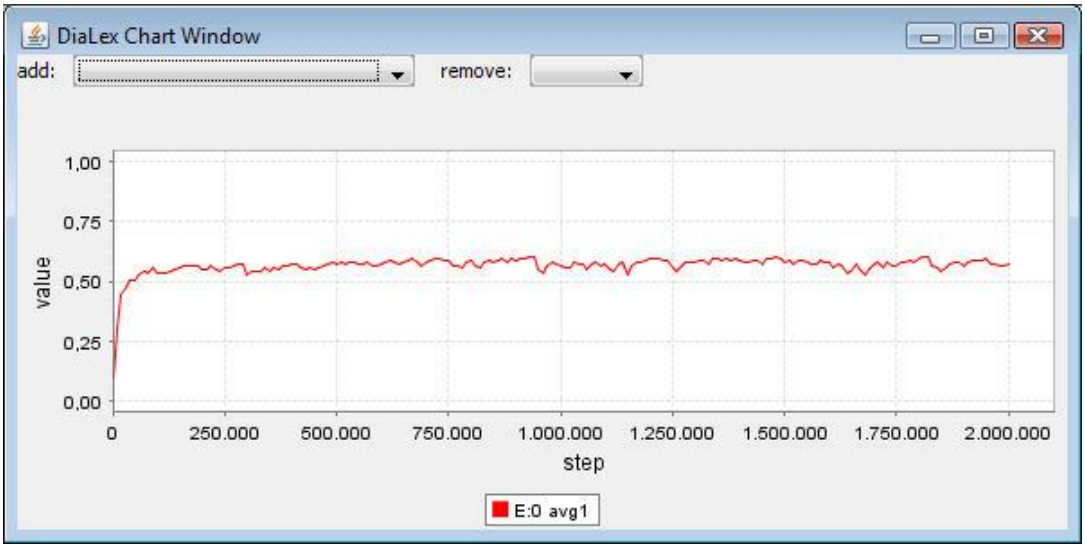


Abb. 75 Graph der Avg1-Werte

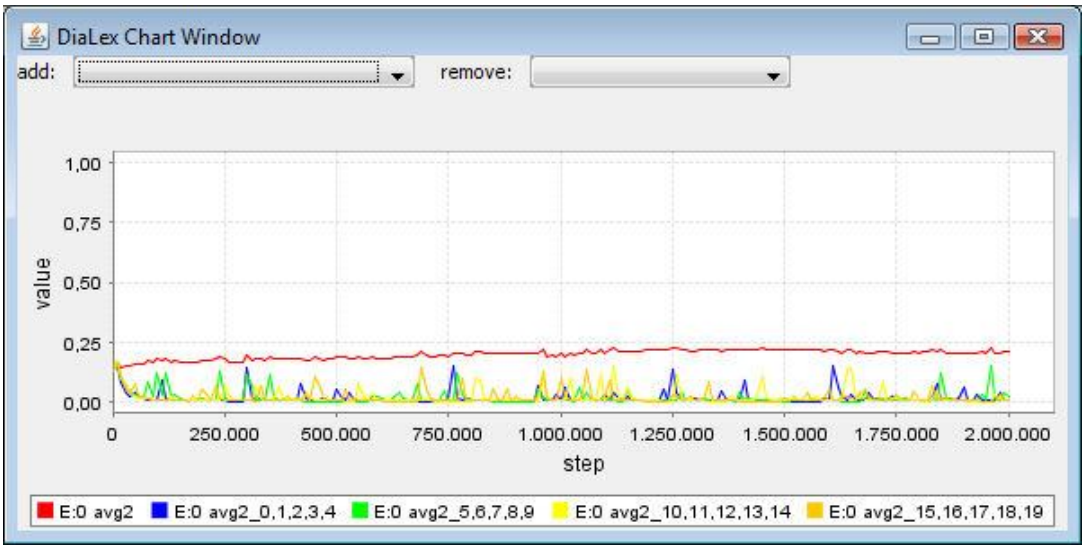


Abb. 76 Graph der Avg2-Werte

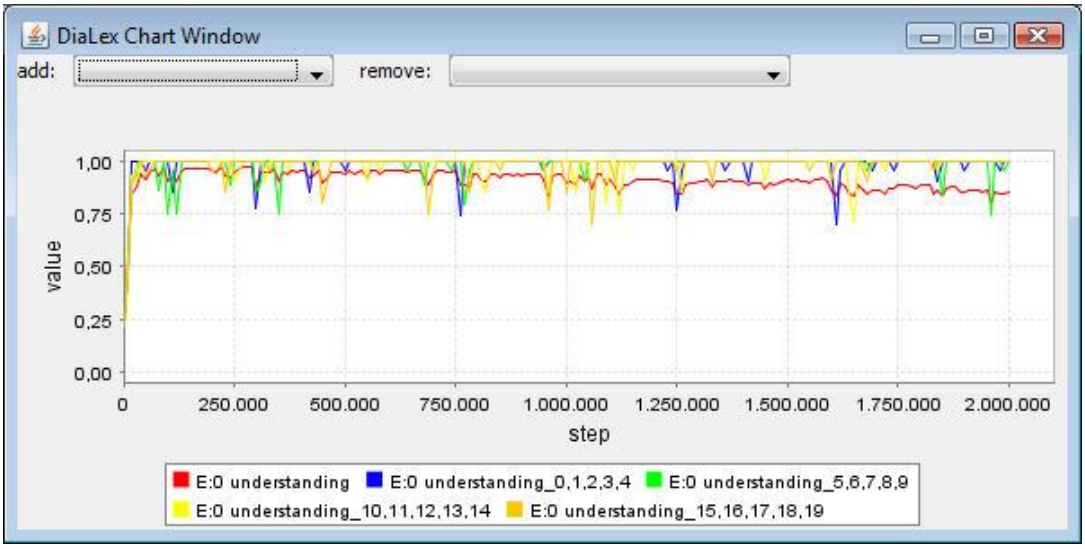


Abb. 77 Graph der Understanding-Werte

Die beiden zugehörigen Tabellen zeigen deutlich die Bildung von vier Agentengruppen mit unterschiedlichen Wortschätzen, die sich, abgesehen von den Lexika der Agentengruppen 2 (Agent 5-9) und 3 (Agent 10-14), stets mindestens um den Avg2-Wert 0.23 unterscheiden. Die beiden genannten Agentengruppen teilen auch zum Ende des Simulationsdurchlaufs einen nahezu identischen Wortschatz.

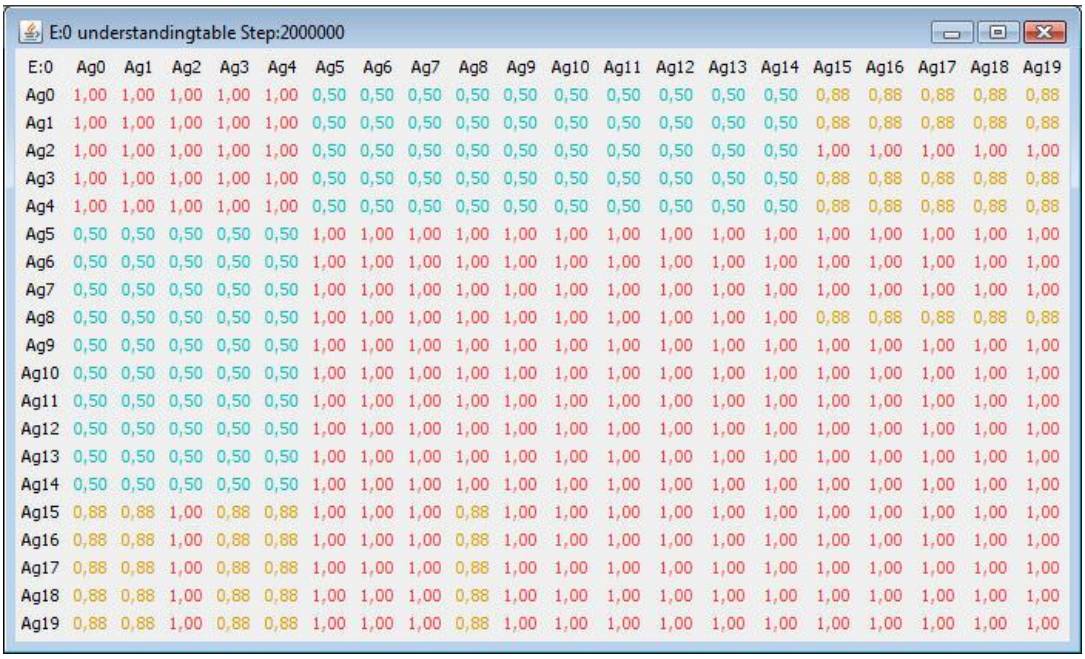
E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,01	0,00	0,01	0,00	0,34	0,34	0,34	0,34	0,34	0,35	0,36	0,36	0,36	0,35	0,32	0,32	0,32	0,33	0,32
Ag1	0,01	0,00	0,01	0,00	0,01	0,34	0,35	0,34	0,34	0,34	0,36	0,36	0,36	0,36	0,36	0,33	0,33	0,33	0,33	0,33
Ag2	0,00	0,01	0,00	0,01	0,00	0,34	0,34	0,34	0,34	0,34	0,35	0,36	0,36	0,36	0,35	0,32	0,32	0,32	0,33	0,32
Ag3	0,01	0,00	0,01	0,00	0,01	0,34	0,35	0,34	0,34	0,35	0,36	0,36	0,36	0,36	0,36	0,33	0,33	0,33	0,33	0,33
Ag4	0,00	0,01	0,00	0,01	0,00	0,34	0,34	0,34	0,34	0,34	0,35	0,36	0,36	0,36	0,35	0,32	0,32	0,32	0,33	0,32
Ag5	0,34	0,34	0,34	0,34	0,34	0,00	0,02	0,01	0,01	0,02	0,07	0,07	0,07	0,07	0,06	0,24	0,24	0,24	0,24	0,24
Ag6	0,34	0,35	0,34	0,35	0,34	0,02	0,00	0,03	0,02	0,03	0,08	0,08	0,08	0,08	0,08	0,24	0,24	0,24	0,24	0,24
Ag7	0,34	0,34	0,34	0,34	0,34	0,01	0,03	0,00	0,01	0,01	0,06	0,07	0,07	0,06	0,06	0,24	0,24	0,24	0,24	0,24
Ag8	0,34	0,34	0,34	0,34	0,34	0,01	0,02	0,01	0,00	0,02	0,07	0,07	0,07	0,07	0,07	0,24	0,24	0,24	0,24	0,24
Ag9	0,34	0,34	0,34	0,35	0,34	0,02	0,03	0,01	0,02	0,00	0,06	0,06	0,06	0,06	0,06	0,23	0,23	0,23	0,24	0,23
Ag10	0,35	0,36	0,35	0,36	0,35	0,07	0,08	0,06	0,07	0,06	0,00	0,01	0,01	0,00	0,00	0,23	0,23	0,23	0,23	0,23
Ag11	0,36	0,36	0,36	0,36	0,36	0,07	0,08	0,07	0,07	0,06	0,01	0,00	0,00	0,00	0,01	0,23	0,23	0,23	0,23	0,23
Ag12	0,36	0,36	0,36	0,36	0,36	0,07	0,08	0,07	0,07	0,06	0,01	0,00	0,00	0,01	0,01	0,23	0,23	0,23	0,23	0,23
Ag13	0,36	0,36	0,36	0,36	0,36	0,07	0,08	0,06	0,07	0,06	0,00	0,00	0,01	0,00	0,00	0,23	0,23	0,23	0,23	0,23
Ag14	0,35	0,36	0,35	0,36	0,35	0,06	0,08	0,06	0,07	0,06	0,00	0,01	0,01	0,00	0,00	0,23	0,23	0,23	0,23	0,23
Ag15	0,32	0,33	0,32	0,33	0,32	0,24	0,24	0,24	0,24	0,23	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,01	0,00
Ag16	0,32	0,33	0,32	0,33	0,32	0,24	0,24	0,24	0,24	0,23	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,01	0,00
Ag17	0,32	0,33	0,32	0,33	0,32	0,24	0,24	0,24	0,24	0,23	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,01	0,00
Ag18	0,33	0,33	0,33	0,33	0,33	0,24	0,24	0,24	0,24	0,24	0,23	0,23	0,23	0,23	0,23	0,01	0,01	0,01	0,00	0,01
Ag19	0,32	0,33	0,32	0,33	0,32	0,24	0,24	0,24	0,24	0,23	0,23	0,23	0,23	0,23	0,23	0,00	0,00	0,00	0,01	0,00

Abb. 78 Avg2-Tabelle in Simulationsschritt 2000000

Obwohl die Beobachtungen aus der Avg2-Tabelle ein stark eingeschränktes Verständnis von Agenten anderer Untergruppen erwarten lassen würden, zeigt Abb. 79, die Tabelle des gegenseitigen Verständnisses, ein anderes Bild. Mit Ausnahme der Kombinationen Gruppe 1 und 2 und Gruppe 1 und 3 erfolgt die Kommunikation zwischen allen Agenten nahezu perfekt.

Dies überrascht, vor allem, da die durchschnittliche Wortdifferenz (Avg2) zwischen Gruppe 1 und 4 mit 0.32 kaum niedriger ist als zwischen Gruppe 1 und 2 mit 0.34. Diese Grenze ist jedoch von Simulationsdurchlauf zu Simulationsdurchlauf verschieden, wie sich im Folgenden zeigen wird.



E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,88	0,88	0,88	0,88	0,88
Ag1	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,88	0,88	0,88	0,88	0,88
Ag2	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag3	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,88	0,88	0,88	0,88	0,88
Ag4	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,88	0,88	0,88	0,88	0,88
Ag5	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag6	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag7	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag8	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,88	0,88	0,88
Ag9	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag10	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag11	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag12	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag13	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag14	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag15	0,88	0,88	1,00	0,88	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag16	0,88	0,88	1,00	0,88	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag17	0,88	0,88	1,00	0,88	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag18	0,88	0,88	1,00	0,88	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Ag19	0,88	0,88	1,00	0,88	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Abb. 79 Understanding-Tabelle in Simulationsschritt 2000000

Sterbewahrscheinlichkeit 0.00001

Dieser Simulationsdurchlauf mit einer Sterbewahrscheinlichkeit von 0.00001 zeigt eine nahezu perfekte Simulation einer Sprachspaltung. Der Avg1-Wert liegt konstant in einem Intervall zwischen 0.5 und 0.55, damit unterscheiden sich die verbalen Repräsentationen stark genug.

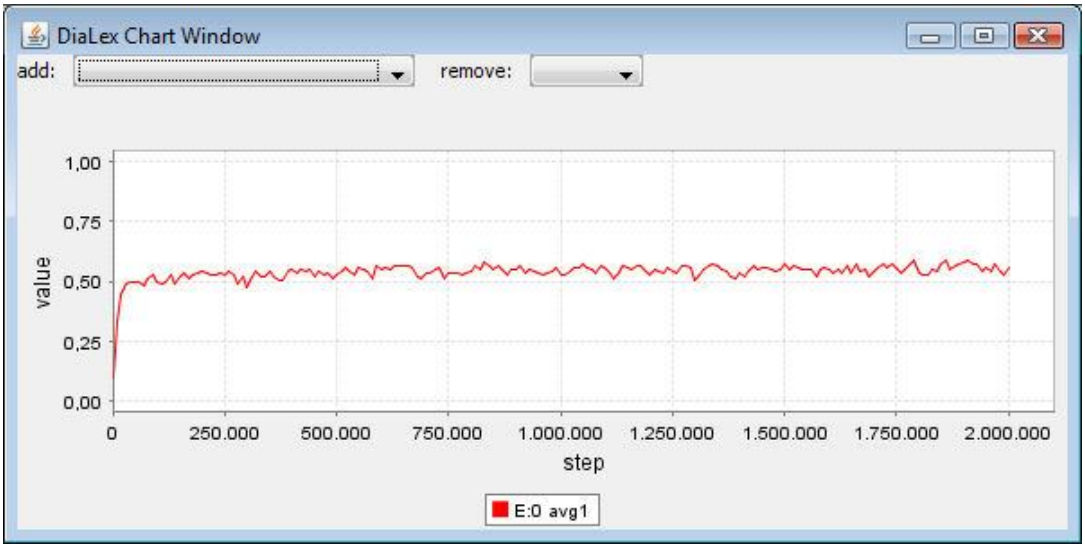


Abb. 80 Graph der Avg1-Werte

Der Avg2-Graph zeigt das gewünschte Bild: Die Werte der einzelnen Agentenuntergruppen befinden sich permanent, das heisst, mit Ausnahme der „Ausreißer“, die durch neu initialisierte Agenten verursacht werden, in einem Bereich nahe 0.0. Im Gegensatz dazu wächst der Avg2-Wert der gesamten Population konstant im Laufe der Simulation an, von 0.15 zu Beginn der Gruppenaufteilung bis zu 0.32 im letzten Simulationsschritt.

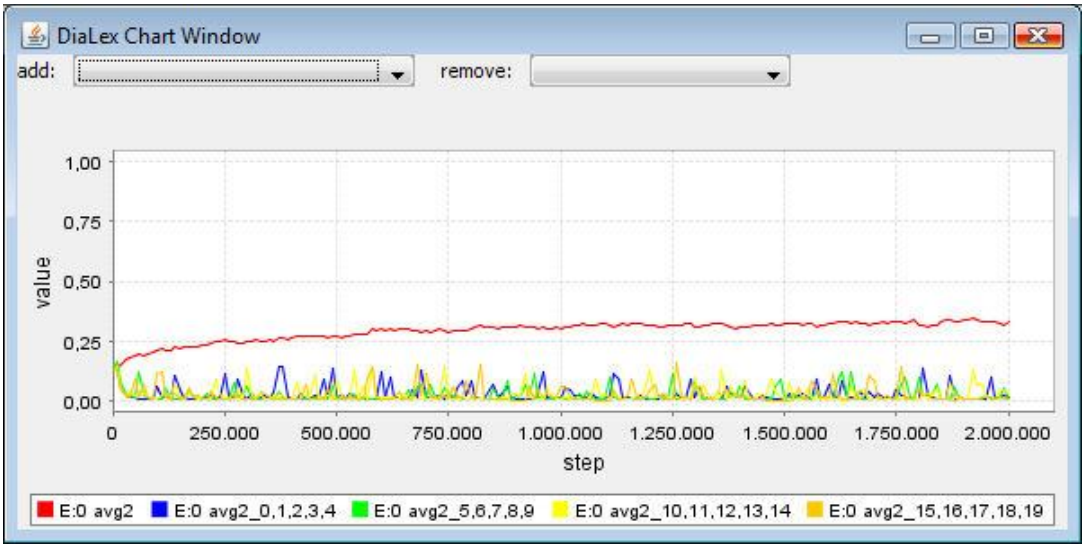


Abb. 81 Graph der Avg2-Werte

Der Graph des gegenseitigen Verständnisses zeichnet ein nahezu gespiegeltes Bild. Während hier die Understanding-Werte innerhalb der Agentengruppen bei Werten nahe 1.0 verharren, sinkt der Wert des gemeinschaftlichen Verständnisses im Laufe des Simulationsdurchlaufs von 0.95 auf einen Wert minimal größer als 0.5. Somit ist eine erfolgreiche Kommunikation zwischen Agenten verschiedener Gruppen nicht mehr zu erwarten.

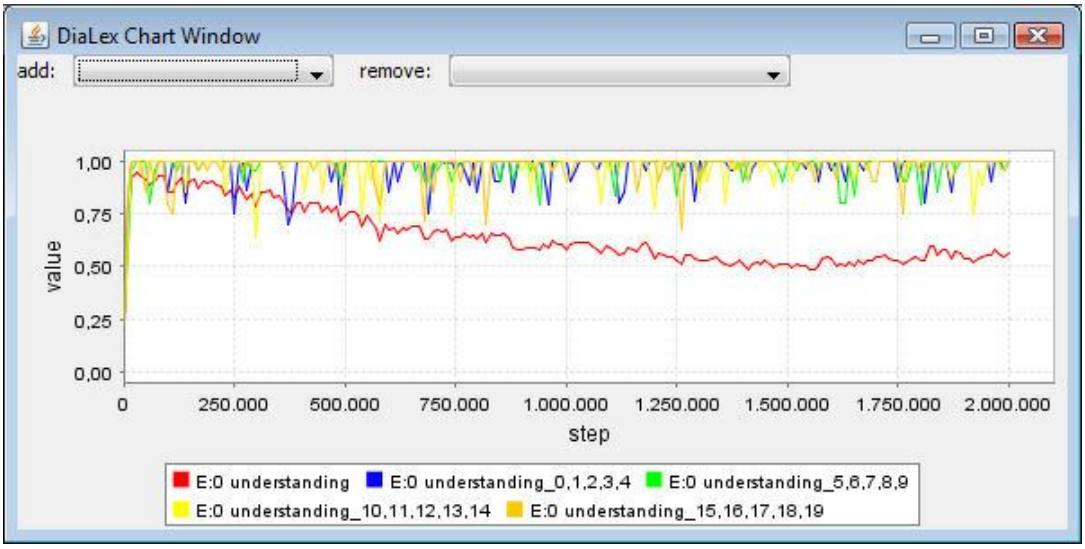


Abb. 82 Graph der Understanding-Werte

Dies belegt ebenfalls der Vergleich der Wortmuster einer visuellen Szene von Agenten verschiedener Gruppenzugehörigkeit über die gesamte Länge des Simulationsdurchlaufs. Die verbalen Repräsentationen entfernen sich stetig voneinander fort.

Die nach Betrachtung der Graphen zu erwartende Gruppeneinteilung der Agenten wird in Abb. 84 deutlich. Während die Agenten der vier Gruppen untereinander einen nahezu identischen Wortschatz teilen, unterscheidet er sich stark von dem der Agenten anderer Gruppen.

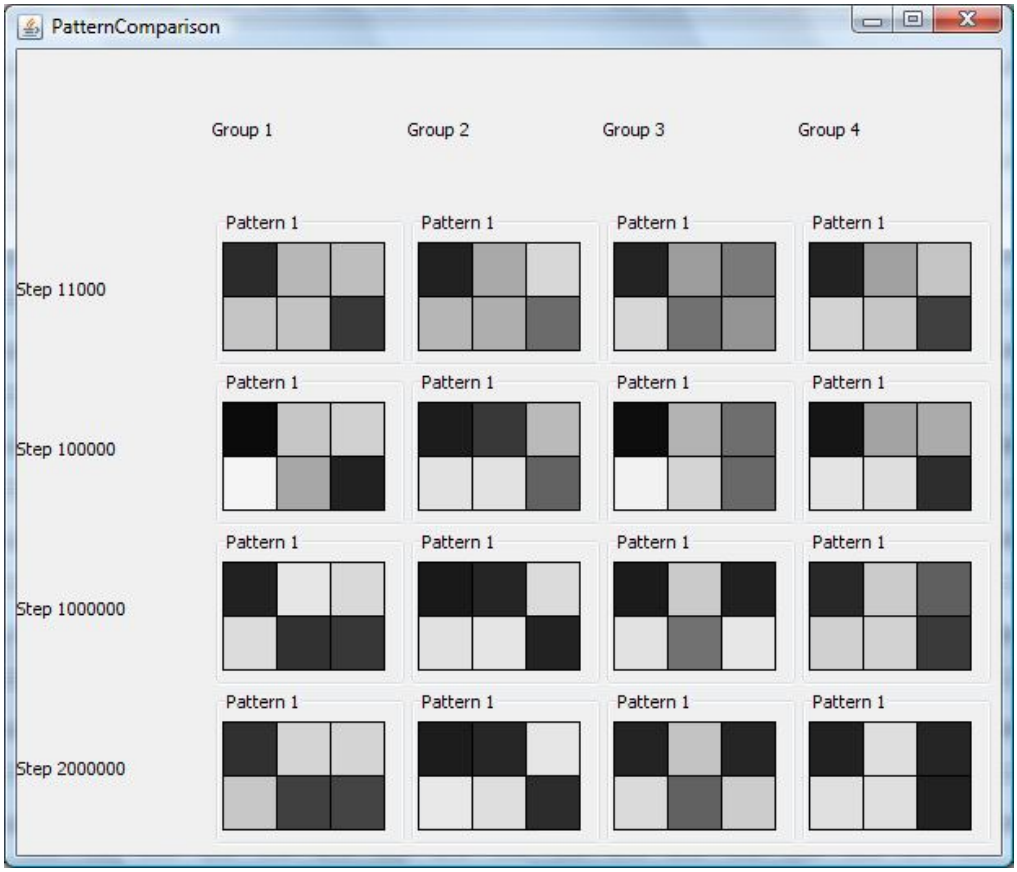


Abb. 83 Vergleich der Wortmuster

E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,01	0,01	0,00	0,02	0,43	0,44	0,44	0,43	0,43	0,30	0,29	0,28	0,29	0,29	0,40	0,40	0,40	0,40	0,40
Ag1	0,01	0,00	0,01	0,01	0,01	0,44	0,44	0,44	0,44	0,44	0,30	0,30	0,28	0,30	0,30	0,40	0,40	0,41	0,41	0,40
Ag2	0,01	0,01	0,00	0,01	0,01	0,43	0,44	0,44	0,44	0,44	0,30	0,30	0,28	0,30	0,30	0,40	0,40	0,41	0,41	0,40
Ag3	0,00	0,01	0,01	0,00	0,02	0,43	0,43	0,44	0,43	0,43	0,30	0,29	0,28	0,29	0,29	0,40	0,40	0,40	0,40	0,40
Ag4	0,02	0,01	0,01	0,02	0,00	0,44	0,44	0,44	0,44	0,44	0,31	0,30	0,29	0,30	0,30	0,41	0,41	0,41	0,41	0,41
Ag5	0,43	0,44	0,43	0,43	0,44	0,00	0,01	0,01	0,00	0,00	0,47	0,47	0,46	0,47	0,47	0,44	0,44	0,45	0,44	0,44
Ag6	0,44	0,44	0,44	0,43	0,44	0,01	0,00	0,01	0,01	0,00	0,47	0,48	0,46	0,47	0,47	0,45	0,45	0,45	0,45	0,44
Ag7	0,44	0,44	0,44	0,44	0,44	0,01	0,01	0,00	0,01	0,01	0,48	0,48	0,46	0,48	0,48	0,45	0,45	0,45	0,45	0,45
Ag8	0,43	0,44	0,44	0,43	0,44	0,00	0,01	0,01	0,00	0,00	0,47	0,47	0,46	0,47	0,47	0,44	0,44	0,45	0,44	0,44
Ag9	0,43	0,44	0,44	0,43	0,44	0,00	0,00	0,01	0,00	0,00	0,47	0,47	0,46	0,47	0,47	0,44	0,44	0,45	0,45	0,44
Ag10	0,30	0,30	0,30	0,30	0,31	0,47	0,47	0,48	0,47	0,47	0,00	0,02	0,05	0,02	0,02	0,42	0,42	0,42	0,42	0,42
Ag11	0,29	0,30	0,30	0,29	0,30	0,47	0,48	0,48	0,47	0,47	0,02	0,00	0,05	0,01	0,01	0,42	0,42	0,42	0,42	0,42
Ag12	0,28	0,28	0,28	0,28	0,29	0,46	0,46	0,46	0,46	0,46	0,05	0,05	0,00	0,05	0,05	0,40	0,40	0,41	0,41	0,40
Ag13	0,29	0,30	0,30	0,29	0,30	0,47	0,47	0,48	0,47	0,47	0,02	0,01	0,05	0,00	0,01	0,41	0,41	0,42	0,42	0,41
Ag14	0,29	0,30	0,30	0,29	0,30	0,47	0,47	0,48	0,47	0,47	0,02	0,01	0,05	0,01	0,00	0,42	0,42	0,42	0,42	0,42
Ag15	0,40	0,40	0,40	0,40	0,41	0,44	0,45	0,45	0,44	0,44	0,42	0,42	0,40	0,41	0,42	0,00	0,00	0,01	0,00	0,00
Ag16	0,40	0,40	0,40	0,40	0,41	0,44	0,45	0,45	0,44	0,44	0,42	0,42	0,40	0,41	0,42	0,00	0,00	0,01	0,00	0,00
Ag17	0,40	0,41	0,41	0,40	0,41	0,45	0,45	0,45	0,45	0,45	0,42	0,42	0,41	0,42	0,42	0,01	0,01	0,00	0,00	0,01
Ag18	0,40	0,41	0,41	0,40	0,41	0,44	0,45	0,45	0,44	0,45	0,42	0,42	0,41	0,42	0,42	0,00	0,00	0,00	0,00	0,00
Ag19	0,40	0,40	0,40	0,40	0,41	0,44	0,44	0,45	0,44	0,44	0,42	0,42	0,40	0,41	0,42	0,00	0,00	0,01	0,00	0,00

Abb. 84 Avg2-Tabelle in Simulationsschritt 2000000

Dies wirkt sich auch auf das gegenseitige Verständnis der Agenten aus. Gruppeninterne Kommunikationsversuche enden demnach ausnahmslos erfolgreich mit einer Quote von 1.00, während dies mit gruppenfremden Agenten nicht möglich ist. Hier bewegen sich die Verständnisquoten in einem weiten Band zwischen einer Verständnisquote von 0.88, einer guten Verständigungsmöglichkeit bis hin zu 0.00, dem totalen Unverständnis.

Verglichen mit den Ergebnissen des vorherigen Simulationsdurchlaufs zeigt sich hier, dass kein direkter Zusammenhang zwischen den Werten des Fehlerindex Avg2 und des gegenseitigen Verständnisses vorhanden ist. Avg2-Werte von 0.32 ließen im vorherigen Simulationsdurchlauf noch eine nahezu fehlerfreie Verständigung zu, während hier Werte von 0.29 schon eine Verständigung nahezu unmöglich machen.

E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,62	0,50	0,62	0,75	0,75	0,50	0,62	0,50	0,75	0,62
Ag1	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,62	0,62	0,62	0,88	0,88	0,50	0,75	0,62	0,75	0,62
Ag2	1,00	1,00	1,00	1,00	1,00	0,38	0,50	0,38	0,38	0,38	0,62	0,50	0,62	0,75	0,88	0,50	0,75	0,50	0,75	0,62
Ag3	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,75	0,62	0,62	0,88	0,88	0,38	0,50	0,38	0,62	0,50
Ag4	1,00	1,00	1,00	1,00	1,00	0,38	0,50	0,38	0,38	0,38	0,62	0,50	0,50	0,75	0,75	0,50	0,75	0,62	0,75	0,62
Ag5	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,00	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag6	0,38	0,38	0,50	0,38	0,50	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,50	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag7	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,25	0,00	0,00	0,38	0,38	0,50	0,50	0,38
Ag8	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag9	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag10	0,62	0,62	0,62	0,75	0,62	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag11	0,50	0,62	0,50	0,62	0,50	0,00	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,38	0,38	0,38	0,38	0,25
Ag12	0,62	0,62	0,62	0,62	0,50	0,38	0,50	0,25	0,38	0,38	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50
Ag13	0,75	0,88	0,75	0,88	0,75	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag14	0,75	0,88	0,88	0,88	0,75	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag15	0,50	0,50	0,50	0,38	0,50	0,38	0,38	0,38	0,38	0,38	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag16	0,62	0,75	0,75	0,50	0,75	0,38	0,38	0,38	0,38	0,38	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag17	0,50	0,62	0,50	0,38	0,62	0,50	0,50	0,50	0,50	0,50	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag18	0,75	0,75	0,75	0,62	0,75	0,50	0,50	0,50	0,50	0,50	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag19	0,62	0,62	0,62	0,50	0,62	0,38	0,38	0,38	0,38	0,38	0,38	0,25	0,50	0,38	0,38	1,00	1,00	1,00	1,00	1,00

Abb. 85 Understanding-Tabelle in Simulationsschritt 2000000

Sterbewahrscheinlichkeit 0.0005

Der Simulationsdurchlauf mit einer Sterbewahrscheinlichkeit von 0.0005 zeigt die Auswirkungen einer zu hohen Sterbewahrscheinlichkeit auf eine Agentengemeinschaft. Sie ist aufgrund der zu kurzen Lebenserwartung der Agenten gar nicht fähig, ein gemeinsames Lexikon auszubilden, somit wird sie auch niemals in vier Untergruppen aufgeteilt.

Der Avg1-Wert bleibt konstant in bei einem, durch die zufällige Initialisierung der Verbindungsgewichtungen der neuronalen Netze der Agenten errechneten Wert von 0.12.

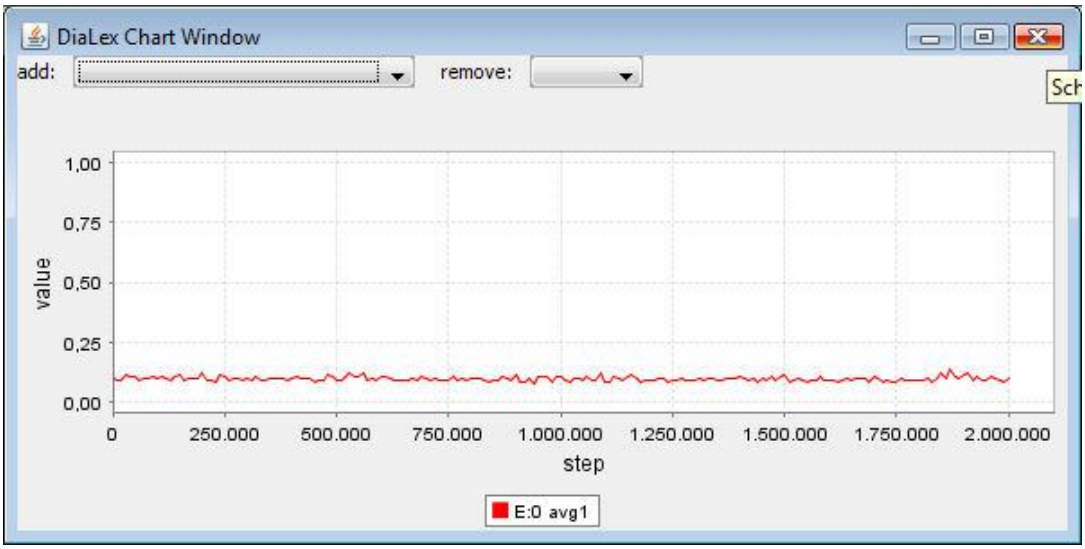


Abb. 86 Graph der Avg1-Werte

Der Graph der Avg2-Werte zeigt ein ähnliches Bild. Die Differenz der Wortrepräsentationen liegt konstant über alle Agentengruppen hinweg bei 0.12.

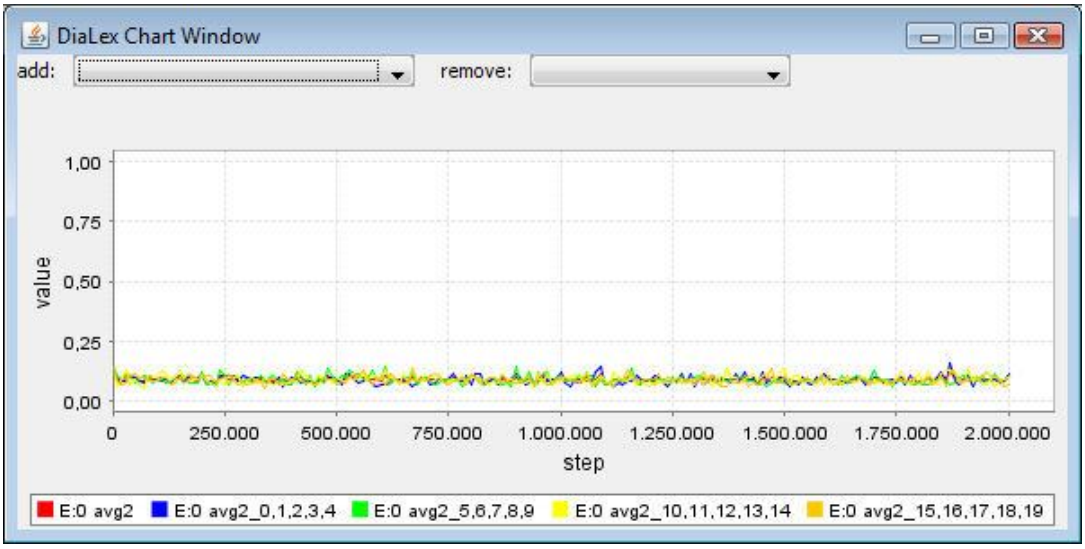


Abb. 87 Graph der Avg2-Werte

Dies lässt den Schluss zu, dass auch die Erfolgsquote einer Kommunikation zwischen zwei Agenten durch den Zufall bestimmt ist. Bei vier möglichen

Eingabemustern und somit vier verschiedenen Worten liegt die erwartete zufällige Erfolgsquote bei 0.25. Abbildung 88 zeigt konstant den hier zu erwartenden Wert.

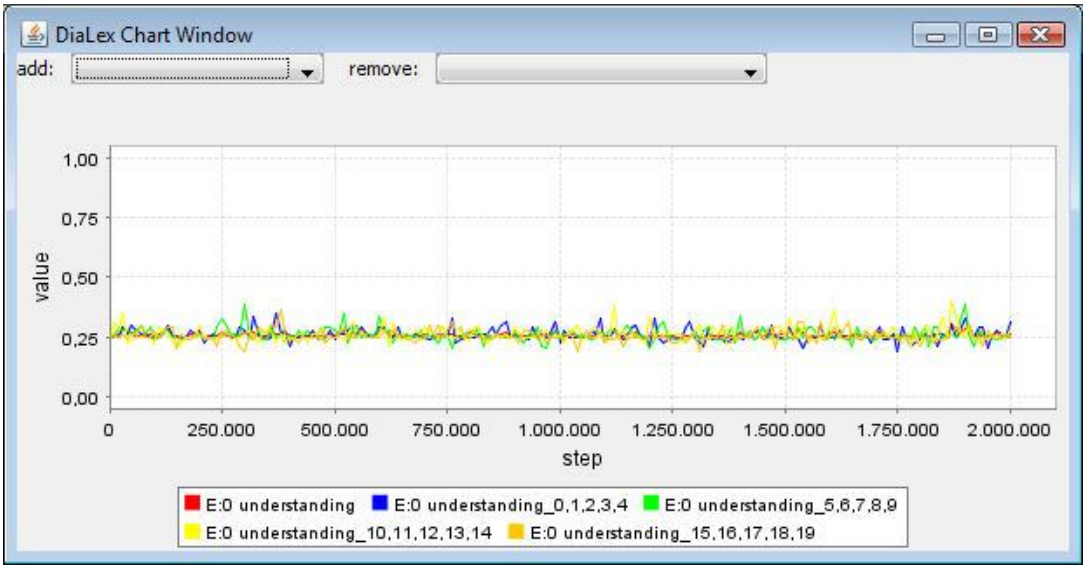


Abb. 88 Graph der Understanding-Werte

4.5 Simulation 5: Vergessen durch Abschwächen

In diesem Unterkapitel werden die Auswirkungen des Störfaktors Decay erläutert, der die Verbindungsgewichtungen der neuronalen Netze zufällig bis zu einem möglichen Maximalwert abschwächt. Diese so genannte Decay-Funktion wird explizit in Abschnitt B.1.1 erläutert.

4.5.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2
  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen
- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge S
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Es existiert keine Sterbewahrscheinlichkeit für die Agenten
- f_{dec} = Die Abschwächung der Verbindungsgewichtungen liegt bei 0.02.
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.5.2 Analyse

Wie bereits im vorherigen Unterkapitel lässt sich auch hier der Definitionsbereich für diesen Störfaktor in zwei Intervalle mit unterschiedlichen Auswirkungen zerlegen. Im Intervall (0; 0.06] ist die Agentengemeinschaft in der Lage, ein gemeinsames Lexikon auszubilden. Die Bedingung für die Aufspaltung der Agentengemeinschaft in einzelne Gruppen wird damit erfüllt. Dies wird beispielhaft an einem Simulationsdurchlauf mit einer Abschwächung der Stärke 0.02 gezeigt.

Ist die Abschwächung stärker, so sind die neuronalen Netze der Agenten nicht in der Lage, die gewünschten Informationen zu speichern. Das Ziel eines ersten gemeinsamen Wortschatzes wird nicht erreicht. Es zeigt sich das gleiche Bild wie im vorherigen Unterkapitel bei einer Sterberate von 0.0005. Es wird daher auf eine detailliertere Beschreibung an dieser Stelle verzichtet.

Abschwächung der Stärke 0.02

Als ersten Indikator für einen erfolgreichen Simulationsdurchlauf wird der Graph des Avg1 betrachtet. Es zeigt sich, dass sich die einzelnen verbalen Repräsentationen relativ konstant um einen Wert von 0.5 unterscheiden. Diese Differenz ist groß genug, um von einer konstanten Unterscheidbarkeit der verbalen Muster zu sprechen.

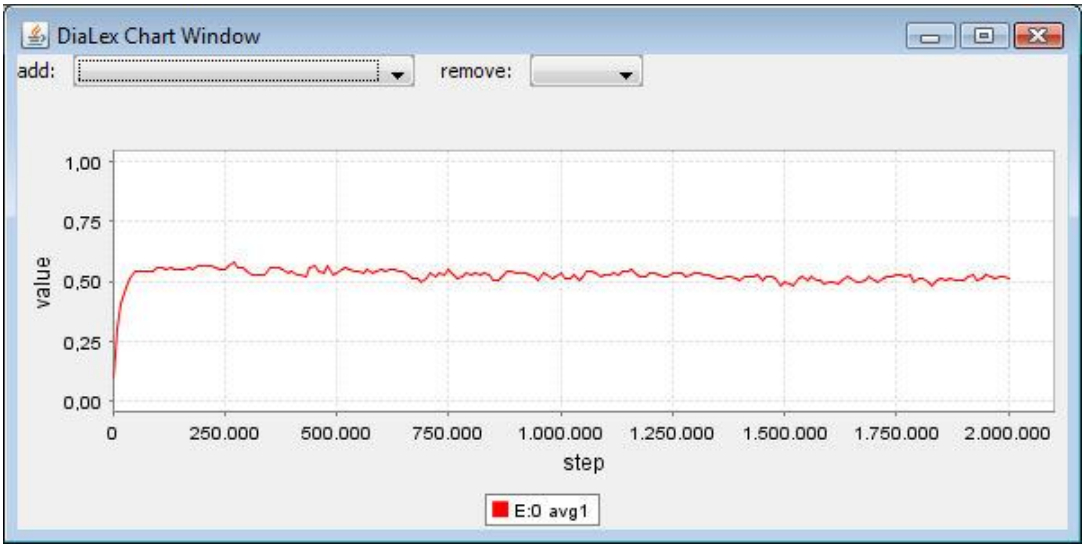


Abb. 89 Graph der Avg1-Werte

Die Graphen des Avg2 und des gegenseitigen Verständnisses zeigen das gewünschte Bild einer erfolgreichen Dialektspaltung. Bald nach der Aufteilung der Agentengemeinschaft in Gruppen entfernen sich die einzelnen Gruppenlexika voneinander. Dies zeigt sich durch ein Ansteigen des Avg2-Graphen der gesamten Population auf einen Wert von nahezu 0.4. Im Gegensatz dazu jedoch bleiben die Unterschiede der Lexika der Agenten einer Untergruppe sehr gering, was sich in einem Verharren der gruppeninternen Avg2-Kurven um einen Wert von 0.1 verdeutlicht.

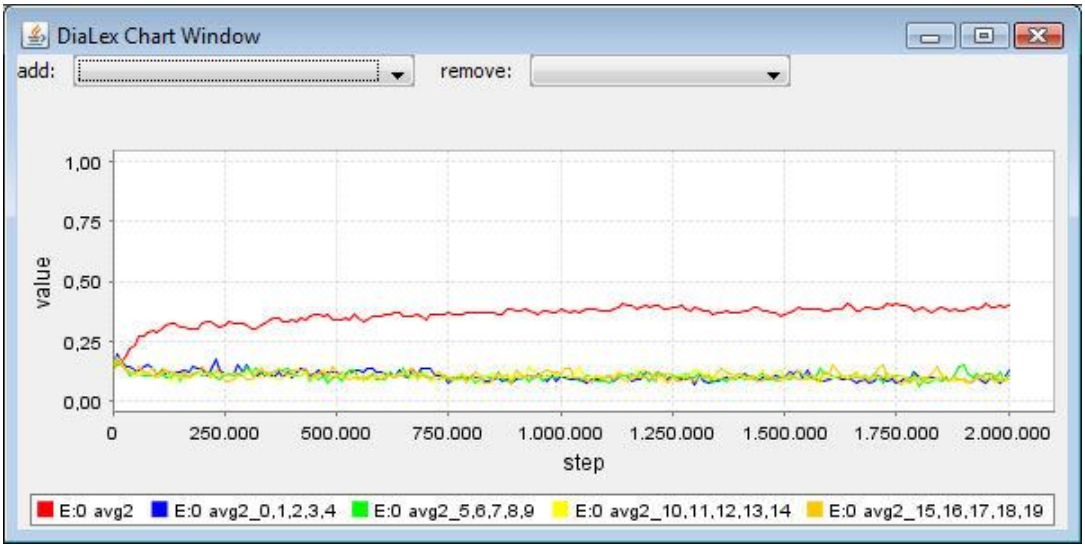


Abb. 90 Graph der Avg2-Werte

Verglichen mit den Avg2-Graphen der Simulationsdurchläufe anderer Störfaktoren sind die gruppeninternen Werte relativ hoch, was auf größere Differenzen der Wortrepräsentationen der einzelnen Agenten einer Gruppe schließen lässt. Dies erklärt sich jedoch durch die Funktionsweise der Decay-Funktion. Sie stört das genaue Abspeichern eines bestimmten Denkmusters und macht es damit unmöglich, dass sich zwei Agenten in nahezu identischer Weise an ein Eingabemuster erinnern.

Dieser Effekt findet sich auch im Graphen des gegenseitigen Verständnisses wieder. Im Gegensatz zu vielen vorherigen Simulationsdurchläufen mit anderen Störfaktoren läuft die gruppeninterne Verständigung in diesem Beispiel konstant schlechter. Werte nahe 1.0, welche eine fehlerfreie Verständigung symbolisieren, werden kaum erreicht. Vielmehr bewegen sich die gruppeninternen Understanding-Werte im Intervall [0.6; 1.0].

Da das gruppenexterne Verständnis im Laufe der Simulation jedoch auf einen Wert von 0.35 absinkt, kann dennoch von einer erfolgreichen Dialektspaltung gesprochen werden. Die Verständigung innerhalb einer Gruppe gelingt sehr viel besser als eine gruppenexterne Kommunikation.

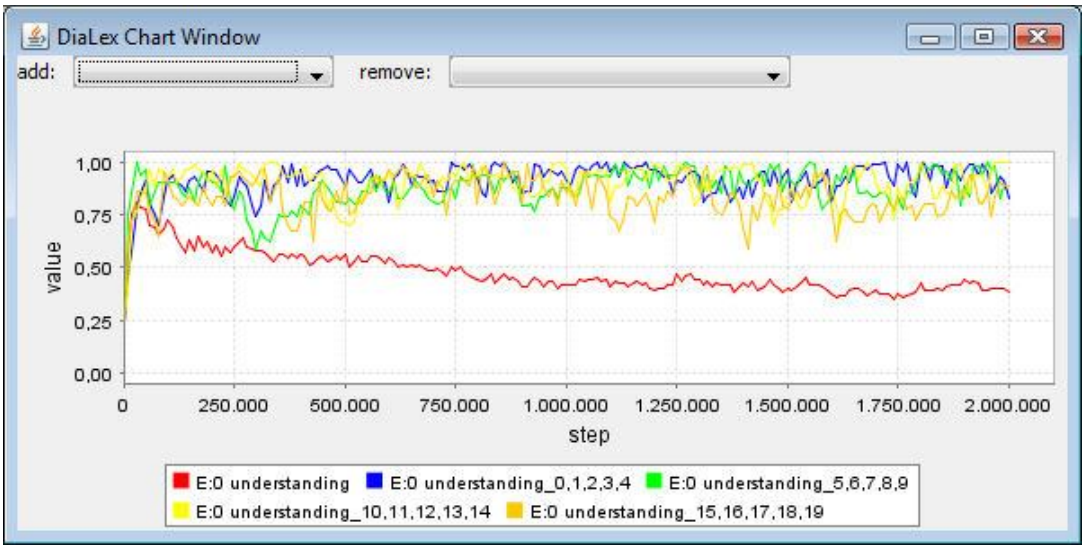


Abb. 91 Graph der Understanding-Werte

Dies zeigt sich auch bei einem Vergleich der Wortmuster einer visuellen Szene von Agenten verschiedener Gruppen über die Simulationsdistanz (vgl. Abb. 92). Die verbalen Repräsentationen entwickeln sich über die Zeit in verschiedene Richtungen. Die Avg2-Tabelle zum Simulationsschritt 2000000 (siehe Abb. 93) zeigt das gewünschte Bild einer deutlichen Gruppeneinteilung der Wortrepräsentationen der Agenten. Deutlich ist jedoch auch hier die leicht höhere Wortdifferenz der Agenten innerhalb einer Gruppe im Vergleich zu anderen Störfaktoren.

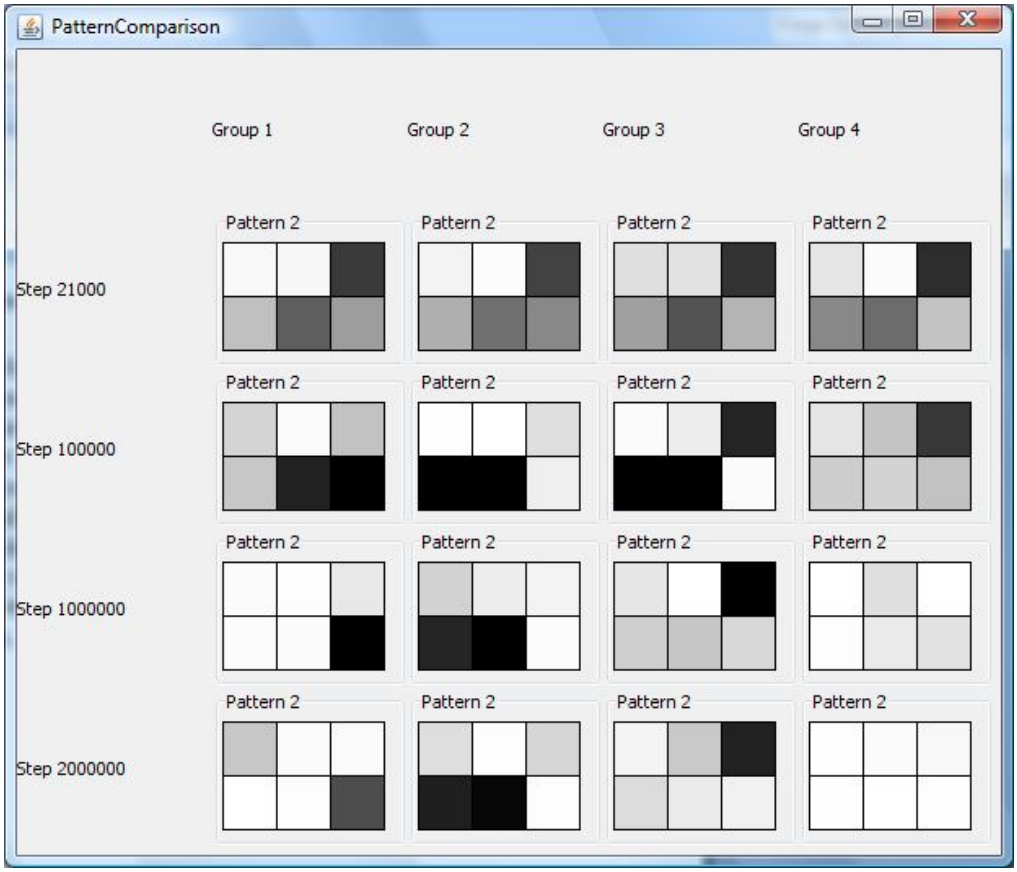


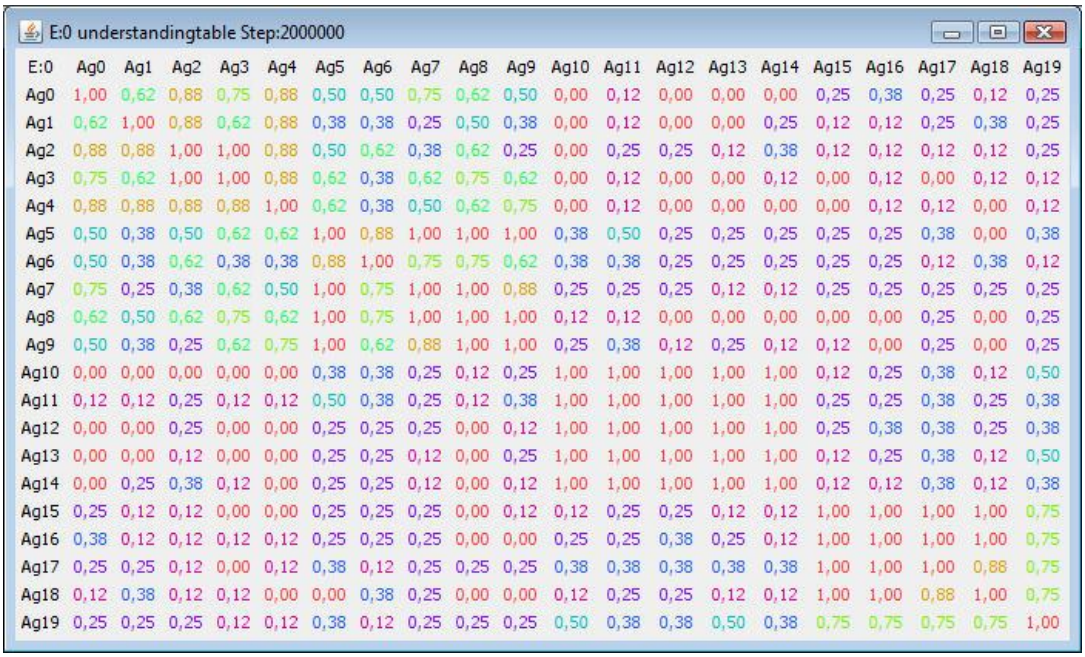
Abb. 92 Vergleich der Wortmuster

E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,11	0,13	0,10	0,11	0,35	0,34	0,33	0,37	0,37	0,54	0,57	0,56	0,55	0,56	0,56	0,56	0,53	0,51	0,54
Ag1	0,11	0,00	0,15	0,10	0,09	0,35	0,34	0,34	0,37	0,38	0,52	0,55	0,54	0,53	0,53	0,52	0,52	0,49	0,48	0,50
Ag2	0,13	0,15	0,00	0,16	0,15	0,39	0,40	0,38	0,41	0,42	0,49	0,52	0,51	0,49	0,51	0,52	0,52	0,49	0,48	0,50
Ag3	0,10	0,10	0,16	0,00	0,15	0,33	0,33	0,32	0,35	0,36	0,55	0,58	0,57	0,56	0,57	0,58	0,58	0,55	0,54	0,56
Ag4	0,11	0,09	0,15	0,15	0,00	0,38	0,37	0,36	0,39	0,39	0,52	0,55	0,53	0,53	0,53	0,53	0,53	0,50	0,48	0,51
Ag5	0,35	0,35	0,39	0,33	0,38	0,00	0,11	0,10	0,08	0,09	0,46	0,48	0,47	0,46	0,47	0,57	0,55	0,54	0,53	0,53
Ag6	0,34	0,34	0,40	0,33	0,37	0,11	0,00	0,09	0,13	0,13	0,47	0,49	0,48	0,47	0,48	0,55	0,54	0,53	0,51	0,53
Ag7	0,33	0,34	0,38	0,32	0,36	0,10	0,09	0,00	0,12	0,11	0,48	0,50	0,49	0,48	0,49	0,54	0,52	0,51	0,50	0,51
Ag8	0,37	0,37	0,41	0,35	0,39	0,08	0,13	0,12	0,00	0,08	0,45	0,47	0,45	0,45	0,46	0,56	0,54	0,53	0,52	0,52
Ag9	0,37	0,38	0,42	0,36	0,39	0,09	0,13	0,11	0,08	0,00	0,44	0,45	0,44	0,43	0,44	0,54	0,52	0,51	0,50	0,51
Ag10	0,54	0,52	0,49	0,55	0,52	0,46	0,47	0,48	0,45	0,44	0,00	0,09	0,10	0,08	0,11	0,46	0,45	0,45	0,45	0,42
Ag11	0,57	0,55	0,52	0,58	0,55	0,48	0,49	0,50	0,47	0,45	0,09	0,00	0,08	0,06	0,08	0,45	0,44	0,44	0,45	0,42
Ag12	0,56	0,54	0,51	0,57	0,53	0,47	0,48	0,49	0,45	0,44	0,10	0,08	0,00	0,09	0,10	0,46	0,45	0,44	0,45	0,42
Ag13	0,55	0,53	0,49	0,56	0,53	0,46	0,47	0,48	0,45	0,43	0,08	0,06	0,09	0,00	0,11	0,46	0,45	0,45	0,45	0,42
Ag14	0,56	0,53	0,51	0,57	0,53	0,47	0,48	0,49	0,46	0,44	0,11	0,08	0,10	0,11	0,00	0,45	0,44	0,44	0,45	0,42
Ag15	0,56	0,52	0,52	0,58	0,53	0,57	0,55	0,54	0,56	0,54	0,46	0,45	0,46	0,46	0,45	0,00	0,05	0,07	0,10	0,09
Ag16	0,56	0,52	0,52	0,58	0,53	0,55	0,54	0,52	0,54	0,52	0,45	0,44	0,45	0,45	0,44	0,05	0,00	0,06	0,10	0,07
Ag17	0,53	0,49	0,49	0,55	0,50	0,54	0,53	0,51	0,53	0,51	0,45	0,44	0,44	0,45	0,44	0,07	0,06	0,00	0,11	0,08
Ag18	0,51	0,48	0,48	0,54	0,48	0,53	0,51	0,50	0,52	0,50	0,45	0,45	0,45	0,45	0,45	0,10	0,10	0,11	0,00	0,13
Ag19	0,54	0,50	0,50	0,56	0,51	0,53	0,53	0,51	0,52	0,51	0,42	0,42	0,42	0,42	0,42	0,09	0,07	0,08	0,13	0,00

Abb. 93 Avg2-Tabelle in Simulationsschritt 2000000

Dies zeichnet sich auch in der Tabelle des gegenseitigen Verständnisses zum Ende des Simulationsdurchlaufs ab. Nicht immer ist eine gruppeninterne Kommunikation erfolgversprechend. Agent 0 und Agent 1 verstehen sich in diesem Fall sogar trotz gemeinsamer Gruppenzugehörigkeit über die komplette Simulationsdauer nur in 62% der Fälle. Dennoch zeichnen sich auch hier klare Grenzen in der Kreuztabelle der Agenten ab, die die Gruppeneinteilung widerspiegeln.



E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	0,62	0,88	0,75	0,88	0,50	0,50	0,75	0,62	0,50	0,00	0,12	0,00	0,00	0,00	0,25	0,38	0,25	0,12	0,25
Ag1	0,62	1,00	0,88	0,62	0,88	0,38	0,38	0,25	0,50	0,38	0,00	0,12	0,00	0,00	0,25	0,12	0,12	0,25	0,38	0,25
Ag2	0,88	0,88	1,00	1,00	0,88	0,50	0,62	0,38	0,62	0,25	0,00	0,25	0,25	0,12	0,38	0,12	0,12	0,12	0,12	0,25
Ag3	0,75	0,62	1,00	1,00	0,88	0,62	0,38	0,62	0,75	0,62	0,00	0,12	0,00	0,00	0,12	0,00	0,12	0,00	0,12	0,12
Ag4	0,88	0,88	0,88	0,88	1,00	0,62	0,38	0,50	0,62	0,75	0,00	0,12	0,00	0,00	0,00	0,00	0,12	0,12	0,00	0,12
Ag5	0,50	0,38	0,50	0,62	0,62	1,00	0,88	1,00	1,00	1,00	0,38	0,50	0,25	0,25	0,25	0,25	0,25	0,38	0,00	0,38
Ag6	0,50	0,38	0,62	0,38	0,38	0,88	1,00	0,75	0,75	0,62	0,38	0,38	0,25	0,25	0,25	0,25	0,25	0,12	0,38	0,12
Ag7	0,75	0,25	0,38	0,62	0,50	1,00	0,75	1,00	1,00	0,88	0,25	0,25	0,25	0,12	0,12	0,25	0,25	0,25	0,25	0,25
Ag8	0,62	0,50	0,62	0,75	0,62	1,00	0,75	1,00	1,00	1,00	0,12	0,12	0,00	0,00	0,00	0,00	0,00	0,25	0,00	0,25
Ag9	0,50	0,38	0,25	0,62	0,75	1,00	0,62	0,88	1,00	1,00	0,25	0,38	0,12	0,25	0,12	0,12	0,00	0,25	0,00	0,25
Ag10	0,00	0,00	0,00	0,00	0,00	0,38	0,38	0,25	0,12	0,25	1,00	1,00	1,00	1,00	1,00	0,12	0,25	0,38	0,12	0,50
Ag11	0,12	0,12	0,25	0,12	0,12	0,50	0,38	0,25	0,12	0,38	1,00	1,00	1,00	1,00	1,00	0,25	0,25	0,38	0,25	0,38
Ag12	0,00	0,00	0,25	0,00	0,00	0,25	0,25	0,25	0,00	0,12	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,38	0,25	0,38
Ag13	0,00	0,00	0,12	0,00	0,00	0,25	0,25	0,12	0,00	0,25	1,00	1,00	1,00	1,00	1,00	0,12	0,25	0,38	0,12	0,50
Ag14	0,00	0,25	0,38	0,12	0,00	0,25	0,25	0,12	0,00	0,12	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,38	0,12	0,38
Ag15	0,25	0,12	0,12	0,00	0,00	0,25	0,25	0,25	0,00	0,12	0,12	0,25	0,25	0,12	0,12	1,00	1,00	1,00	1,00	0,75
Ag16	0,38	0,12	0,12	0,12	0,12	0,25	0,25	0,25	0,00	0,00	0,25	0,25	0,38	0,25	0,12	1,00	1,00	1,00	1,00	0,75
Ag17	0,25	0,25	0,12	0,00	0,12	0,38	0,12	0,25	0,25	0,25	0,38	0,38	0,38	0,38	0,38	1,00	1,00	1,00	0,88	0,75
Ag18	0,12	0,38	0,12	0,12	0,00	0,00	0,38	0,25	0,00	0,00	0,12	0,25	0,25	0,12	0,12	1,00	1,00	0,88	1,00	0,75
Ag19	0,25	0,25	0,25	0,12	0,12	0,38	0,12	0,25	0,25	0,25	0,50	0,38	0,38	0,50	0,38	0,75	0,75	0,75	0,75	1,00

Abb. 94 Understanding-Tabelle in Simulationsschritt 2000000

4.6 Simulation 6: Wortkomprimierung durch die DCT

In diesem Unterkapitel wird die Möglichkeit einer Dialektbildung mit Hilfe der Wortkomprimierung durch die diskrete Kosinustransformation untersucht. Durch den nachfolgenden verlustbehafteten Einsatz einer Quantisierung werden Faktoren, die nur einen schwachen Einfluss auf das Wortbild haben, herausgefiltert. Wirkt dieser Störfaktor dabei wie ein Rauschen und führt eine Ausbildung von Dialekten herbei oder wirkt er stabilisierend und verhindert eine Sprachveränderung?

4.6.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2

  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen

- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge *S*
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Es existiert keine Sterbewahrscheinlichkeit für die Agenten
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Der Schwellenwert nach einer diskreten Kosinustransformation mit anschließender Quantisierung hat eine Stärke von 0.07.

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.6.2 Analyse

Wie bereits bei einigen Störfaktoren vorher bildet das Werteintervall der diskreten Kosinustransformation ebenfalls nur zwei Intervalle mit verschiedenartigen Auswirkungen aus. Ein Schwellenwert im Intervall [0.0;0.9] bei der Quantisierung ermöglicht es der Agentengemeinschaft ein Lexikon auszubilden, das auch nach der Aufspaltung in Agentengruppen nicht in verschiedene Dialekte zerfällt. Als Beispiel dazu wird im Folgenden ein Simulationsdurchlauf mit einem Schwellenwert von 0.07 erläutert. Bei größeren Schwellenwerten sind die Agenten

nicht mehr in der Lage, ein gemeinsames Lexikon auszubilden. Da solche Fälle bereits bei vorherigen Simulationsdurchläufen gezeigt wurden, wird an dieser Stelle darauf verzichtet.

Quantisierungs-Schwellenwert 0.07

Als erster Indikator für einen geglückten Simulationsdurchlauf wird der Indikator Avg1 betrachtet. Dieser steigt sehr schnell auf einen Wert von 0.45 an und konvergiert im Laufe der Simulation gegen den Wert 0.5. Damit ist der Unterschied zwischen den verschiedenen Wortmustern groß genug, eine Unterscheidung ist für die Agenten möglich.

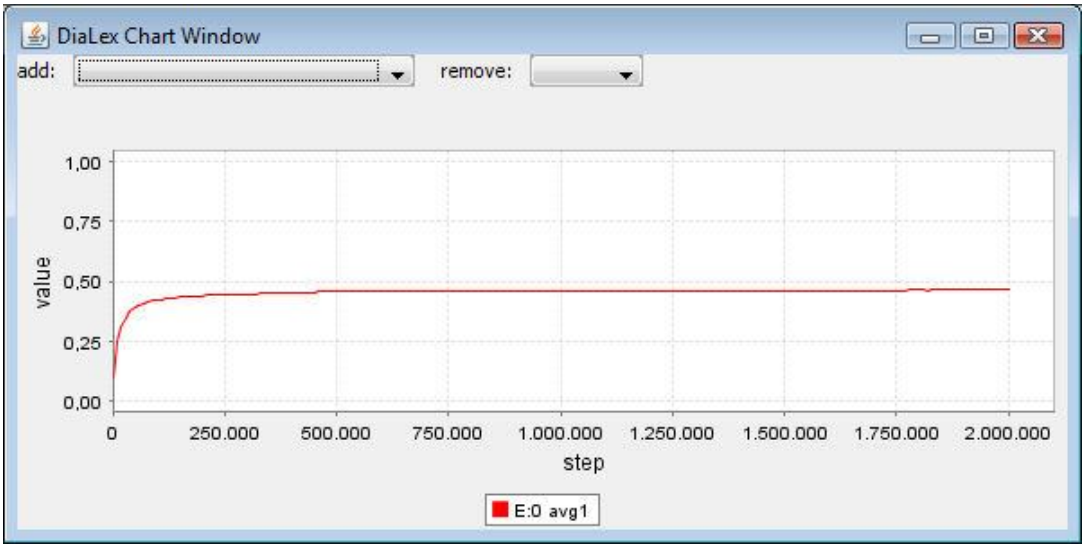


Abb. 95 Graph der Avg1-Werte

Der Graph des Avg2 zeigt ein paralleles, nahezu identisches Absinken des gemeinschaftlichen Avg2-Werts und der Avg2-Werte aller einzelnen Gruppen. Dies beweist, dass absolut keine Gruppenausbildung bezüglich des Wortschatzes stattgefunden hat. Die Wortmuster innerhalb einer Gruppe unterscheiden sich nicht weniger stark von denen anderer Gruppen.

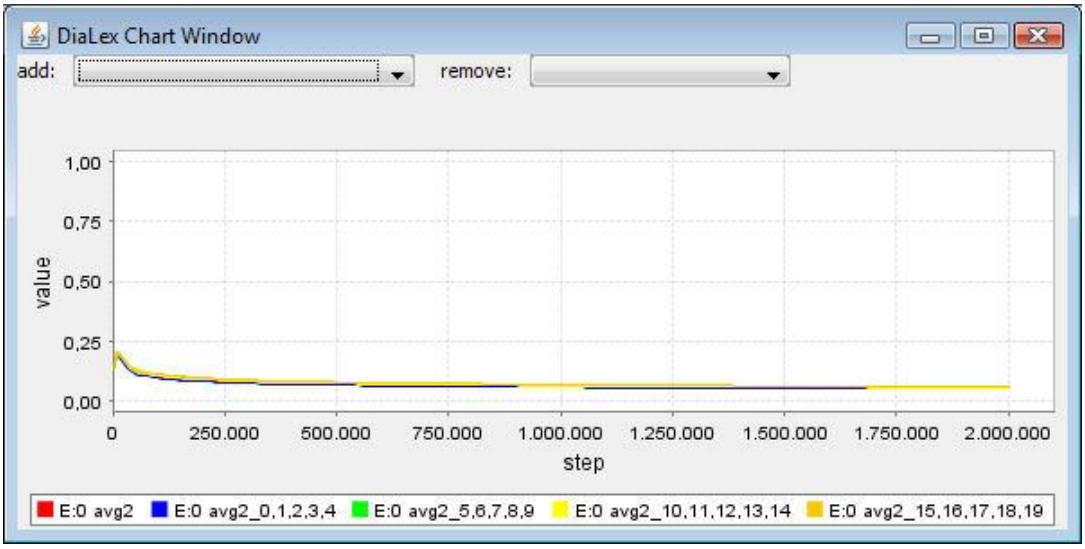


Abb. 96 Graph der Avg2-Werte

Der Understanding-Graph zeigt diese Beobachtung zwar etwas genauer, lässt aber auf die gleichen Folgerungen schließen. Alle Graphen der gegenseitigen Verständlichkeit verlaufen parallel auf hohem Niveau. Eine Dialektbildung in diesem Simulationsdurchlauf ist damit definitiv ausgeschlossen.

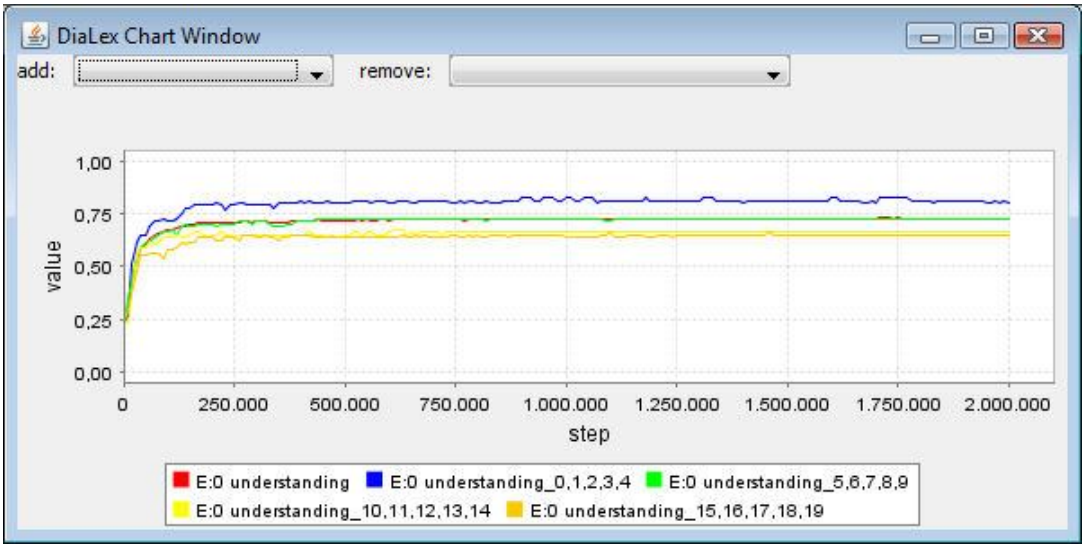
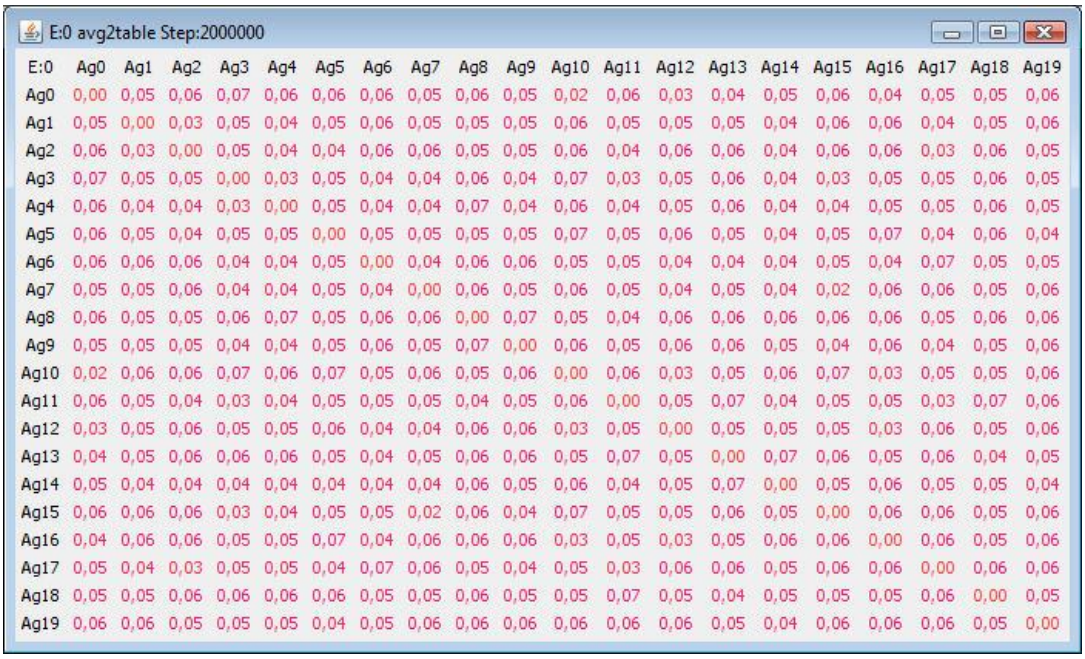


Abb. 97 Graph der Understanding-Werte

Zur letzten Kontrolle werden die Avg2- und Understanding-Tabellen zum Ende des Simulationsdurchlaufs herangezogen. Abbildung 98 zeigt das nach Betrachtung des Avg2-Graphen erwartete Bild: Die Wörter aller Agenten sind sich sehr ähnlich, wenngleich jedoch verschiedener, als sie es ohne einen Störfaktor

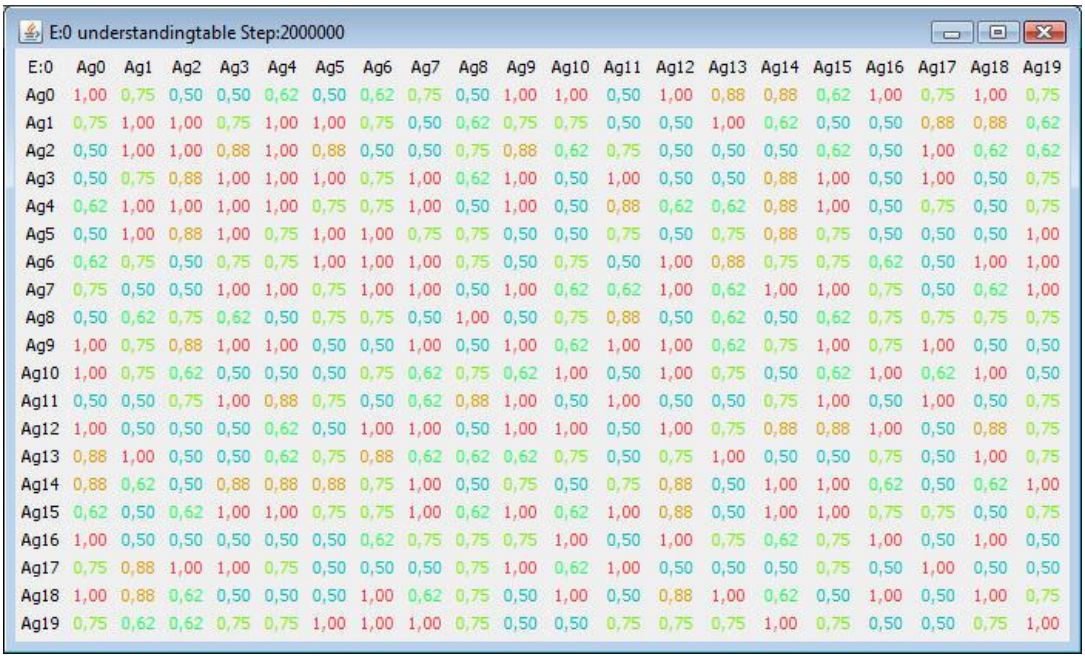
wären. (vgl. Unterkapitel 4.1) Trotz der sehr ähnlichen Wortmuster erreichen die Agenten keinesfalls eine hohe gegenseitige Verständlichkeit. Vielmehr variieren die Werte gruppenintern wie extern in einem Bereich von 0.5 bis 1.0. Gruppenausprägungen sind dabei nicht zu erkennen.



E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,05	0,06	0,07	0,06	0,06	0,06	0,05	0,06	0,05	0,02	0,06	0,03	0,04	0,05	0,06	0,04	0,05	0,05	0,06
Ag1	0,05	0,00	0,03	0,05	0,04	0,05	0,06	0,05	0,05	0,05	0,06	0,05	0,05	0,05	0,04	0,06	0,06	0,04	0,05	0,06
Ag2	0,06	0,03	0,00	0,05	0,04	0,04	0,06	0,06	0,05	0,05	0,06	0,04	0,06	0,06	0,04	0,06	0,06	0,03	0,06	0,05
Ag3	0,07	0,05	0,05	0,00	0,03	0,05	0,04	0,04	0,06	0,04	0,07	0,03	0,05	0,06	0,04	0,03	0,05	0,05	0,06	0,05
Ag4	0,06	0,04	0,04	0,03	0,00	0,05	0,04	0,04	0,07	0,04	0,06	0,04	0,05	0,06	0,04	0,04	0,05	0,05	0,06	0,05
Ag5	0,06	0,05	0,04	0,05	0,05	0,00	0,05	0,05	0,05	0,05	0,07	0,05	0,06	0,05	0,04	0,05	0,07	0,04	0,06	0,04
Ag6	0,06	0,06	0,06	0,04	0,04	0,05	0,00	0,04	0,06	0,06	0,05	0,05	0,04	0,04	0,04	0,05	0,04	0,07	0,05	0,05
Ag7	0,05	0,05	0,06	0,04	0,04	0,05	0,04	0,00	0,06	0,05	0,06	0,05	0,04	0,05	0,04	0,02	0,06	0,06	0,05	0,06
Ag8	0,06	0,05	0,05	0,06	0,07	0,05	0,06	0,06	0,00	0,07	0,05	0,04	0,06	0,06	0,06	0,06	0,06	0,05	0,06	0,06
Ag9	0,05	0,05	0,05	0,04	0,04	0,05	0,06	0,05	0,07	0,00	0,06	0,05	0,06	0,06	0,05	0,04	0,06	0,04	0,05	0,06
Ag10	0,02	0,06	0,06	0,07	0,06	0,07	0,05	0,06	0,05	0,06	0,00	0,06	0,03	0,05	0,06	0,07	0,03	0,05	0,05	0,06
Ag11	0,06	0,05	0,04	0,03	0,04	0,05	0,05	0,05	0,04	0,05	0,06	0,00	0,05	0,07	0,04	0,05	0,05	0,03	0,07	0,06
Ag12	0,03	0,05	0,06	0,05	0,05	0,06	0,04	0,04	0,06	0,06	0,03	0,05	0,00	0,05	0,05	0,05	0,03	0,06	0,05	0,06
Ag13	0,04	0,05	0,06	0,06	0,06	0,05	0,04	0,05	0,06	0,06	0,05	0,07	0,05	0,00	0,07	0,06	0,05	0,06	0,04	0,05
Ag14	0,05	0,04	0,04	0,04	0,04	0,04	0,04	0,04	0,06	0,05	0,06	0,04	0,05	0,07	0,00	0,05	0,06	0,05	0,05	0,04
Ag15	0,06	0,06	0,06	0,03	0,04	0,05	0,05	0,02	0,06	0,04	0,07	0,05	0,05	0,06	0,05	0,00	0,06	0,06	0,05	0,06
Ag16	0,04	0,06	0,06	0,05	0,05	0,07	0,04	0,06	0,06	0,06	0,03	0,05	0,03	0,05	0,06	0,06	0,00	0,06	0,05	0,06
Ag17	0,05	0,04	0,03	0,05	0,05	0,04	0,07	0,06	0,05	0,04	0,05	0,03	0,06	0,06	0,05	0,06	0,06	0,00	0,06	0,06
Ag18	0,05	0,05	0,06	0,06	0,06	0,06	0,05	0,05	0,06	0,05	0,05	0,07	0,05	0,04	0,05	0,05	0,05	0,06	0,00	0,05
Ag19	0,06	0,06	0,05	0,05	0,05	0,04	0,05	0,06	0,06	0,06	0,06	0,06	0,06	0,05	0,04	0,06	0,06	0,06	0,05	0,00

Abb. 98 Avg2-Tabelle in Simulationsschritt 2000000



E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	0,75	0,50	0,50	0,62	0,50	0,62	0,75	0,50	1,00	1,00	0,50	1,00	0,88	0,88	0,62	1,00	0,75	1,00	0,75
Ag1	0,75	1,00	1,00	0,75	1,00	1,00	0,75	0,50	0,62	0,75	0,75	0,50	0,50	1,00	0,62	0,50	0,50	0,88	0,88	0,62
Ag2	0,50	1,00	1,00	0,88	1,00	0,88	0,50	0,50	0,75	0,88	0,62	0,75	0,50	0,50	0,50	0,62	0,50	1,00	0,62	0,62
Ag3	0,50	0,75	0,88	1,00	1,00	1,00	0,75	1,00	0,62	1,00	0,50	1,00	0,50	0,50	0,88	1,00	0,50	1,00	0,50	0,75
Ag4	0,62	1,00	1,00	1,00	1,00	0,75	0,75	1,00	0,50	1,00	0,50	0,88	0,62	0,62	0,88	1,00	0,50	0,75	0,50	0,75
Ag5	0,50	1,00	0,88	1,00	0,75	1,00	1,00	0,75	0,75	0,50	0,50	0,75	0,50	0,75	0,88	0,75	0,50	0,50	0,50	1,00
Ag6	0,62	0,75	0,50	0,75	0,75	1,00	1,00	1,00	0,75	0,50	0,75	0,50	1,00	0,88	0,75	0,75	0,62	0,50	1,00	1,00
Ag7	0,75	0,50	0,50	1,00	1,00	0,75	1,00	1,00	0,50	1,00	0,62	0,62	1,00	0,62	1,00	1,00	0,75	0,50	0,62	1,00
Ag8	0,50	0,62	0,75	0,62	0,50	0,75	0,75	0,50	1,00	0,50	0,75	0,88	0,50	0,62	0,50	0,62	0,75	0,75	0,75	0,75
Ag9	1,00	0,75	0,88	1,00	1,00	0,50	0,50	1,00	0,50	1,00	0,62	1,00	1,00	0,62	0,75	1,00	0,75	1,00	0,50	0,50
Ag10	1,00	0,75	0,62	0,50	0,50	0,50	0,75	0,62	0,75	0,62	1,00	0,50	1,00	0,75	0,50	0,62	1,00	0,62	1,00	0,50
Ag11	0,50	0,50	0,75	1,00	0,88	0,75	0,50	0,62	0,88	1,00	0,50	1,00	0,50	0,50	0,75	1,00	0,50	1,00	0,50	0,75
Ag12	1,00	0,50	0,50	0,50	0,62	0,50	1,00	1,00	0,50	1,00	1,00	0,50	1,00	0,75	0,88	0,88	1,00	0,50	0,88	0,75
Ag13	0,88	1,00	0,50	0,50	0,62	0,75	0,88	0,62	0,62	0,62	0,75	0,50	0,75	1,00	0,50	0,50	0,75	0,50	1,00	0,75
Ag14	0,88	0,62	0,50	0,88	0,88	0,88	0,75	1,00	0,50	0,75	0,50	0,75	0,88	0,50	1,00	1,00	0,62	0,50	0,62	1,00
Ag15	0,62	0,50	0,62	1,00	1,00	0,75	0,75	1,00	0,62	1,00	0,62	1,00	0,88	0,50	1,00	1,00	0,75	0,75	0,50	0,75
Ag16	1,00	0,50	0,50	0,50	0,50	0,50	0,62	0,75	0,75	0,75	1,00	0,50	1,00	0,75	0,62	0,75	1,00	0,50	1,00	0,50
Ag17	0,75	0,88	1,00	1,00	0,75	0,50	0,50	0,50	0,75	1,00	0,62	1,00	0,50	0,50	0,50	0,75	0,50	1,00	0,50	0,50
Ag18	1,00	0,88	0,62	0,50	0,50	0,50	1,00	0,62	0,75	0,50	1,00	0,50	0,88	1,00	0,62	0,50	1,00	0,50	1,00	0,75
Ag19	0,75	0,62	0,62	0,75	0,75	1,00	1,00	1,00	0,75	0,50	0,50	0,75	0,75	0,75	1,00	0,75	0,50	0,50	0,75	1,00

Abb. 99 Understanding-Tabelle in Simulationsschritt 2000000

4.7 Simulation 7: Wechselwirkungen

Zusätzlich zu der Tatsache, ob Störfaktoren eine Sprachspaltung ermöglichen, stellt sich die Frage, ob Wechselwirkungen zwischen verschiedenen Störfaktoren existieren. Dass hierbei ein „je mehr desto mehr“ Effekt bei einer Kombination zweier Faktoren, die zu einer Dialektbildung führen, existiert, ist zu erwarten und bedarf daher keiner separaten Aufführung eines Simulationsdurchlaufs. Vielmehr wird ein Beispiel hierzu in der nachfolgenden Simulation (siehe Unterkapitel 4.8) integriert. Entscheidender ist vielmehr die Frage, ob ein Faktor, der nicht zu einer Dialektbildung geführt hat, einen anderen dialektbildenden Faktor neutralisieren kann. Da die Auswahl an Störfaktoren, die keine Sprachspaltung auslösen, sehr begrenzt ist und ein Rauschen während der Kommunikation zu sehr schlechten Avg1-Werten führt, drängt sich die diskrete Kosinustransformation als eventueller „Neutralisierer“ eines anderen Störfaktors auf. Im folgenden Beispiel wird daher untersucht, was eine Kombination aus einer diskreten Kosinusanalyse mit einer Sterbewahrscheinlichkeit von Agenten, die alleine eine Dialektbildung bewirkt, hervorruft.

4.7.1 Simulationsparameter

Visuelle Szenen:

- m = 4
- S = { ((1,0),(0,0)),
  ((0,1),(0,0)),
  ((0,0),(1,0)),
  ((0,0),(0,1)) }
- F = {1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 2x2, 3x2, 2x2

  Ein- und Ausgabeschicht mit 4 Neuronen, eine verborgene Schicht mit 6 Neuronen

- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1
- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge *S*
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Die Sterbewahrscheinlichkeit für die Agenten beträgt 0.00001
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Die diskrete Kosinustransformation hat eine Stärke von 0.07

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechn. Wahrscheinlichkeiten des Manhattan-Distanzmaßes) }

4.7.2 Analyse

Aus Gründen der einfachen Vergleichbarkeit wird eine Kombination der diskreten Kosinustransformation, in der Stärke der vorhergegangen Simulation in Unterkapitel 4.6, mit einer Agentensterblichkeit, die in der Stärke in Unterkapitel 4.4 gezeigt wurde, als Beispiel eines Simulationsdurchlaufs gewählt.

Sterbewahrscheinlichkeit 0.00001 & Quantisierungs-Schwellenwert 0.07

Wie in Abbildung 100 ersichtlich, steigt der Indikator Avg1 nach Beginn des Simulationsdurchlaufs stetig auf einen Wert größer als 0.5 an und verbleibt permanent in einem Intervall zwischen 0.5 und 0.6. Damit ist eine Unterscheidung der Wortmuster definitiv gewährleistet, der Durchlauf somit erfolgreich.

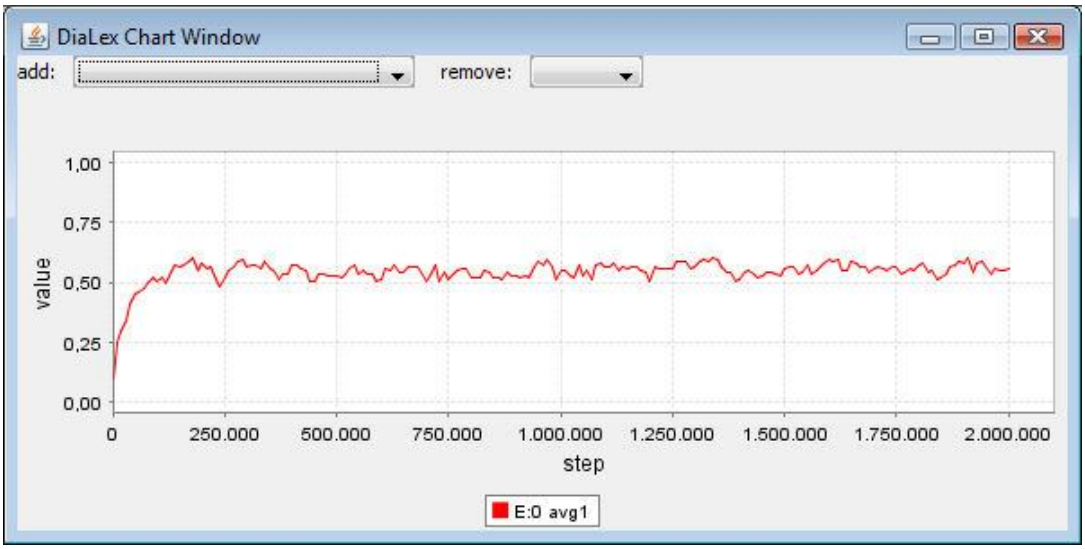


Abb. 100 Graph der Avg1-Werte

Ein Blick auf den Indikator Avg2 lässt vermuten, dass keine Dialektbildung stattgefunden hat. Der Kurvenverlauf des Avg2 der gesamten Population unterscheidet sich kaum von denen der einzelnen Untergruppen, die Wortdifferenzen sind allgemein sehr niedrig. Einzelne Ausreißer nach oben deuten vielmehr auf den Einfluss ungelernter Agenten in die Fehlerberechnung ein, die in der Folge jedoch das Lexikon erlernen.

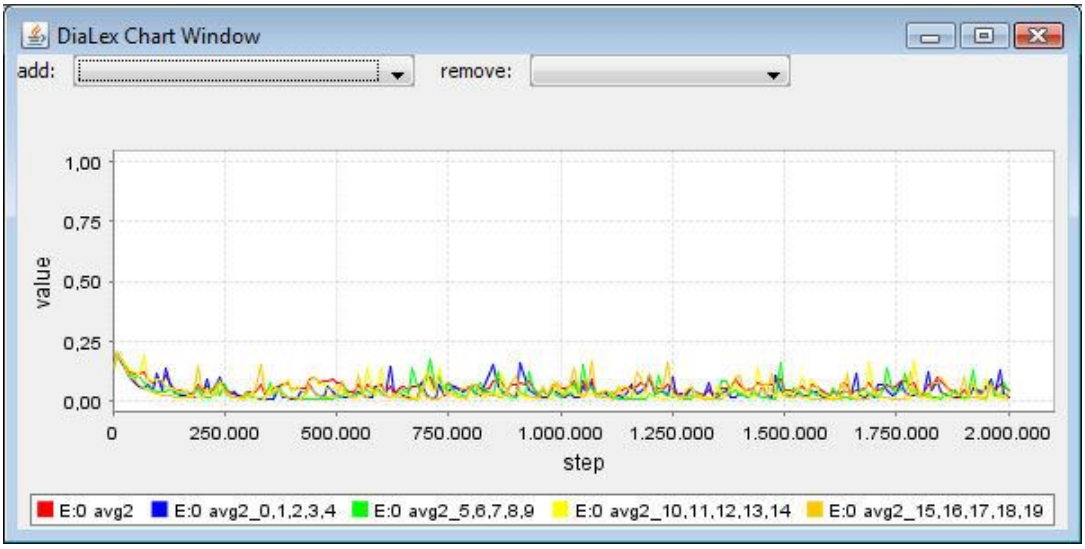


Abb. 101 Graph der Avg2-Werte

Der Understanding-Graph verdeutlicht dieses Bild. Auch hier verlaufen die Kurven ähnlich, sie liegen nahezu alle permanent bei einem Optimalwert von 1.0. Ausreißer nach unten können wieder durch ungelernte Agenten erklärt werden. Eine Dialektbildung ist jedoch auch hier nicht zu erkennen.

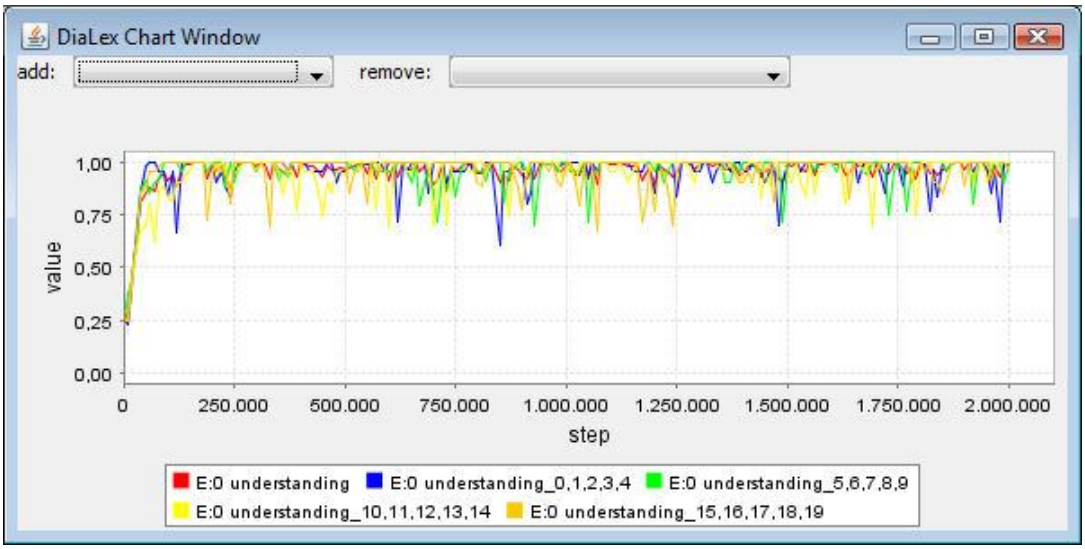
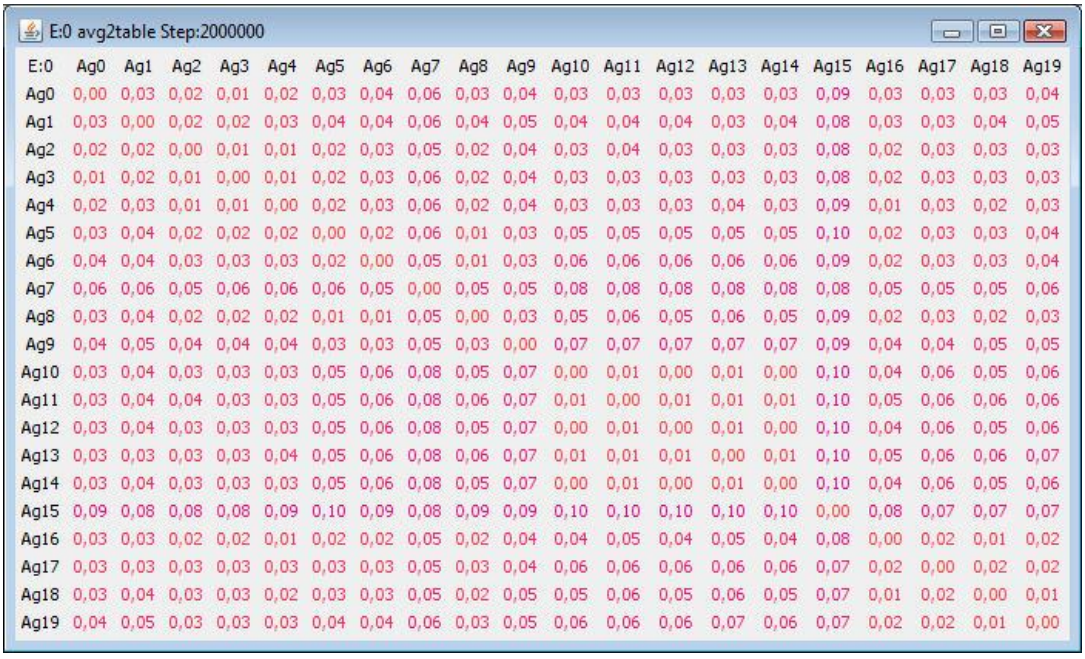


Abb. 102 Graph der Understanding-Werte

Letzte Gewissheit bringen die Avg2- und Understanding-Tabellen zum Ende des Simulationsdurchlaufs in Simulationsschritt 2000000. Gruppeneinteilungen sind wenn überhaupt minimal zu erkennen, die Wortrepräsentationen aller Agenten unterscheiden sich maximal um einen Wert von 0.1 bei Agent 15. Die Tabelle der gegenseitigen Verständlichkeit zeigt ein sowohl gruppeninternes als auch

gruppenexternes sehr hohes Maß an Verständlichkeit. Alle Agenten mit Ausnahme von Agent 15 können sich selbst nach dieser langen Gruppenunterteilung fehlerfrei verständigen. Ein Blick in die Agentendetails erklärt diesen Ausreißer. Agent 15 hat in der Tat noch nicht bei vielen Kommunikationsversuchen teilgenommen und ist somit weniger trainiert als die anderen Agenten.



E:0 avg2table Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,03	0,02	0,01	0,02	0,03	0,04	0,06	0,03	0,04	0,03	0,03	0,03	0,03	0,03	0,09	0,03	0,03	0,03	0,04
Ag1	0,03	0,00	0,02	0,02	0,03	0,04	0,04	0,06	0,04	0,05	0,04	0,04	0,04	0,03	0,04	0,08	0,03	0,03	0,04	0,05
Ag2	0,02	0,02	0,00	0,01	0,01	0,02	0,03	0,05	0,02	0,04	0,03	0,04	0,03	0,03	0,03	0,08	0,02	0,03	0,03	0,03
Ag3	0,01	0,02	0,01	0,00	0,01	0,02	0,03	0,06	0,02	0,04	0,03	0,03	0,03	0,03	0,03	0,08	0,02	0,03	0,03	0,03
Ag4	0,02	0,03	0,01	0,01	0,00	0,02	0,03	0,06	0,02	0,04	0,03	0,03	0,03	0,04	0,03	0,09	0,01	0,03	0,02	0,03
Ag5	0,03	0,04	0,02	0,02	0,02	0,00	0,02	0,06	0,01	0,03	0,05	0,05	0,05	0,05	0,05	0,10	0,02	0,03	0,03	0,04
Ag6	0,04	0,04	0,03	0,03	0,03	0,02	0,00	0,05	0,01	0,03	0,06	0,06	0,06	0,06	0,06	0,09	0,02	0,03	0,03	0,04
Ag7	0,06	0,06	0,05	0,06	0,06	0,06	0,05	0,00	0,05	0,05	0,08	0,08	0,08	0,08	0,08	0,08	0,05	0,05	0,05	0,06
Ag8	0,03	0,04	0,02	0,02	0,02	0,01	0,01	0,05	0,00	0,03	0,05	0,06	0,05	0,06	0,05	0,09	0,02	0,03	0,02	0,03
Ag9	0,04	0,05	0,04	0,04	0,04	0,03	0,03	0,05	0,03	0,00	0,07	0,07	0,07	0,07	0,07	0,09	0,04	0,04	0,05	0,05
Ag10	0,03	0,04	0,03	0,03	0,03	0,05	0,06	0,08	0,05	0,07	0,00	0,01	0,00	0,01	0,00	0,10	0,04	0,06	0,05	0,06
Ag11	0,03	0,04	0,04	0,03	0,03	0,05	0,06	0,08	0,06	0,07	0,01	0,00	0,01	0,01	0,01	0,10	0,05	0,06	0,06	0,06
Ag12	0,03	0,04	0,03	0,03	0,03	0,05	0,06	0,08	0,05	0,07	0,00	0,01	0,00	0,01	0,00	0,10	0,04	0,06	0,05	0,06
Ag13	0,03	0,03	0,03	0,03	0,04	0,05	0,06	0,08	0,06	0,07	0,01	0,01	0,01	0,00	0,01	0,10	0,05	0,06	0,06	0,07
Ag14	0,03	0,04	0,03	0,03	0,03	0,05	0,06	0,08	0,05	0,07	0,00	0,01	0,00	0,01	0,00	0,10	0,04	0,06	0,05	0,06
Ag15	0,09	0,08	0,08	0,08	0,09	0,10	0,09	0,08	0,09	0,09	0,10	0,10	0,10	0,10	0,10	0,00	0,08	0,07	0,07	0,07
Ag16	0,03	0,03	0,02	0,02	0,01	0,02	0,02	0,05	0,02	0,04	0,04	0,05	0,04	0,05	0,04	0,08	0,00	0,02	0,01	0,02
Ag17	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,05	0,03	0,04	0,06	0,06	0,06	0,06	0,06	0,07	0,02	0,00	0,02	0,02
Ag18	0,03	0,04	0,03	0,03	0,02	0,03	0,03	0,05	0,02	0,05	0,05	0,06	0,05	0,06	0,05	0,07	0,01	0,02	0,00	0,01
Ag19	0,04	0,05	0,03	0,03	0,03	0,04	0,04	0,06	0,03	0,05	0,06	0,06	0,06	0,07	0,06	0,07	0,02	0,02	0,01	0,00

Abb. 103 Avg2-Tabelle in Simulationsschritt 2000000

E:0 understandingtable Step:2000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag1	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag2	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag3	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag4	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag6	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag7	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag8	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag9	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag10	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag11	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag12	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag13	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag14	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag15	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	1,00	0,88	0,88	0,88	0,88
Ag16	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag17	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag18	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00
Ag19	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,88	1,00	1,00	1,00	1,00

Abb. 104 Understanding-Tabelle in Simulationsschritt 2000000

Ein Blick auf die verbalen Repräsentationen von Agenten verschiedener Gruppen über den gesamten Simulationsverlauf bestätigt das Bild eines globalen gegenseitigen Verständnisses. Die Wortmuster der Agenten werden sich im Laufe der Simulation offensichtlich gar immer ähnlicher.

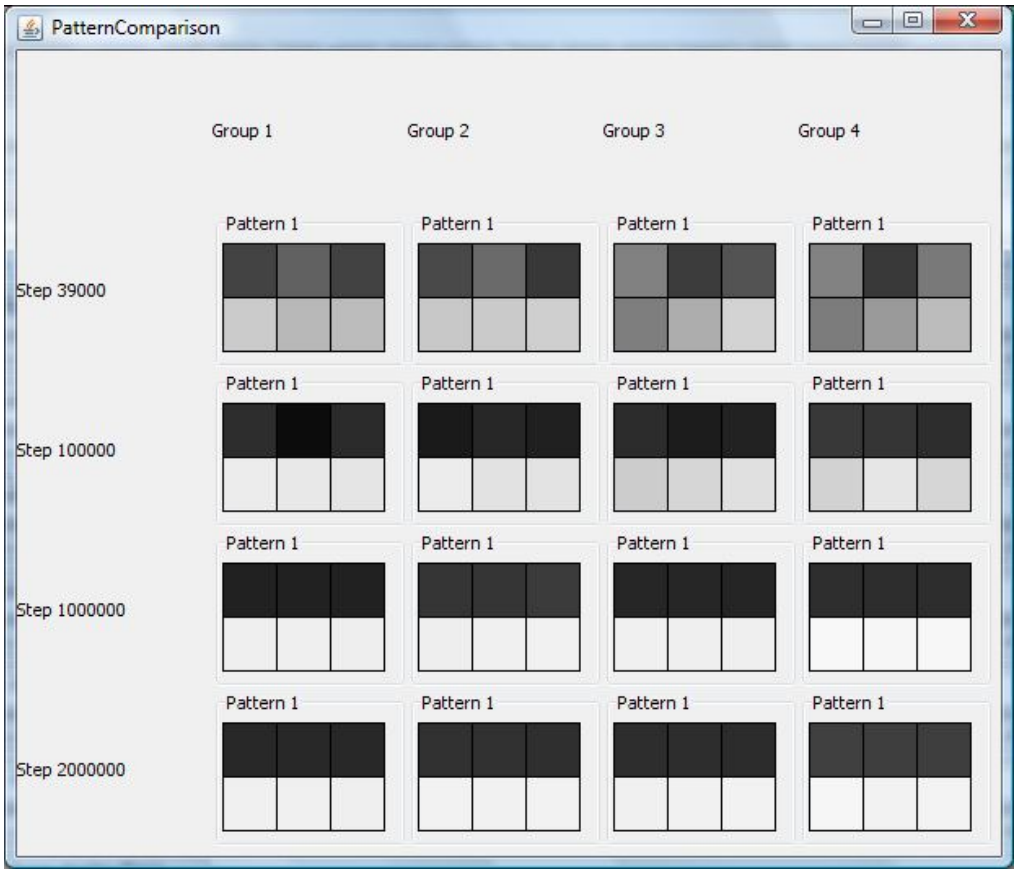


Abb. 105 Vergleich der Wortmuster

Zusammenfassend lässt sich sagen, dass der Einsatz der diskreten Kosinustransformation eine stabilisierende Wirkung auf Simulationsdurchläufe haben kann. Während eine Sterbewahrscheinlichkeit der gleichen Stärke ohne sie stets zu einer Dialektspaltung führt, bleibt diese in diesem Simulationsdurchlauf aus. Weitere Versuche ergaben, dass sie auch bei den anderen Störfaktoren die gleiche Wirkung erzeugt. Aufgrund der Ähnlichkeit der Beobachtungen werden diese jedoch an dieser Stelle nicht mehr aufgeführt.

Der zweite Störfaktor, das Rauschen während der Kommunikation, das ebenfalls keine Dialektbildung bewirkt hat, ermöglicht ebenfalls eine Unterdrückung einer Sprachspaltung. In diesem Falle sind die erzielten Ergebnisse aufgrund eines starken Rauschens jedoch weniger eindeutig und schlechter erkennbar. Aus diesem Grund wird auf eine genaue Betrachtung in dieser Dissertationsschrift verzichtet.

4.8 Simulation 8: Sprachspaltung und Wiedervereinigung

In diesem Unterkapitel werden die Ergebnisse einer Simulation mit gelungener Sprachspaltung und nachfolgender Wiedervereinigung der Agentenuntergruppen bei einer größeren Menge von Eingabemustern, die in Abbildung 106 dargestellt werden, benutzt. Diese Eingabemuster entsprechen den von Hutchins und Hazlehurst in [HH95] benutzten Mondphasen. Mit dieser Simulation sollen zwei weitere Aspekte dieser Simulationsreihe gezeigt werden.

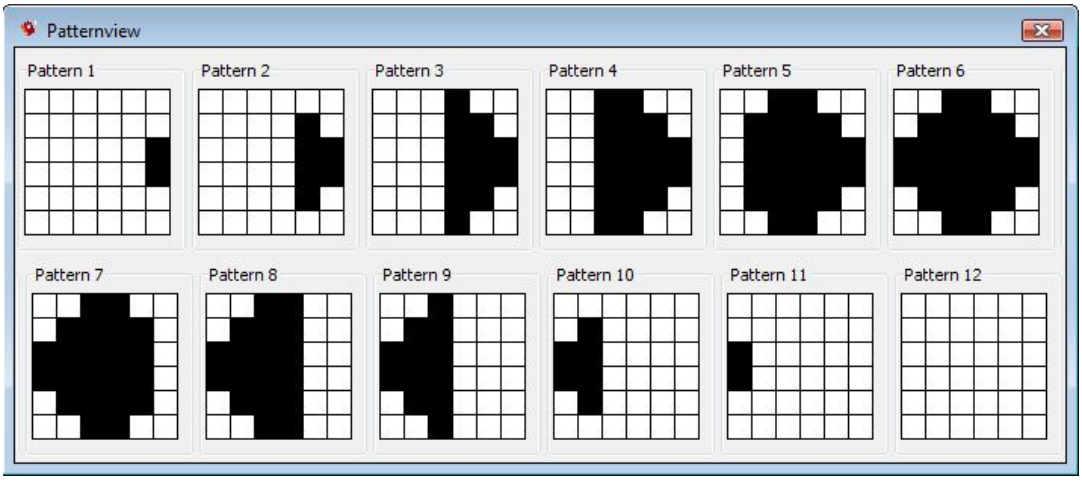


Abb. 106 Die Eingabemuster der Mondphasen

Zunächst zeigt dieser Simulationsdurchlauf, dass die bisher gezeigten Effekte ebenso in Simulationsumgebungen mit einer größeren Anzahl von Eingabemustern funktionieren und keinesfalls auf simple Beispiele begrenzt sind. Außerdem wird gezeigt, was geschieht, wenn eine, in Untergruppen aufgeteilte Agentengemeinschaft mit jeweiligen Dialekten wiedervereinigt wird. Als Störfaktor dient in diesem Simulationsdurchlauf eine Sterbewahrscheinlichkeit der Agenten.

4.8.1 Simulationsparameter

Visuelle Szenen:

- m = 12
- S = {((0,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,1),(0,0,0,0,0,1),(0,0,0,0,0,0),(0,0,0,0,0,0)),
  ((0,0,0,0,0,0),(0,0,0,0,1,0),(0,0,0,0,1,1),(0,0,0,0,1,1),(0,0,0,0,1,0),(0,0,0,0,0,0)),
  ((0,0,0,1,0,0),(0,0,0,1,1,0),(0,0,0,1,1,1),(0,0,0,1,1,1),(0,0,0,1,1,0),(0,0,0,1,0,0)),
  ((0,0,1,1,0,0),(0,0,1,1,1,0),(0,0,1,1,1,1),(0,0,1,1,1,1),(0,0,1,1,1,0),(0,0,1,1,0,0)),
  ((0,0,1,1,0,0),(0,1,1,1,1,0),(0,1,1,1,1,1),(0,1,1,1,1,1),(0,1,1,1,1,0),(0,0,1,1,0,0)),
  ((0,0,1,1,0,0),(0,1,1,1,1,0),(1,1,1,1,1,1),(1,1,1,1,1,1),(0,1,1,1,1,0),(0,0,1,1,0,0)),
  ((0,0,1,1,0,0),(0,1,1,1,1,0),(1,1,1,1,1,0),(1,1,1,1,1,0),(0,1,1,1,1,0),(0,0,1,1,0,0)),
  ((0,0,1,1,0,0),(0,1,1,1,0,0),(1,1,1,1,0,0),(1,1,1,1,0,0),(0,1,1,1,0,0),(0,0,1,1,0,0)),
  ((0,0,1,0,0,0),(0,1,1,0,0,0),(1,1,1,0,0,0),(1,1,1,0,0,0),(0,1,1,0,0,0),(0,0,1,0,0,0)),
  ((0,0,0,0,0,0),(0,1,0,0,0,0),(1,1,0,0,0,0),(1,1,0,0,0,0),(0,1,0,0,0,0),(0,0,0,0,0,0)),
  ((0,0,0,0,0,0),(0,0,0,0,0,0),(1,0,0,0,0,0),(1,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,0)),
  ((0,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,0),(0,0,0,0,0,0))}.
- F = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0}

Simulationsumgebung:

- n = 20
- D = {1000, 1000}
- f_{pos} = Die Agenten werden durch feste Koordinaten positioniert
- C = { (0,0), (0,0), (0,0), (0,0), (0,0),
  (999,0), (999,0), (999,0), (999,0), (999,0),
  (0,999), (0,999), (0,999), (0,999), (0,999),
  (999,999), (999,999), (999,999), (999,999), (999,999) }

Agent:

- f_{arch} = 6x6, 3x3, 3x3, 6x6
  Ein- und Ausgabeschicht mit 36 Neuronen, zwei verborgene Schichten mit 9 Neuronen
- W = Menge von zufälligen reellen Zahlen zwischen -0.5 und +0.5
- w = 1

- μ = 0.75
- ψ = 0.9

Kommunikation:

- f_{scene} = zufällige Auswahl einer visuellen Szene aus der Menge *S*
- f_{sp} = zufällige Auswahl eines Sprechers aus der Gemeinschaft der Agenten
- f_{li} = zufällige Auswahl eines Zuhörers aus der Gemeinschaft der Agenten
- f_{com} = ein Agent fungiert als Sprecher, ein anderer als Zuhörer
- ς = 5

Störfaktoren:

- f_{death} = Ein Agent wird in jedem Schritt mit der Wahrscheinlichkeit 0.00001 durch einen neu erzeugten, untrainierten Agenten ersetzt
- f_{dec} = Es existiert keine Abschwächung der Verbindungsgewichtungen
- f_{nip} = Es existiert kein Rauschen innerhalb der visuellen Szenen
- f_{ncom} = Es existiert kein Rauschen innerhalb der Kommunikation
- f_{dct} = Eine diskrete Kosinustransformation wird nicht benutzt

Ereignisse:

- E = { (wenn Understanding > 0.75, f_{li} = zufällige Zuhörerauswahl mit errechneten Wahrscheinlichkeiten des Manhattan-Distanzmaßes),
  (wenn Simulationsschritt = 8000000, f_{li} = zufällige Zuhörerauswahl) }

4.8.2 Analyse

Zu Beginn der Untersuchung wird anhand eines Graphen gezeigt, dass der Fehlerindikator Avg1 stets auf einem Niveau bleibt, das eine Unterscheidung zwischen einzelnen Wortrepräsentationen ermöglicht.

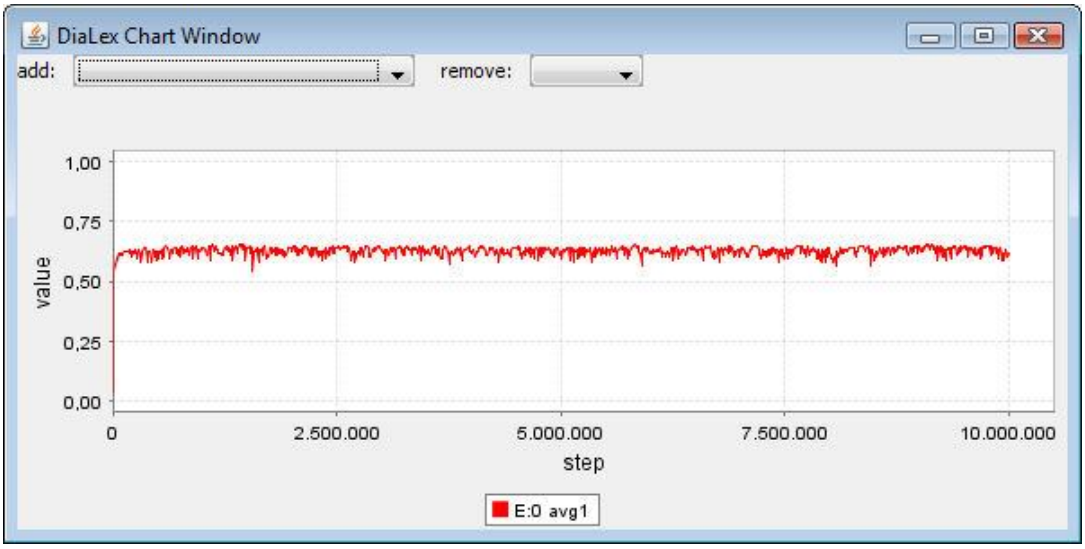


Abb. 107 Graph der Avg1-Werte

Es wird deutlich, dass sich dieser Wert im Laufe der kompletten Simulation stabil in einem Wertebereich zwischen 0.55 und 0.65 befindet. Die Wortrepräsentationen eines jeden Agenten sind damit zu jeder Zeit von den anderen unterscheidbar.

Als Nächstes muss gezeigt werden, dass zum Zeitpunkt der Aufspaltung der Agentengemeinschaft in separate Gruppen in Simulationsschritt 27000 ein gemeinsames Lexikon existiert haben muss. Zur Illustration werden im Folgenden die Tabelle des gegenseitigen Verständnisses und die Avg2-Tabelle der Agentengemeinschaft im Simulationsschritt 25000 angeführt.

Beide zeigen die vorher erwarteten Werte. Die Werte des gegenseitigen Verständnisses befinden sich auf einem Niveau, in dem von einem guten gegenseitigen Verständnis gesprochen werden kann. Einzig Agent 1 stellt eine Ausnahme dar. Dies lässt sich jedoch durch sein „Alter“ erklären. Er wurde kurz vor der Berechnung dieser Tabelle neu initialisiert. Somit befindet sich sein neuronales Netz in einem noch nahezu untrainierten Zustand.

Die Avg2-Werte zeigen ebenfalls ein sehr positives Bild. Die verbalen Repräsentationen unterscheiden sich in einem sehr geringen Maße. Zusammen mit den beiden anderen Indikatoren existieren damit genug Anzeichen um behaupten zu können, dass die Gemeinschaft der Agenten zu diesem Zeitpunkt eine

gemeinsame Sprache besitzt, mit der sie sich nahezu fehlerfrei untereinander verständigen kann.

E:0 understandingtable Step:25000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	0,17	0,12	0,17	0,12	0,17	0,25	0,21	0,17	0,21	0,21	0,12	0,17	0,25	0,12	0,17	0,12	0,25	0,21	0,17
Ag1	0,17	1,00	1,00	0,79	0,54	0,92	0,67	0,83	0,88	1,00	0,92	0,96	0,88	0,92	0,96	0,79	0,92	0,96	0,96	0,83
Ag2	0,12	1,00	1,00	0,71	0,50	0,96	0,62	0,67	0,96	1,00	0,92	1,00	1,00	0,79	1,00	1,00	0,96	0,96	1,00	0,92
Ag3	0,17	0,79	0,71	1,00	0,21	0,75	0,58	0,46	0,67	0,67	0,71	0,75	0,62	0,54	0,83	0,54	0,67	0,71	0,75	0,58
Ag4	0,12	0,54	0,50	0,21	1,00	0,46	0,21	0,46	0,33	0,67	0,54	0,42	0,42	0,62	0,42	0,50	0,42	0,42	0,42	0,17
Ag5	0,17	0,92	0,96	0,75	0,46	1,00	0,58	0,75	0,96	0,83	0,79	0,92	0,71	0,62	0,96	0,79	0,83	0,71	0,96	0,88
Ag6	0,25	0,67	0,62	0,58	0,21	0,58	1,00	0,58	0,50	0,67	0,71	0,50	0,46	0,58	0,54	0,42	0,46	0,71	0,42	0,42
Ag7	0,21	0,83	0,67	0,46	0,46	0,75	0,58	1,00	0,75	0,79	0,75	0,75	0,62	0,75	0,71	0,71	0,62	0,46	0,79	0,71
Ag8	0,17	0,88	0,96	0,67	0,33	0,96	0,50	0,75	1,00	0,83	0,88	0,88	0,67	0,67	1,00	0,88	0,79	0,75	1,00	1,00
Ag9	0,21	1,00	1,00	0,67	0,67	0,83	0,67	0,79	0,83	1,00	0,92	1,00	0,88	1,00	1,00	0,67	0,83	1,00	0,83	0,83
Ag10	0,21	0,92	0,92	0,71	0,54	0,79	0,71	0,75	0,88	0,92	1,00	0,92	0,88	0,75	0,92	0,71	0,88	0,96	0,75	0,79
Ag11	0,12	0,96	1,00	0,75	0,42	0,92	0,50	0,75	0,88	1,00	0,92	1,00	0,83	0,88	0,96	0,83	0,92	1,00	0,96	0,88
Ag12	0,17	0,88	1,00	0,62	0,42	0,71	0,46	0,62	0,67	0,88	0,88	0,83	1,00	1,00	0,88	0,79	0,79	0,83	0,54	0,67
Ag13	0,25	0,92	0,79	0,54	0,62	0,62	0,58	0,75	0,67	1,00	0,75	0,88	1,00	1,00	0,71	0,46	0,75	0,96	0,67	0,38
Ag14	0,12	0,96	1,00	0,83	0,42	0,96	0,54	0,71	1,00	1,00	0,92	0,96	0,88	0,71	1,00	0,92	0,88	0,96	1,00	1,00
Ag15	0,17	0,79	1,00	0,54	0,50	0,79	0,42	0,71	0,88	0,67	0,71	0,83	0,79	0,46	0,92	1,00	0,83	0,54	0,79	0,75
Ag16	0,12	0,92	0,96	0,67	0,42	0,83	0,46	0,62	0,79	0,83	0,88	0,92	0,79	0,75	0,88	0,83	1,00	0,96	0,79	0,71
Ag17	0,25	0,96	0,96	0,71	0,42	0,71	0,71	0,46	0,75	1,00	0,96	1,00	0,83	0,96	0,96	0,54	0,96	1,00	0,67	0,92
Ag18	0,21	0,96	1,00	0,75	0,42	0,96	0,42	0,79	1,00	0,83	0,75	0,96	0,54	0,67	1,00	0,79	0,79	0,67	1,00	0,96
Ag19	0,17	0,83	0,92	0,58	0,17	0,88	0,42	0,71	1,00	0,83	0,79	0,88	0,67	0,38	1,00	0,75	0,71	0,92	0,96	1,00

Abb. 108 Understanding-Tabelle in Simulationsschritt 25000

E:0 avg2table Step:25000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,34	0,34	0,33	0,32	0,32	0,31	0,32	0,33	0,33	0,33	0,34	0,33	0,33	0,34	0,33	0,33	0,33	0,33	0,33
Ag1	0,34	0,00	0,07	0,12	0,16	0,10	0,13	0,12	0,10	0,08	0,09	0,06	0,10	0,10	0,07	0,13	0,08	0,08	0,09	0,11
Ag2	0,34	0,07	0,00	0,13	0,17	0,08	0,15	0,14	0,09	0,09	0,11	0,08	0,12	0,12	0,06	0,09	0,07	0,09	0,08	0,09
Ag3	0,33	0,12	0,13	0,00	0,24	0,14	0,17	0,17	0,12	0,16	0,13	0,11	0,15	0,16	0,12	0,15	0,14	0,12	0,13	0,15
Ag4	0,32	0,16	0,17	0,24	0,00	0,19	0,17	0,17	0,19	0,13	0,18	0,19	0,17	0,13	0,19	0,19	0,18	0,17	0,19	0,20
Ag5	0,32	0,10	0,08	0,14	0,19	0,00	0,16	0,14	0,08	0,13	0,12	0,10	0,14	0,13	0,09	0,11	0,09	0,13	0,08	0,12
Ag6	0,31	0,13	0,15	0,17	0,17	0,16	0,00	0,15	0,15	0,13	0,15	0,16	0,13	0,15	0,15	0,16	0,15	0,14	0,16	0,16
Ag7	0,32	0,12	0,14	0,17	0,17	0,14	0,15	0,00	0,12	0,12	0,14	0,13	0,14	0,15	0,13	0,17	0,16	0,15	0,13	0,14
Ag8	0,33	0,10	0,09	0,12	0,19	0,08	0,15	0,12	0,00	0,11	0,10	0,09	0,13	0,14	0,06	0,13	0,11	0,12	0,07	0,10
Ag9	0,33	0,08	0,09	0,16	0,13	0,13	0,13	0,12	0,11	0,00	0,11	0,10	0,12	0,10	0,09	0,15	0,11	0,09	0,11	0,12
Ag10	0,33	0,09	0,11	0,13	0,18	0,12	0,15	0,14	0,10	0,11	0,00	0,09	0,10	0,12	0,10	0,15	0,11	0,10	0,12	0,14
Ag11	0,34	0,06	0,08	0,11	0,19	0,10	0,16	0,13	0,09	0,10	0,09	0,00	0,12	0,12	0,06	0,14	0,09	0,10	0,07	0,11
Ag12	0,33	0,10	0,12	0,15	0,17	0,14	0,13	0,14	0,13	0,12	0,10	0,12	0,00	0,11	0,12	0,14	0,12	0,11	0,14	0,16
Ag13	0,33	0,10	0,12	0,16	0,13	0,13	0,15	0,15	0,14	0,10	0,12	0,12	0,11	0,00	0,14	0,15	0,12	0,11	0,14	0,17
Ag14	0,34	0,07	0,06	0,12	0,19	0,09	0,15	0,13	0,06	0,09	0,10	0,06	0,12	0,14	0,00	0,13	0,09	0,09	0,06	0,08
Ag15	0,33	0,13	0,09	0,15	0,19	0,11	0,16	0,17	0,13	0,15	0,15	0,14	0,14	0,15	0,13	0,00	0,10	0,14	0,14	0,14
Ag16	0,33	0,08	0,07	0,14	0,18	0,09	0,15	0,16	0,11	0,11	0,11	0,09	0,12	0,12	0,09	0,10	0,00	0,09	0,11	0,13
Ag17	0,33	0,08	0,09	0,12	0,17	0,13	0,14	0,15	0,12	0,09	0,10	0,10	0,11	0,11	0,09	0,14	0,09	0,00	0,12	0,11
Ag18	0,33	0,09	0,08	0,13	0,19	0,08	0,16	0,13	0,07	0,11	0,12	0,07	0,14	0,14	0,06	0,14	0,11	0,12	0,00	0,10
Ag19	0,33	0,11	0,09	0,15	0,20	0,12	0,16	0,14	0,10	0,12	0,14	0,11	0,16	0,17	0,08	0,14	0,13	0,11	0,10	0,00

Abb. 109 Avg2-Tabelle in Simulationsschritt 25000

Im nächsten Teil der Analyse soll gezeigt werden, was nach der Trennung der Agentenpopulation in vier verschiedene Untergruppen im Laufe der Zeit mit der Sprache der Agenten geschieht. Hierzu werden die bekannten Indikatoren benutzt. Ein komplett anderes Bild zeigt sich bei der näheren Betrachtung der Avg2-Werte über den Simulationszeitraum.

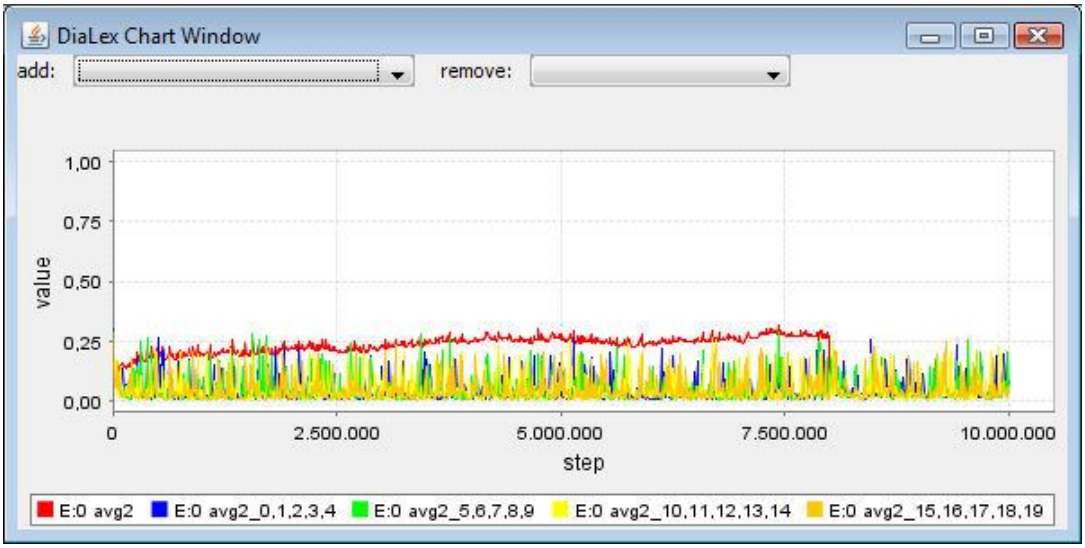


Abb. 110 Graph der Avg2-Werte

Es zeigt sich eindeutig, dass der Avg2-Wert der Gesamtpopulation im Laufe der Zeit ansteigt, bis er in Simulationsschritt 8000000 des Simulationsdurchlaufs einen Wert von 0.3 erreicht. Im Gegensatz dazu verharren die Avg2-Werte der einzelnen Agentengruppen auf einem konstant niedrigen Wert nahe 0.0. Dies belegt, dass die Wortschichten einer einzelnen Agentengruppe stets ähnliche Wörter produzieren, während sich die Wörter der Gruppen im Laufe der Zeit immer stärker bis zur Wiedervereinigung der Agentengruppen unterscheiden. Die einzelnen Ausschläge der Agentengruppen lassen sich durch den Einfluss von zu diesem Zeitpunkt untrainierten Agenten erklären. In Abbildung 111 wird eine Vergrößerung des Graphs der ersten 100000 Schritte gezeigt, die demonstriert, dass sich die Avg2-Werte mit dem Zeitpunkt der Trennung der Gesamtpopulation von den anderen Kurven zu entfernen beginnen.

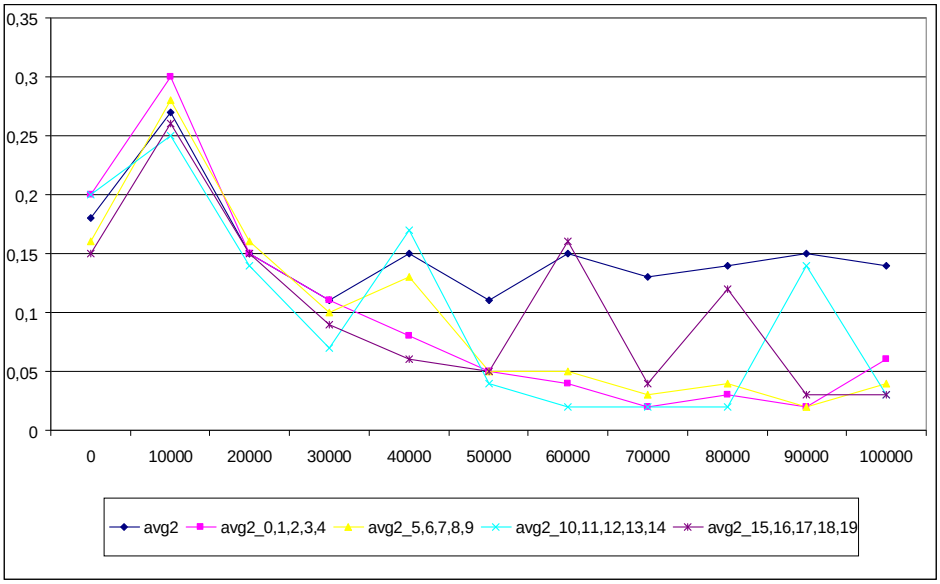


Abb. 111 Ausschnitt des Graphs der Avg2-Werte

Nachdem die Agentengruppen aufgelöst werden, fällt der gemeinschaftliche Avg2-Wert rapide ab und erreicht innerhalb von nur 100000 Simulationsschritten das Niveau der gruppeninternen Avg2-Werte. Bis zum Ende des Simulationsdurchlaufs verharren alle Avg2-Kurven auf einem niedrigen Niveau nahe 0.0. Als dritter Indikator wird der Graph der gegenseitigen Verständlichkeit angeführt. Er beweist die Vermutung, die sich nach Betrachtung der Avg1- und Avg2-Graphen aufdrängt.

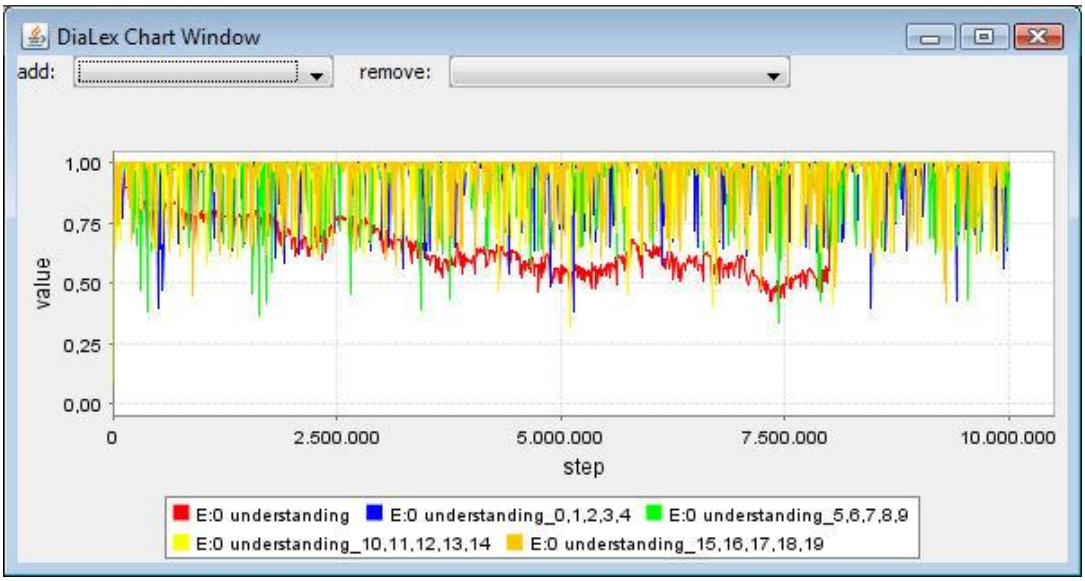


Abb. 112 Graph der Understanding-Werte

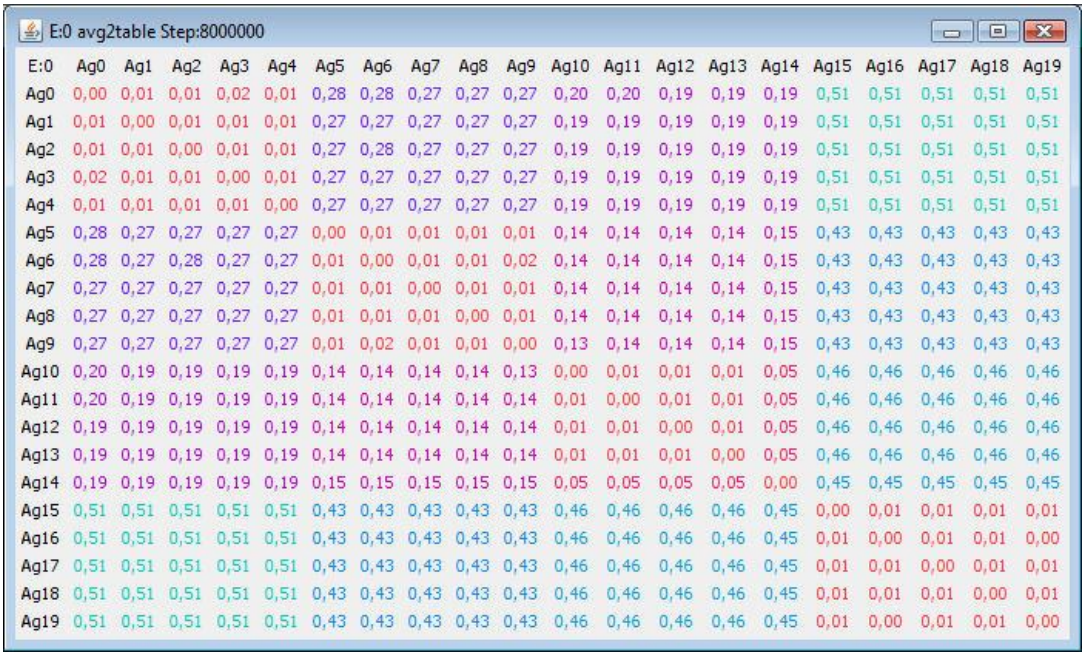
Das Verständnis in den Gruppen untereinander bleibt über den gesamten Verlauf des Simulationsdurchlaufs konstant auf einem hohen Niveau, während sich das gegenseitige Verständnis in der Gesamtpopulation bis Simulationsschritt 8000000 stetig verschlechtert. Zum Zeitpunkt der Wiedervereinigung der Agentengruppen ist es temporär unter den Wert 0.5 gefallen. Es ist auch eindeutig ersichtlich, dass sich eine Untergruppe nach einem starken Einbruch des gegenseitigen Verständnisses durch neu initialisierte Agenten wieder auf ein gemeinsames Lexikon einigen kann.

Als Nächstes wird die Veränderung des Fehlermaßes Avg2 und der gegenseitigen Verständlichkeit der Agenten analog zu den Abbildungen 111 und 112 tabellarisch im Laufe der Zeit verglichen. Dazu werden die Simulationsschritte 1000000 und 8000000 beispielhaft ausgewählt.

E:0 avg2table Step:1000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,05	0,01	0,02	0,02	0,15	0,14	0,14	0,15	0,15	0,18	0,18	0,18	0,21	0,18	0,22	0,22	0,22	0,22	0,22
Ag1	0,05	0,00	0,05	0,06	0,06	0,16	0,16	0,16	0,16	0,17	0,20	0,19	0,20	0,22	0,20	0,24	0,24	0,24	0,24	0,24
Ag2	0,01	0,05	0,00	0,02	0,02	0,15	0,14	0,15	0,15	0,15	0,19	0,19	0,19	0,21	0,19	0,22	0,21	0,21	0,21	0,21
Ag3	0,02	0,06	0,02	0,00	0,03	0,15	0,15	0,15	0,15	0,16	0,19	0,19	0,19	0,21	0,19	0,22	0,22	0,22	0,22	0,22
Ag4	0,02	0,06	0,02	0,03	0,00	0,15	0,15	0,15	0,15	0,16	0,20	0,20	0,20	0,22	0,20	0,21	0,21	0,21	0,21	0,21
Ag5	0,15	0,16	0,15	0,15	0,15	0,00	0,02	0,02	0,02	0,05	0,23	0,24	0,24	0,24	0,24	0,27	0,27	0,26	0,27	0,26
Ag6	0,14	0,16	0,14	0,15	0,15	0,02	0,00	0,02	0,01	0,05	0,23	0,23	0,23	0,23	0,23	0,27	0,27	0,27	0,27	0,27
Ag7	0,14	0,16	0,15	0,15	0,15	0,02	0,02	0,00	0,02	0,05	0,23	0,23	0,23	0,24	0,23	0,27	0,27	0,27	0,27	0,27
Ag8	0,15	0,16	0,15	0,15	0,15	0,02	0,01	0,02	0,00	0,05	0,23	0,24	0,23	0,24	0,24	0,27	0,27	0,27	0,27	0,27
Ag9	0,15	0,17	0,15	0,16	0,16	0,05	0,05	0,05	0,05	0,00	0,24	0,24	0,24	0,23	0,24	0,27	0,26	0,26	0,26	0,26
Ag10	0,18	0,20	0,19	0,19	0,20	0,23	0,23	0,23	0,23	0,24	0,00	0,02	0,01	0,08	0,01	0,34	0,34	0,34	0,34	0,34
Ag11	0,18	0,19	0,19	0,19	0,20	0,24	0,23	0,23	0,24	0,24	0,02	0,00	0,02	0,08	0,02	0,35	0,34	0,34	0,34	0,34
Ag12	0,18	0,20	0,19	0,19	0,20	0,24	0,23	0,23	0,23	0,24	0,01	0,02	0,00	0,08	0,01	0,35	0,34	0,34	0,34	0,34
Ag13	0,21	0,22	0,21	0,21	0,22	0,24	0,23	0,24	0,24	0,23	0,08	0,08	0,08	0,00	0,08	0,35	0,35	0,35	0,35	0,35
Ag14	0,18	0,20	0,19	0,19	0,20	0,24	0,23	0,23	0,24	0,24	0,01	0,02	0,01	0,08	0,00	0,34	0,34	0,34	0,34	0,34
Ag15	0,22	0,24	0,22	0,22	0,21	0,27	0,27	0,27	0,27	0,27	0,34	0,35	0,35	0,35	0,34	0,00	0,01	0,01	0,01	0,01
Ag16	0,22	0,24	0,21	0,22	0,21	0,27	0,27	0,27	0,27	0,26	0,34	0,34	0,34	0,35	0,34	0,01	0,00	0,01	0,01	0,00
Ag17	0,22	0,24	0,21	0,22	0,21	0,26	0,27	0,27	0,27	0,26	0,34	0,34	0,34	0,35	0,34	0,01	0,01	0,00	0,01	0,01
Ag18	0,22	0,24	0,21	0,22	0,21	0,27	0,27	0,27	0,27	0,26	0,34	0,34	0,34	0,35	0,34	0,01	0,01	0,01	0,00	0,01
Ag19	0,22	0,24	0,21	0,22	0,21	0,26	0,27	0,27	0,27	0,26	0,34	0,34	0,34	0,35	0,34	0,01	0,00	0,01	0,01	0,00

Abb. 113 Avg2-Tabelle in Simulationsschritt 1000000

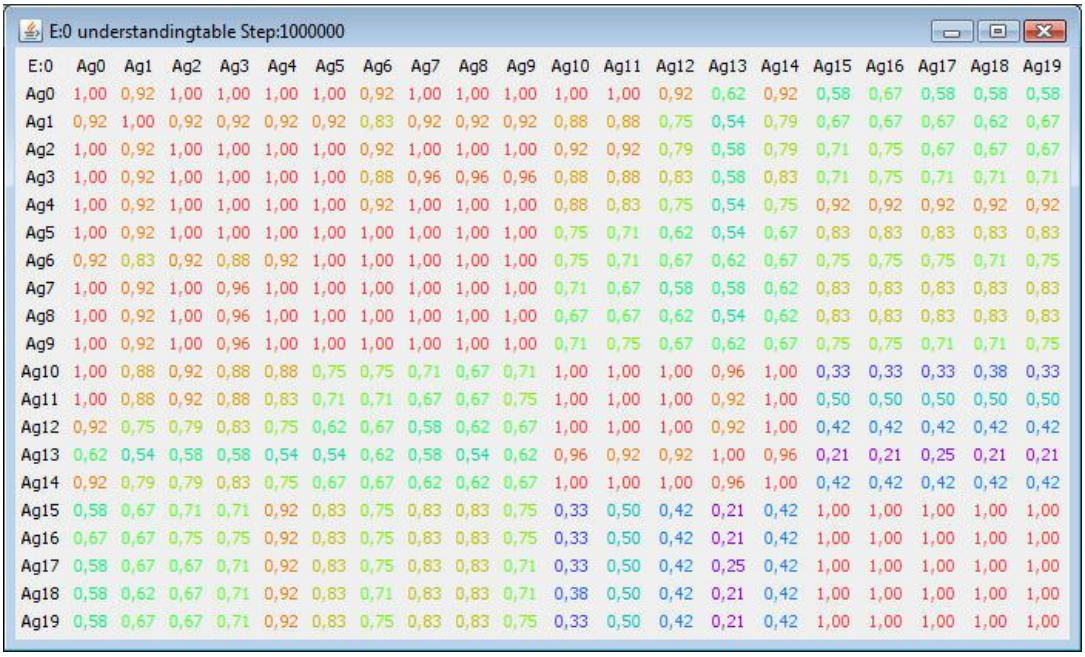


E:0 avg2table Step:8000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,01	0,01	0,02	0,01	0,28	0,28	0,27	0,27	0,27	0,20	0,20	0,19	0,19	0,19	0,51	0,51	0,51	0,51	0,51
Ag1	0,01	0,00	0,01	0,01	0,01	0,27	0,27	0,27	0,27	0,27	0,19	0,19	0,19	0,19	0,19	0,51	0,51	0,51	0,51	0,51
Ag2	0,01	0,01	0,00	0,01	0,01	0,27	0,28	0,27	0,27	0,27	0,19	0,19	0,19	0,19	0,19	0,51	0,51	0,51	0,51	0,51
Ag3	0,02	0,01	0,01	0,00	0,01	0,27	0,27	0,27	0,27	0,27	0,19	0,19	0,19	0,19	0,19	0,51	0,51	0,51	0,51	0,51
Ag4	0,01	0,01	0,01	0,01	0,00	0,27	0,27	0,27	0,27	0,27	0,19	0,19	0,19	0,19	0,19	0,51	0,51	0,51	0,51	0,51
Ag5	0,28	0,27	0,27	0,27	0,27	0,00	0,01	0,01	0,01	0,01	0,14	0,14	0,14	0,14	0,15	0,43	0,43	0,43	0,43	0,43
Ag6	0,28	0,27	0,28	0,27	0,27	0,01	0,00	0,01	0,01	0,02	0,14	0,14	0,14	0,14	0,15	0,43	0,43	0,43	0,43	0,43
Ag7	0,27	0,27	0,27	0,27	0,27	0,01	0,01	0,00	0,01	0,01	0,14	0,14	0,14	0,14	0,15	0,43	0,43	0,43	0,43	0,43
Ag8	0,27	0,27	0,27	0,27	0,27	0,01	0,01	0,01	0,00	0,01	0,14	0,14	0,14	0,14	0,15	0,43	0,43	0,43	0,43	0,43
Ag9	0,27	0,27	0,27	0,27	0,27	0,01	0,02	0,01	0,01	0,00	0,13	0,14	0,14	0,14	0,15	0,43	0,43	0,43	0,43	0,43
Ag10	0,20	0,19	0,19	0,19	0,19	0,14	0,14	0,14	0,14	0,13	0,00	0,01	0,01	0,01	0,05	0,46	0,46	0,46	0,46	0,46
Ag11	0,20	0,19	0,19	0,19	0,19	0,14	0,14	0,14	0,14	0,14	0,01	0,00	0,01	0,01	0,05	0,46	0,46	0,46	0,46	0,46
Ag12	0,19	0,19	0,19	0,19	0,19	0,14	0,14	0,14	0,14	0,14	0,01	0,01	0,00	0,01	0,05	0,46	0,46	0,46	0,46	0,46
Ag13	0,19	0,19	0,19	0,19	0,19	0,14	0,14	0,14	0,14	0,14	0,01	0,01	0,01	0,00	0,05	0,46	0,46	0,46	0,46	0,46
Ag14	0,19	0,19	0,19	0,19	0,19	0,15	0,15	0,15	0,15	0,15	0,05	0,05	0,05	0,05	0,00	0,45	0,45	0,45	0,45	0,45
Ag15	0,51	0,51	0,51	0,51	0,51	0,43	0,43	0,43	0,43	0,43	0,46	0,46	0,46	0,46	0,45	0,00	0,01	0,01	0,01	0,01
Ag16	0,51	0,51	0,51	0,51	0,51	0,43	0,43	0,43	0,43	0,43	0,46	0,46	0,46	0,46	0,45	0,01	0,00	0,01	0,01	0,00
Ag17	0,51	0,51	0,51	0,51	0,51	0,43	0,43	0,43	0,43	0,43	0,46	0,46	0,46	0,46	0,45	0,01	0,01	0,00	0,01	0,01
Ag18	0,51	0,51	0,51	0,51	0,51	0,43	0,43	0,43	0,43	0,43	0,46	0,46	0,46	0,46	0,45	0,01	0,01	0,01	0,00	0,01
Ag19	0,51	0,51	0,51	0,51	0,51	0,43	0,43	0,43	0,43	0,43	0,46	0,46	0,46	0,46	0,45	0,01	0,00	0,01	0,01	0,00

Abb. 114 Avg2-Tabelle in Simulationsschritt 8000000

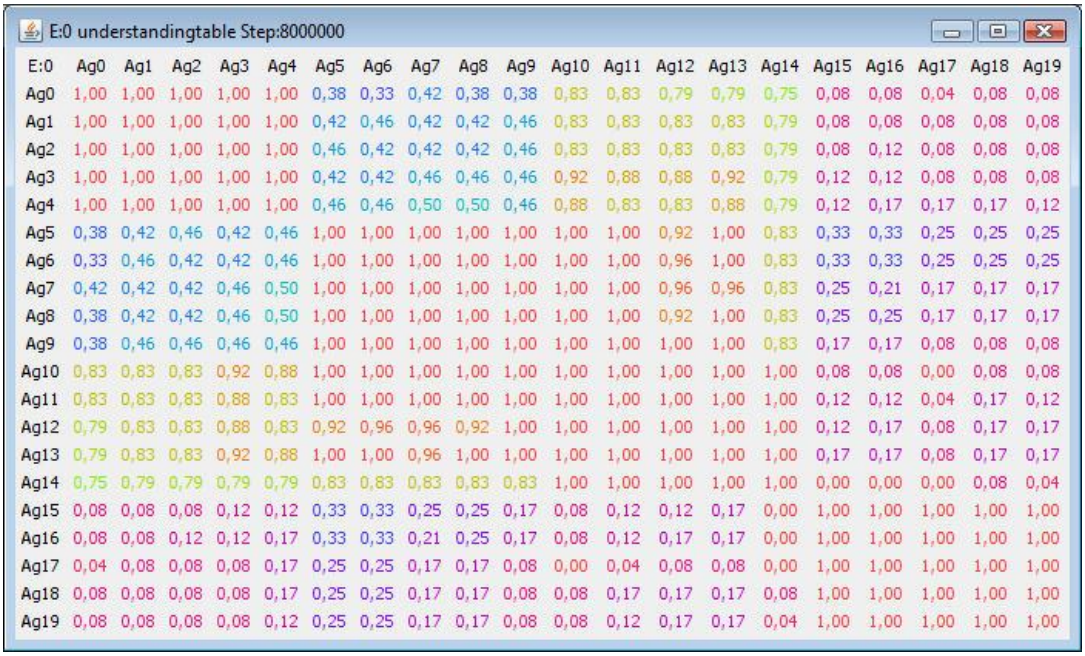
Es wird deutlich, dass die gruppeninternen verbalen Repräsentationen über die gesamte Simulation konstant ähnlich bleiben, während die Unterschiede zu anderen Gruppen im Laufe der Simulation anwachsen.



E:0 understandingtable Step:1000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	0,92	1,00	1,00	1,00	1,00	0,92	1,00	1,00	1,00	1,00	1,00	0,92	0,62	0,92	0,58	0,67	0,58	0,58	0,58
Ag1	0,92	1,00	0,92	0,92	0,92	0,92	0,83	0,92	0,92	0,92	0,88	0,88	0,75	0,54	0,79	0,67	0,67	0,67	0,62	0,67
Ag2	1,00	0,92	1,00	1,00	1,00	1,00	0,92	1,00	1,00	1,00	0,92	0,92	0,79	0,58	0,79	0,71	0,75	0,67	0,67	0,67
Ag3	1,00	0,92	1,00	1,00	1,00	1,00	0,88	0,96	0,96	0,96	0,88	0,88	0,83	0,58	0,83	0,71	0,75	0,71	0,71	0,71
Ag4	1,00	0,92	1,00	1,00	1,00	1,00	0,92	1,00	1,00	1,00	0,88	0,83	0,75	0,54	0,75	0,92	0,92	0,92	0,92	0,92
Ag5	1,00	0,92	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,71	0,62	0,54	0,67	0,83	0,83	0,83	0,83	0,83
Ag6	0,92	0,83	0,92	0,88	0,92	1,00	1,00	1,00	1,00	1,00	0,75	0,71	0,67	0,62	0,67	0,75	0,75	0,75	0,71	0,75
Ag7	1,00	0,92	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,71	0,67	0,58	0,58	0,62	0,83	0,83	0,83	0,83	0,83
Ag8	1,00	0,92	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,67	0,67	0,62	0,54	0,62	0,83	0,83	0,83	0,83	0,83
Ag9	1,00	0,92	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,71	0,75	0,67	0,62	0,67	0,75	0,75	0,71	0,71	0,75
Ag10	1,00	0,88	0,92	0,88	0,88	0,75	0,75	0,71	0,67	0,71	1,00	1,00	1,00	0,96	1,00	0,33	0,33	0,33	0,38	0,33
Ag11	1,00	0,88	0,92	0,88	0,83	0,71	0,71	0,67	0,67	0,75	1,00	1,00	1,00	0,92	1,00	0,50	0,50	0,50	0,50	0,50
Ag12	0,92	0,75	0,79	0,83	0,75	0,62	0,67	0,58	0,62	0,67	1,00	1,00	1,00	0,92	1,00	0,42	0,42	0,42	0,42	0,42
Ag13	0,62	0,54	0,58	0,58	0,54	0,54	0,62	0,58	0,54	0,62	0,96	0,92	0,92	1,00	0,96	0,21	0,21	0,25	0,21	0,21
Ag14	0,92	0,79	0,79	0,83	0,75	0,67	0,67	0,62	0,62	0,67	1,00	1,00	1,00	0,96	1,00	0,42	0,42	0,42	0,42	0,42
Ag15	0,58	0,67	0,71	0,71	0,92	0,83	0,75	0,83	0,83	0,75	0,33	0,50	0,42	0,21	0,42	1,00	1,00	1,00	1,00	1,00
Ag16	0,67	0,67	0,75	0,75	0,92	0,83	0,75	0,83	0,83	0,75	0,33	0,50	0,42	0,21	0,42	1,00	1,00	1,00	1,00	1,00
Ag17	0,58	0,67	0,67	0,71	0,92	0,83	0,75	0,83	0,83	0,71	0,33	0,50	0,42	0,25	0,42	1,00	1,00	1,00	1,00	1,00
Ag18	0,58	0,62	0,67	0,71	0,92	0,83	0,71	0,83	0,83	0,71	0,38	0,50	0,42	0,21	0,42	1,00	1,00	1,00	1,00	1,00
Ag19	0,58	0,67	0,67	0,71	0,92	0,83	0,75	0,83	0,83	0,75	0,33	0,50	0,42	0,21	0,42	1,00	1,00	1,00	1,00	1,00

Abb. 115 Understanding-Tabelle in Simulationsschritt 1000000



E:0 understandingtable Step:8000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	0,38	0,33	0,42	0,38	0,38	0,83	0,83	0,79	0,79	0,75	0,08	0,08	0,04	0,08	0,08
Ag1	1,00	1,00	1,00	1,00	1,00	0,42	0,46	0,42	0,42	0,46	0,83	0,83	0,83	0,83	0,79	0,08	0,08	0,08	0,08	0,08
Ag2	1,00	1,00	1,00	1,00	1,00	0,46	0,42	0,42	0,42	0,46	0,83	0,83	0,83	0,83	0,79	0,08	0,12	0,08	0,08	0,08
Ag3	1,00	1,00	1,00	1,00	1,00	0,42	0,42	0,46	0,46	0,46	0,92	0,88	0,88	0,92	0,79	0,12	0,12	0,08	0,08	0,08
Ag4	1,00	1,00	1,00	1,00	1,00	0,46	0,46	0,50	0,50	0,46	0,88	0,83	0,83	0,88	0,79	0,12	0,17	0,17	0,17	0,12
Ag5	0,38	0,42	0,46	0,42	0,46	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,92	1,00	0,83	0,33	0,33	0,25	0,25	0,25
Ag6	0,33	0,46	0,42	0,42	0,46	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	0,83	0,33	0,33	0,25	0,25	0,25
Ag7	0,42	0,42	0,42	0,46	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	0,96	0,83	0,25	0,21	0,17	0,17	0,17
Ag8	0,38	0,42	0,42	0,46	0,50	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,92	1,00	0,83	0,25	0,25	0,17	0,17	0,17
Ag9	0,38	0,46	0,46	0,46	0,46	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,83	0,17	0,17	0,08	0,08	0,08
Ag10	0,83	0,83	0,83	0,92	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,08	0,08	0,00	0,08	0,08
Ag11	0,83	0,83	0,83	0,88	0,83	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,04	0,17	0,12
Ag12	0,79	0,83	0,83	0,88	0,83	0,92	0,96	0,96	0,92	1,00	1,00	1,00	1,00	1,00	1,00	0,12	0,17	0,08	0,17	0,17
Ag13	0,79	0,83	0,83	0,92	0,88	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,17	0,17	0,08	0,17	0,17
Ag14	0,75	0,79	0,79	0,79	0,79	0,83	0,83	0,83	0,83	0,83	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,00	0,08	0,04
Ag15	0,08	0,08	0,08	0,12	0,12	0,33	0,33	0,25	0,25	0,17	0,08	0,12	0,12	0,17	0,00	1,00	1,00	1,00	1,00	1,00
Ag16	0,08	0,08	0,12	0,12	0,17	0,33	0,33	0,21	0,25	0,17	0,08	0,12	0,17	0,17	0,00	1,00	1,00	1,00	1,00	1,00
Ag17	0,04	0,08	0,08	0,08	0,17	0,25	0,25	0,17	0,17	0,08	0,00	0,04	0,08	0,08	0,00	1,00	1,00	1,00	1,00	1,00
Ag18	0,08	0,08	0,08	0,08	0,17	0,25	0,25	0,17	0,17	0,08	0,08	0,17	0,17	0,17	0,08	1,00	1,00	1,00	1,00	1,00
Ag19	0,08	0,08	0,08	0,08	0,12	0,25	0,25	0,17	0,17	0,08	0,08	0,12	0,17	0,17	0,04	1,00	1,00	1,00	1,00	1,00

Abb. 116 Understanding-Tabelle in Simulationsschritt 8000000

Die Tabellen des gegenseitigen Verständnisses (Abb. 115 und Abb. 116) zeichnen ein ähnliches Bild. Während das gruppeninterne Verständnis permanent sehr hoch anzusiedeln ist, zeigt sich auch hier, dass das Verständnis zwischen Mitgliedern unterschiedlicher Gruppen stetig schlechter wird.

Dies ändert sich jedoch, nachdem die Agenten wieder alle miteinander kommunizieren können. Innerhalb von nur 100000 Schritten wird ein gemeinsames Lexikon aller Agenten gebildet. Die Wortrepräsentationen der einzelnen Agenten unterscheiden sich, wie in den folgenden Abbildungen aus dem letzten Schritt des Simulationdurchlaufs ersichtlich, nur noch marginal.

E:0 avg2table Step:10000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	0,00	0,12	0,02	0,03	0,09	0,07	0,02	0,02	0,02	0,02	0,03	0,11	0,02	0,02	0,02	0,02	0,03	0,02	0,12	0,02
Ag1	0,12	0,00	0,13	0,11	0,12	0,13	0,12	0,12	0,12	0,12	0,11	0,12	0,12	0,12	0,12	0,12	0,12	0,12	0,10	0,12
Ag2	0,02	0,13	0,00	0,03	0,09	0,07	0,02	0,02	0,01	0,02	0,02	0,11	0,01	0,02	0,01	0,02	0,03	0,01	0,12	0,02
Ag3	0,03	0,11	0,03	0,00	0,09	0,07	0,03	0,03	0,03	0,03	0,03	0,09	0,03	0,03	0,03	0,03	0,04	0,03	0,11	0,03
Ag4	0,09	0,12	0,09	0,09	0,00	0,06	0,09	0,09	0,10	0,10	0,08	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,10	0,09
Ag5	0,07	0,13	0,07	0,07	0,06	0,00	0,07	0,06	0,07	0,07	0,06	0,10	0,06	0,06	0,06	0,06	0,06	0,07	0,10	0,07
Ag6	0,02	0,12	0,02	0,03	0,09	0,07	0,00	0,02	0,02	0,02	0,02	0,10	0,02	0,02	0,02	0,02	0,04	0,02	0,12	0,02
Ag7	0,02	0,12	0,02	0,03	0,09	0,06	0,02	0,00	0,02	0,02	0,02	0,10	0,02	0,02	0,02	0,02	0,03	0,02	0,12	0,01
Ag8	0,02	0,12	0,01	0,03	0,10	0,07	0,02	0,02	0,00	0,02	0,02	0,10	0,02	0,02	0,01	0,02	0,03	0,01	0,12	0,01
Ag9	0,02	0,12	0,02	0,03	0,10	0,07	0,02	0,02	0,02	0,00	0,03	0,10	0,02	0,02	0,01	0,02	0,04	0,02	0,12	0,02
Ag10	0,03	0,11	0,02	0,03	0,08	0,06	0,02	0,02	0,02	0,03	0,00	0,10	0,02	0,02	0,02	0,02	0,04	0,02	0,11	0,02
Ag11	0,11	0,12	0,11	0,09	0,09	0,10	0,10	0,10	0,10	0,10	0,10	0,00	0,10	0,10	0,11	0,10	0,10	0,10	0,11	0,11
Ag12	0,02	0,12	0,01	0,03	0,09	0,06	0,02	0,02	0,02	0,02	0,02	0,10	0,00	0,01	0,01	0,02	0,04	0,01	0,12	0,02
Ag13	0,02	0,12	0,02	0,03	0,09	0,06	0,02	0,02	0,02	0,02	0,02	0,10	0,01	0,00	0,01	0,02	0,04	0,01	0,11	0,01
Ag14	0,02	0,12	0,01	0,03	0,09	0,06	0,02	0,02	0,01	0,01	0,02	0,11	0,01	0,01	0,00	0,02	0,03	0,01	0,12	0,01
Ag15	0,02	0,12	0,02	0,03	0,09	0,06	0,02	0,02	0,02	0,02	0,02	0,10	0,02	0,02	0,02	0,00	0,04	0,02	0,11	0,02
Ag16	0,03	0,12	0,03	0,04	0,09	0,06	0,04	0,03	0,03	0,04	0,04	0,10	0,04	0,04	0,03	0,04	0,00	0,03	0,11	0,04
Ag17	0,02	0,12	0,01	0,03	0,09	0,07	0,02	0,02	0,01	0,02	0,02	0,10	0,01	0,01	0,01	0,02	0,03	0,00	0,11	0,01
Ag18	0,12	0,10	0,12	0,11	0,10	0,10	0,12	0,12	0,12	0,12	0,11	0,11	0,12	0,11	0,12	0,11	0,11	0,11	0,00	0,12
Ag19	0,02	0,12	0,02	0,03	0,09	0,07	0,02	0,01	0,01	0,02	0,02	0,11	0,02	0,01	0,01	0,02	0,04	0,01	0,12	0,00

Abb. 117 Avg2-Tabelle in Simulationsschritt 10000000

Dies führt ebenso zu einem hohen Level des gegenseitigen Verständnisses, was in der nachfolgenden Abbildung gezeigt wird. Die Werte erreichen nahezu ausnahmslos den Idealwert 1.0. Abweichungen hier sind nachweislich durch noch junge, untrainierte Agenten bedingt.

E:0 understandingtable Step:10000000

E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag1	0,88	1,00	0,88	0,88	0,75	0,71	0,88	0,88	0,88	0,83	0,88	0,75	0,83	0,88	0,88	0,88	0,83	0,88	0,83	0,88
Ag2	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag3	1,00	0,88	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag4	1,00	0,75	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	0,79	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag5	1,00	0,71	1,00	0,96	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,83	1,00	1,00	1,00	1,00	0,96	1,00	0,92	1,00
Ag6	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag7	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag8	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag9	1,00	0,83	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag10	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag11	0,96	0,75	0,96	0,96	0,79	0,83	0,96	0,96	0,96	0,96	0,96	1,00	0,96	0,96	0,96	0,96	0,92	0,96	0,88	0,96
Ag12	1,00	0,83	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag13	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag14	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag15	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag16	1,00	0,83	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	0,92	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag17	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00
Ag18	0,96	0,83	0,96	0,96	0,96	0,92	0,96	0,96	0,96	0,96	0,96	0,88	0,96	0,96	0,96	0,96	0,96	0,96	1,00	0,96
Ag19	1,00	0,88	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00	1,00	1,00	1,00	1,00	1,00	0,96	1,00

Abb. 118 Understanding-Tabelle in Simulationsschritt 10000000

In einem letzten Teil der Analyse werden beispielhaft verbale Repräsentationen eines Eingabemusters von Mitgliedern verschiedener Gruppen miteinander verglichen.

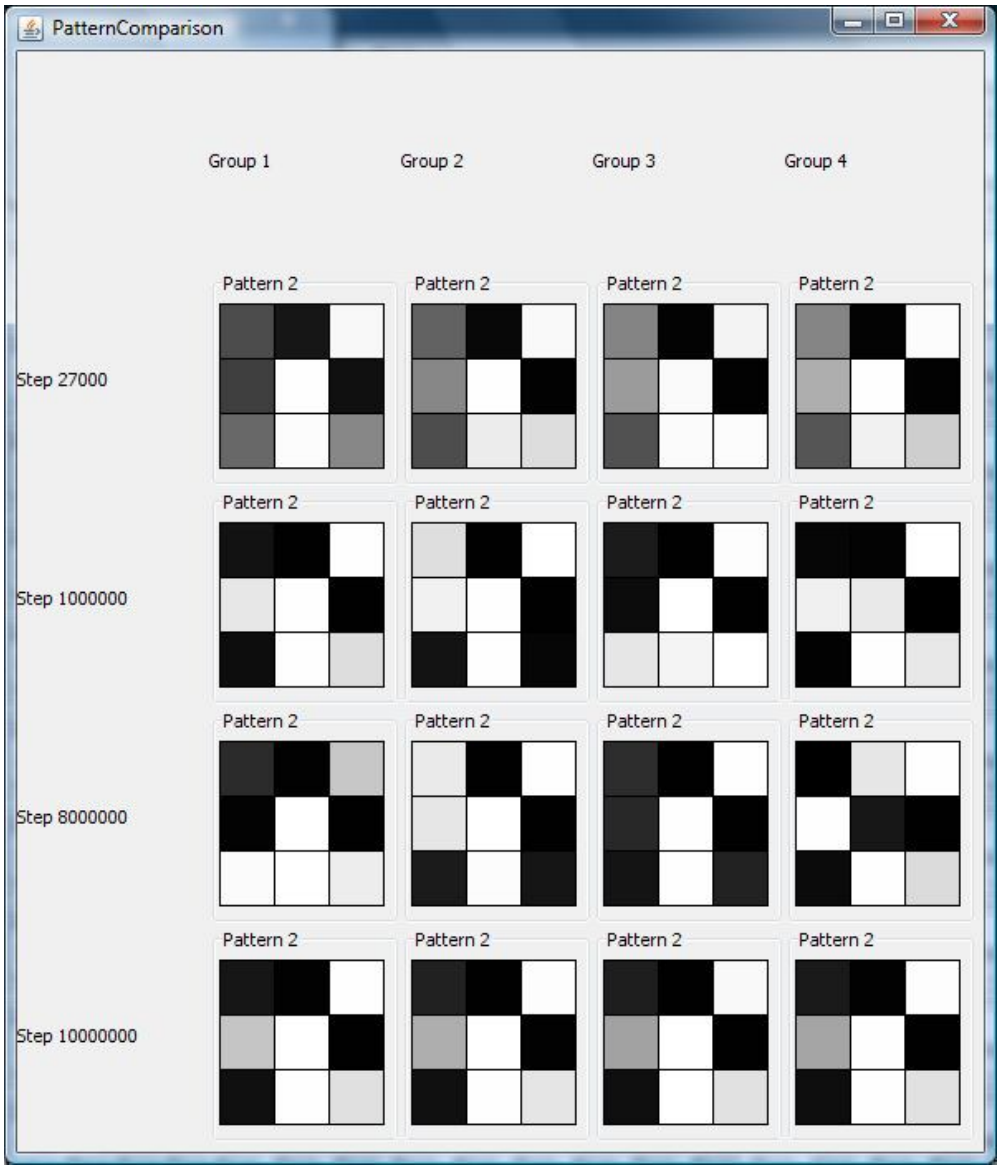


Abb. 119 Vergleich der Wortmuster

Im Laufe der Simulation entwickeln sich die Wortrepräsentationen der verschiedenen Gruppen auseinander. In der Ursprache der Agenten, in Schritt 27000, bevor die Gesamtpopulation in verschiedene Gruppen aufgeteilt wird, sind die Wortmuster der Agenten kaum zu unterscheiden. Bereits in Schritt 1000000 jedoch haben sich die Muster der Agentengruppen in verschiedene Richtungen verändert. Dennoch gleichen sich in diesem Simulationsdurchlauf die Wörter der

Gruppen 1 und 4 bei dem ausgewählten Eingabemuster. Außerdem sind klare Ähnlichkeiten zu den Wortpattern der Ursprache zu erkennen.

In Simulationsschritt 8000000 ist die gruppenbezogene, synchrone Ähnlichkeit ebenfalls aufgehoben. Die Wörter der Agentengruppen erscheinen voneinander vollkommen unabhängig. Diese späten Wortrepräsentationen ähneln jedoch immer noch ihren jeweiligen Vorstufen. Eine diachrone Verwandtschaft ist somit ersichtlich.

Nach der Agentenwiedervereinigung hat sich wieder ein gemeinsames Lexikon innerhalb der Agentengemeinschaft ausgebildet. Unterschiede zwischen den einzelnen Gruppen sind nicht mehr erkennbar. Erstaunlicherweise gleicht das „neue“ Gemeinschaftswort stark dem Wort in der Ursprache. Dies lässt sich jedoch durch den Einfluss des Ursprungsworts auf alle nachfolgenden Sprachen erklären, sodass die Kombination derer ein dem Ursprungswort ähnliches Wortpattern erzeugt. [17]

Während des Simulationsdurchlaufs hat sich damit die gemeinsame Sprache einer Agentenpopulation über mehrere Zwischenstadien dahingehend verändert, dass vor der Auflösung der Agentenuntergruppen keine synchronen Gemeinsamkeiten zwischen einzelnen Wörtern der Agentengruppen erkennbar sind, obwohl eine diachrone Betrachtung der Wortpattern der Untergruppen der Agentenpopulation zur gemeinsamen Ursprache führt. Damit ist es gelungen, ein existentes funktionierendes gemeinsames Lexikon einer Agentenpopulation durch eine Menge neuer, einander verschiedener Lexika neuer Agentenuntergruppen abzulösen.

Nach dem Zusammenführen aller Agenten verlieren sich die dialektalen Unterschiede jedoch in einem sehr kurzen Zeitraum, eine neue Gemeinschaftssprache etabliert sich, die Merkmale aller Vorgängersprachen trägt und somit auch der Ursprungssprache ähnelt. Eine Simulation einer Sprachspaltung mit nachfolgender Wiedervereinigung der Sprachgruppen ist hiermit geglückt.

[17] In Simulationsdurchläufen, in denen mehrere Agentengruppen unabhängig voneinander ein Lexikon erlernt haben, tritt ein vergleichbarer Effekt ein. Das durch Gruppenvereinigung entstehende neue Wort ist dabei eine Konkatenation der jeweils häufigsten Ausprägung der Aktivierungswerte der Ursprungswörter. Bei gleich häufiger verschiedenartiger Ausprägung eines Wertes bildet sich erst ein Mittelwert, der im Laufe der Zeit zu einem der beiden Extreme konvergiert.

5. Fazit und Ausblick

Dieses Kapitel als Abschluss dieser Dissertation reflektiert die Erkenntnisse dieser Arbeit, die aus den Ergebnissen der im vorangegangenen Kapitel angeführten Simulationen gewonnen werden. Anschließend gibt es einen Ausblick auf mögliche weitere Schritte in der Forschung auf dem Weg zu dem, realistisch betrachtet, weit entfernten Ziel, ein Modell zu entwickeln, das alle Facetten der menschlichen Sprache und ihrer Entstehung beinhaltet.

5.1 Fazit

Die im vorigen Kapitel erläuterten Simulationen innerhalb des in dieser Arbeit erstellten Modells zeigen, dass eine Dialektbildung und, in weiterer Folge, eine Sprachspaltung, nachgestellt werden können. Damit ist es gelungen, eine weitere Eigenschaft der natürlichen Sprachen in einem Simulationsmodell abzubilden und simulierbar zu machen.

Die aus diesem Grund im Modell enthaltenen Störfaktoren zeigen jedoch kein einheitliches Bild. (vgl Abb. 120) Nur drei der fünf Faktoren ermöglichen bei einem alleinigen Einsatz überhaupt eine Dialektbildung zwischen Agentengruppen. Diese werden in den Unterkapiteln 4.1 bis 4.6 mit einer identischen Simulation ohne Störfaktoren, die erwartungsgemäß keine Dialekte ausbildet, verglichen und ausgewertet. Eine Dialektspaltung ist mit dem Einsatz der Störfaktoren Agentensterblichkeit, Vergessen durch Abschwächung und Rauschen während der Perzeption möglich. Ein Rauschen innerhalb der Kommunikation und eine Wortkomprimierung mittels diskreter Kosinustransformation führen jedoch niemals zu einer Ausbildung von Dialekten.

Erstaunlich ist dabei die Tatsache, dass die beiden Formen des Rauschens bis auf die Stelle des Einwirkens absolut identisch implementiert wurden. Offensichtlich sind die neuronalen Netze der Agenten weniger anfällig für Fehler innerhalb eines Wortes als für eine fehlerbehaftete Perzeption.

Die hemmende Wirkung der Wortkomprimierung ist sogar so stark, dass sie den dialektbildenden Einfluss anderer Störfaktoren aufhebt. Dies wird in Unterkapitel 4.7 gezeigt, wo sie beispielhaft den Faktor Agentensterblichkeit egalisiert.

Unter-kapitel	Störfaktor	Stärke bei nachfolgender Auswirkung		
		Keine Auswirkung	Dialekt-bildung	Kein Lexikon
4.1	kein	n.a.	n.a.	n.a.
4.2	Rauschen während Perzeption	0.001	0.2	0.45
4.3	Rauschen während Kommunikation	0.62	-	0.62
4.4	Sterblichkeit der Agenten	-	0.000005 0.00001	0.0005
4.5	Vergessen durch Abschwächen		0.2	
4.6	Wortkomprimierung durch DCT	0.7		
4.7	Wechselwirkungen (Sterblichkeit / DCT)	0.00001 / 0.7		
4.8	Sprachspaltung + Wiedervereinigung (Sterblichkeit)		0.00001	

Abb. 120 Übersicht der Simulationsdurchläufe in Kapitel 4

Die achte und gleichzeitig letzte in dieser Arbeit analysierte Simulation untersucht in Unterkapitel 4.8 den Prozess einer Dialektspaltung oder Sprachspaltung[18] mit anschließender Wiedervereinigung. Sie soll damit Aufschluss über die Entstehung von Mischsprachen geben. Beispiele unter natürlichen Sprachen finden sich in einem sehr großen Umfang. An dieser Stelle seien die verschiedenen rätoromanischen Dialekte zu nennen, die sich möglicherweise durch die Einführung einer künstlich geschaffenen gemeinsamen Schriftsprache vor mehreren Jahren wieder in einem Prozess der Annäherung befinden. Als weiteres Beispiel ist das Portuñol zu nennen, eine in Uruguay entstandene Mischsprache aus Spanisch und Portugiesisch. Diese beiden Sprachen gehen ebenfalls auf eine gemeinsame Ursprache, die lateinische Sprache, zurück und haben in diesem Gebiet, nicht zuletzt aufgrund mehrfacher territorialer Verschiebungen, eine Wiedervereinigung erlebt.

[18] Eine genaue Unterscheidung kann wegen des fließenden Übergangs nicht getroffen werden.

Das Simulationsmodell der Arbeit bietet, als Ganzes gesehen, die Möglichkeit, einen grundsätzlich unbeobachten sozialen Prozess zu rekonstruieren. Die reale Entsprechung innerhalb menschlicher Gesellschaften der hier gezeigten Effekte dauert eher Jahrtausende als Jahrzehnte und ist stets nur in der Endphase, nach einer Verschriftlichung, in einer Weise dokumentiert worden, die wissenschaftlich untersuchbar ist. Diese Art der Dokumentierung allerdings hat zur Folge, dass die niedergeschriebene Sprache zu einer Standardsprache wird, die durch die Verschriftlichung weitaus stabiler gegenüber Veränderungen ist als eine nicht niedergeschriebene Sprachvariante. Somit beinhalten die empirisch untersuchbaren Artefakte, die die Entwicklung von Sprachen und Dialekten technisch möglich machen, nicht mehr die Prozesse, die eigentlich untersucht werden wollen.

Die Arbeit reiht sich somit in eine neu entstehende Disziplin ein, die man *Computational Archeology* nennen könnte und die sich innerhalb der *Computational Social Science* wachsender Beliebtheit erfreut.

5.2 Ausblick

Wie bereits in Abschnitt 3.5.1 angedeutet, sind aktuelle Simulationsmodelle nicht einmal ansatzweise in der Lage, die Komplexität einer menschlichen Sprache nachzubilden. Dies hat zwei Gründe: Zuerst ist die Rechenleistung derzeit nicht ausreichend, um einen menschlichen Organismus oder auch nur das menschliche Gehirn in annehmbarer Zeit zu simulieren. Aufgrund stetiger Weiterentwicklungen im Hardware-Bereich scheint diese Aufgabe jedoch auf mittlere Sicht zu lösen zu sein. Viel schwerer wiegt das Fehlen eines Modells, das die Vielfältigkeit der menschlichen Sprachen abbilden und in den menschlichen Gehirnen nachempfundenen neuronalen Netzen kodieren kann.[19]

Selbstverständlich ist es an dieser Stelle nicht möglich, ein derartiges Modell zu beschreiben. Es scheint jedoch sinnvoll, auf Aspekte hinzuweisen, die mit den neuronalen Netzen, die in diesem Modell eingesetzt werden, nicht realisierbar sind, um eine Entwicklung leistungsfähigerer neuronaler Netze voranzutreiben.

Die in dieser Arbeit eingesetzten neuronalen Netze sind in der genutzten Form nicht in der Lage, ein Objekt anhand seiner Eigenschaften zu beschreiben. Kategorisierungen nach Formen und Farben beispielsweise sind somit nicht möglich. Abhilfe könnten hier neuronale Netze, die auf einer Fuzzy-Logik basieren, schaffen. Jedoch scheint der Bezug zu biologischen Neuronen fraglich, da die Prozesse an Synapsenverbindungen keine Gemeinsamkeiten mit einer Fuzzy-Funktion bieten.

[19] Eine andere Form der Wissenskodierung erscheint aufgrund der Ähnlichkeit zwischen der Funktionsweise neuronaler Netze und Nervenzellen und damit auch dem Gehirn wenig praktikabel.

Eine weitere Schwachstelle von neuronalen Netzen bietet die fehlende Möglichkeit, eine Mehrsprachigkeit der Agenten innerhalb eines einzigen neuronalen Netzes darzustellen. Dies ist nur durch unterschiedliche Eingabemuster realisierbar. Wünschenswert wäre es jedoch, dass aus einem Objekt mehrere Bezeichnungen hergeleitet werden könnten. Hierbei fällt eine Ähnlichkeit zu der Fähigkeit der Kategorisierung auf. Möglicherweise stehen beide Punkte miteinander in Beziehung und lassen sich auf eine ähnliche Weise lösen.

Zuletzt bleibt die Problematik der Kodierung einer Syntax im neuronalen Netz zu erwähnen. Wie kann ein einziges neuronales Netz gleichzeitig lexikalische als auch syntaktische Fähigkeiten entfalten? Schwieriger noch, wie gelingt es einem neuronalen Netz im Laufe der Sprachentwicklung einer Gemeinschaft, eine Grammatik aus dem Grund der effizienteren Wort/Satz-Bedeutungskodierung „aus dem Nichts“ zu generieren? Auf diesem Gebiet ist offensichtlich noch viel Forschungsarbeit vonnöten.

Derartig mächtige, auf neuronalen Netzen basierende Modelle sind in naher Zukunft nicht zu erwarten. Dennoch scheinen sie zu diesen Schritten in der Lage zu sein. Einen Beweis liefert in diesem Fall die Natur. Die menschlichen Sprachen sind in Laufe der Evolution in nichts anderem als in großen neuronalen Netzen entstanden. Daher ist es sehr wahrscheinlich, dass das in dieser Arbeit genutzte Modell in der Art erweiterbar ist, dass es auf lange Sicht durchaus Sprachen erzeugen kann, die den menschlichen Sprachen nahe kommen.

5.3 Weitere Anwendungen

Neben der Simulation von Sprachveränderungen in natürlichen Sprachen bietet das Modell dieser Arbeit ebenfalls die Möglichkeit weitere sprachliche Veränderungen nachzustellen.

Ein mögliches Anwendungsszenario ist das Verhalten der Internetnutzer beim Tagging von Fotos. Auf diversen Webseiten wie beispielsweise Flickr[20] existiert die Möglichkeit, die dort zur Verfügung stehenden Fotos aller Nutzer mit Schlagworten zu versehen. Die durch diese Tags entstandenen kollaborativen Tag-Wolken werden als Folksonomien bezeichnet, ein Kofferwort, gebildet aus Folk und Taxonomie. Eine Folksonomie ist also die Menge von Klassifizierungen, die von einer breiten Nutzerbasis, dem Volk, gemeinschaftlich vorgenommen wird.
Abbasi und Staab untersuchen in [AS10] den Zusammenhang zwischen Fotos und der Verteilung der entsprechenden Tags eines Fotos. Es zeigt sich, dass viele Fotos mit gleichbedeutenden Tags verschiedener Sprachen und Verallgemeinerungen

[20] http://www.flickr.com

bzw. Spezialisierungen existieren. Abbildung 121 demonstriert dies am Beispiel der Tags des Worts „hibiscus“.

Tag	Generalized Tags
hibiscus	hibiscus(1.00), flower(0.61),flowers(0.25),red(0.17),macro(0.15)
hibiskus	hibiskus(1.00), hibiscus(0.67),flower(0.47),blume(0.33),garten(0.29)
ibisco	ibisco(1.00), flower(0.65),hibiscus(0.52),flor(0.43),red(0.30)
ibiscus	ibiscus(1.00), flower(0.55),hibiscus(0.45),nature(0.41),flowers(0.41)
hibisco	hibisco(1.00), flor(0.66),flower(0.65),hibiscus(0.39),macro(0.21)
rosemallow	rosemallow(1.00), hibiscus(0.52),flower(0.48),malvaceae(0.38),garden(0.24)
gumamela	gumamela(1.00), flower(0.58),philippines(0.38),hibiscus(0.35),red(0.23)
roseofsharon	roseofsharon(1.00), flower(0.66),macro(0.28),flowers(0.28),hibiscus(0.21)
malvaceae	malvaceae(1.00), flower(0.73),hibiscus(0.62),flowers(0.28),macro(0.22)

Abb. 121 Semantische Beziehungen des Tags „hibiscus“. Quelle: [AS10]

Interessant in diesem Zusammenhang könnte die Fragestellung sein, ob sich die verschiedensprachlichen Tag-Bezeichnungen in Folksonomies in ähnlicher Weise wie die Substantive natürlicher Sprachen verhalten. Durch die zunehmende Globalisierung und den zunehmenden Kontakt zwischen Menschen verschiedenster Länder und Muttersprachen könnte sich beispielsweise ein gemeinsamer Bezeichner für ähnliche Tags verschiedener Sprachen, möglicherweise sogar ein globaler, universaler Bezeichner für ein bestimmtes Objekt in Folksonomies herausbilden. Dieser Prozess könnte der Bildung eines gemeinsamen Lexikons einer Mischsprache sehr nahe kommen. Aus diesem Grund scheint das Modell dieser Arbeit ebenfalls für diesen Anwendungsfall benutzbar zu sein. Die Eingabemuster des Modells entsprächen dabei den Fotos einer Webseite, die Wortmuster der Agenten den Tags, Agentengruppen mit einem gemeinsamen Lexikon einer Gruppe von Personen, die ein gemeinsames Tag für ein Foto benutzen.

Die in Unterkapitel 5.2 erläuterten möglichen Erweiterungen des Modells wie Mehrsprachigkeit und Kategorisierungen könnten innerhalb der Untersuchungen zum Tagging-Verhalten ebenfalls genutzt werden. Mit ihnen wäre es möglich, Tags verschiedener Sprachen, Spezialisierungen und Verallgemeinerungen nachzubilden.

Leider gibt es, bedingt durch die sehr kurze Existenz von Folksonomies, bisher noch keine belastbaren Untersuchungen zu den diachronischen Veränderungen von Tag-Bezeichnungen. Es ist jedoch davon auszugehen, dass auch diese Veränderungen innerhalb des Internets weit schneller vonstatten gehen werden als natürlichsprachliche Veränderungen. Die notwendigen Daten werden daher in naher Zukunft, wenn auch für relativ kurze Zeitspannen, erhebbar sein, sodass eine vorausschauende Simulation dieser Prozesse bereits heute zweckmäßig erscheint.

Anhang

A. Anforderungsdefinition, Architektur und Entwurf

Dieses Kapitel beschreibt die Anforderungen, die benötigten Architekturen und eine Entwurfsvorstellung für die spätere Implementierung des im vorherigen Unterkapitel erläuterten Modells dieser Arbeit. In diesem Kapitel wird eine iterative Vorgehensweise deutlich: Ausgehend von der Modellbeschreibung leiten sich mehre Anforderungen ab, aus denen ein Entwurf erstellt wird, der die Einbindung verschiedener Architekturen fordert.

A.1 Anforderungsdefinition

Im folgenden Unterkapitel werden die Anforderungen für eine Implementierung des Modells dieser Arbeit hinsichtlich der Funktionalität aufgelistet. Der größte Teil dieser Anforderungen ergibt sich bereits aus der Beschreibung des Modells und wird in diesem Unterkapitel im Stile einer formalen Anforderungsdefinition wiedergegeben. Unter Berücksichtigung erster Entwurfsüberlegungen wird zwischen einer Einteilung in funktionale Anforderungen der Simulation, der Benutzerinteraktion und zwischen sonstigen Anforderungen unterschieden.

A.1.1 Simulation

Simulation

- Die Simulation muss agentenbasiert sein.
- Die Simulation muss eine Simulationsumgebung besitzen.
- Die Simulation muss deterministisch ablaufen. Eine festgelegte Aktion bei einem festgelegten Zustand muss immer zu einem identischen Folgezustand führen.
- Die Simulation muss die Entstehung eines gemeinsamen Lexikons der Agenten ermöglichen. Dies geschieht durch eine Optimierung der Fehlerindizes Avg1 und Avg2.

- Die Simulation kann aus beliebig vielen Simulationsumgebungen bestehen, in denen unabhängig voneinander parallel ablaufende Simulationsdurchläufe durchführbar sind.

Agent und Sprache

- Ein Agent muss aus einem neuronalen Netz bestehen, dessen Größe beliebig, aber bei der Initialisierung bekannt sein muss.
- Die neuronalen Netze der Agenten müssen sowohl den Feedforward Algorithmus als auch den Backpropagation of Error Algorithmus durchführen können.
- Ein Agent muss zwei Eingabeschnittstellen besitzen. Eine muss dabei der Sinneswahrnehmung des Hörens entsprechen, eine der Sinneswahrnehmung des Sehens.
- Ein Agent muss eine Ausgabeschnittstelle besitzen. Diese muss der Sinneswahrnehmung des Sprechens entsprechen.
- Ein Agent muss mit anderen Agenten interagieren können.

Simulationsumgebung

- Eine Simulationsumgebung muss eine beliebig große Menge visueller Szenen besitzen.
- Eine Simulationsumgebung kann eine Menge von Häufigkeiten visueller Szenen besitzen und diese den visuellen Szenen zuordnen.
- Eine Simulationsumgebung muss aus einer beliebig großen Gemeinschaft von Agenten bestehen.
- Eine Simulationsumgebung muss aus einem beliebig großen zweidimensionalen Raum bestehen, auf dem die Agenten positioniert werden können.

Kommunikation

- Innerhalb einer Simulationsumgebung muss ein Agent, basierend auf den im Modell deklarierten Algorithmen, aus der Gemeinschaft der Agenten auswählt werden können.
- Innerhalb einer Simulationsumgebung muss dem Sprecher, basierend auf den im Modell deklarierten Entfernungsalgorithmen und damit verbundenen Wahrscheinlichkeiten, ein Zuhörer ausgewählt werden können.
- Innerhalb einer Simulationsumgebung muss eine visuelle Szene, basierend auf den im Modell erläuterten Verfahren, ausgewählt werden können.
- Beide im Modell deklarierten Kommunikationsverfahren müssen durchführbar sein.

Störfaktoren

- Die Simulation muss eine Sterbewahrscheinlichkeit für Agenten unterstützen und ausgeschiedene Agenten durch neu initialisierte ersetzen können.
- Die Simulation muss ein Abschwächen der Verbindungsgewichtungen der neuronalen Netze unterstützen.
- Die Simulation muss ein Rauschen während der visuellen Wahrnehmung der Agenten implementieren.
- Die Simulation muss ein Rauschen während der Kommunikation zweier Agenten unterstützen.
- Die Simulation muss in der Lage sein, die Wortmuster der Agenten anhand der diskreten Kosinustransformation mit anschließender Quantisierung zu komprimieren und zu dekomprimieren.

Indikatoren

- Die Simulation muss in der Lage sein, den Indikator Avg1 sowohl für die gesamte Agentengemeinschaft, als auch für definierte Untergruppen zu berechnen.
- Die Simulation muss in der Lage sein, den Indikator Avg2 sowohl für die gesamte Agentengemeinschaft, als auch für definierte Untergruppen zu berechnen.
- Die Simulation muss in der Lage sein, den Indikator Understanding sowohl für die gesamte Agentengemeinschaft, als auch für definierte Untergruppen zu berechnen.
- Die Simulation muss in der Lage sein, ein schrittbasiertes Lexikon der Agenten zu berechnen und Veränderungen dessen über die Zeit zu zeigen.

Ereignisse

- Die Simulation muss eine beliebig große Menge von Ereignissen unterstützen, ihre Bedingungen auf Eintritt überprüfen und die Veränderungen zur Laufzeit umsetzen.

A.1.2 Benutzerinteraktion

Benutzergruppe

- Das Simulationstool muss von Personen mit Vorkenntnissen in Agentensimulation verwendet werden können.
- Das Simulationstool soll von Benutzern ohne Programmierkenntnisse verwendet werden können.
- Das Simulationstool soll von Benutzern ohne Erfahrung im Umgang mit neuronalen Netzen verwendet werden können.

Benutzerschnittstelle

- Die Benutzerschnittstelle muss grafisch umgesetzt werden.
- Die Benutzerschnittstelle muss einfach strukturiert gestaltet werden.
- Die Benutzerschnittstelle muss das Speichern von Simulationen ermöglichen.
- Die Benutzerschnittstelle muss das Laden von Simulationen ermöglichen.
- Die Benutzerschnittstelle muss das Erstellen von Simulationen ermöglichen.
- Die Benutzerschnittstelle muss das Editieren von Simulationen ermöglichen.
- Das Sammeln gezielter Simulationsdaten muss steuerbar sein.
- Die Indikatoren Avg1, Avg2 und Understanding müssen sowohl in Form eines Graphen, als auch als schrittgebundene Kreuztabelle dargestellt werden können.
- Die schrittbasierten Lexika der Agenten und Veränderungen zwischen ihnen müssen sichtbar gemacht werden können.
- Die Leistungen der Agenten einer Simulationsgemeinschaft in Bezug auf Wortbildung und Nachbilden einer visuellen Szene müssen schrittbasiert dargestellt werden können.

A.1.3 Sonstige Anforderungen

Programmiersprache

- Das Simulationstool muss in einer modernen Programmiersprache, vorzugsweise Java, geschrieben sein.

A.2 Architektur

Dieses Unterkapitel ist nahezu unverändert der vorangegangenen Arbeit [FK07] entnommen. Von einem erneuten Erstellen eines ähnlichen Textes wurde abgesehen, da er sich kaum von der bereits erarbeiteten und hier dargestellten Version unterscheiden würde.

Eine Softwarearchitektur beschreibt die grundlegenden Komponenten eines Softwaresystems und ihr Zusammenwirken miteinander. Helmut Balzert[21] nennt sie *„eine strukturierte oder hierarchische Anordnung der Systemkomponenten sowie Beschreibung ihrer Beziehungen“* (siehe [Bal01], S. 715).

In diesem Unterkapitel werden die fertigen Softwarepakete beschrieben, auf denen die Implementierung des Tools basiert. Die selbst entworfenen Teile des Programms werden im Unterkapitel A.3 genauer erläutert.

Abbildung 122 zeigt die für eine Implementierung intendierte Architektur im Schichtenstil. Die unterste Schicht stellt dabei die Applikationsschicht dar, während die oberste die Interaktionsschicht repräsentiert.

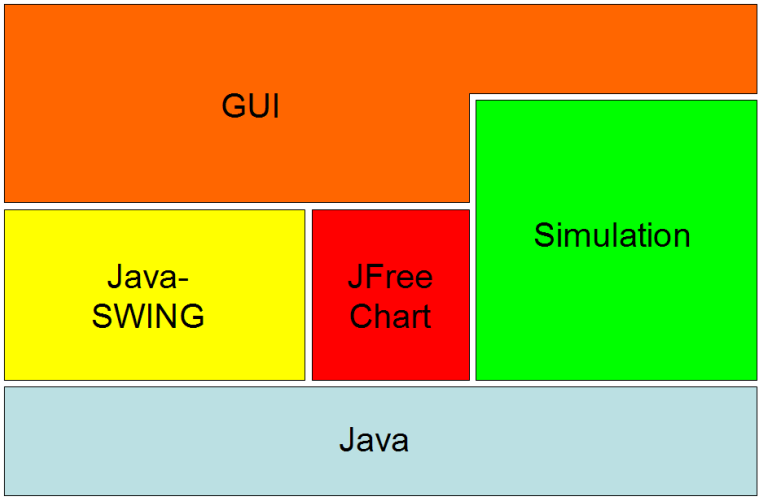


Abb. 122 Architektur

[21] Helmut Balzert ist Inhaber des Lehrstuhls für Software-Technik in der Fakultät für Elektrotechnik und Informationstechnik an der Ruhr-Universität Bochum.

A.2.1 Java

Java[22], eine objektorientierte Hochsprache, wurde als Programmiersprache des zu implementierenden Tools ausgewählt, da sie eine *plattformübergreifende Nutzung* des Programms gewährleistet und sich auch durch eine hohe Verbreitung auszeichnet.

A.2.2 Java-Swing

Um eine plattformübergreifende Nutzung von DiaLex zu garantieren, wurden für die Erstellung einer grafischen Benutzeroberfläche die Swing Grafikbibliotheken der Java Foundation Classes (JFC) ausgewählt. In Anlehnung an Guido Krüger (siehe [Kru00, S. 740-743]) bietet Swing, welches auf dem Abstract Windowing Toolkit (AWT) aufbaut, drei entscheidende Eigenschaften:

Leichtgewichtige Komponenten

Swing-Komponenten nutzen plattformspezifische GUI-Ressourcen nur noch in sehr eingeschränkter Form, d. h. abgesehen von Top-Level-Fenstern, Dialogen und grafischen Primitivoperationen werden alle GUI-Elemente von Swing eigenständig erzeugt. Neben der erheblichen Codevereinfachung und der plattformübergreifenden Vereinheitlichung von Aussehen und Bedienbarkeit, ist das Erstellen von komplexeren Dialogelementen wie Bäumen, Tabellen und Tooltips einer der Hauptvorteile der Swing-Bibliotheken. Abbildung 123 gibt eine Übersicht über die in Swing vorhandenen Komponenten. Ein *Container* nimmt Swing-Komponenten auf und setzt sie mit Hilfe eines Layoutmanagers an die richtige Position.

Austauschbares „Look-and-Feel“

Eine der weiteren wichtigen Eigenschaften von Swing, welche in DiaLex jedoch nicht genutzt wird, ist die Möglichkeit das Aussehen und die Bedienung („Look-and-Feel“) einer Anwendung zur Laufzeit umzuschalten. Mögliche Varianten für ein „Look-and-Feel“ sind beispielsweise Swing, Motif oder Windows.

[22] http://java.sun.com

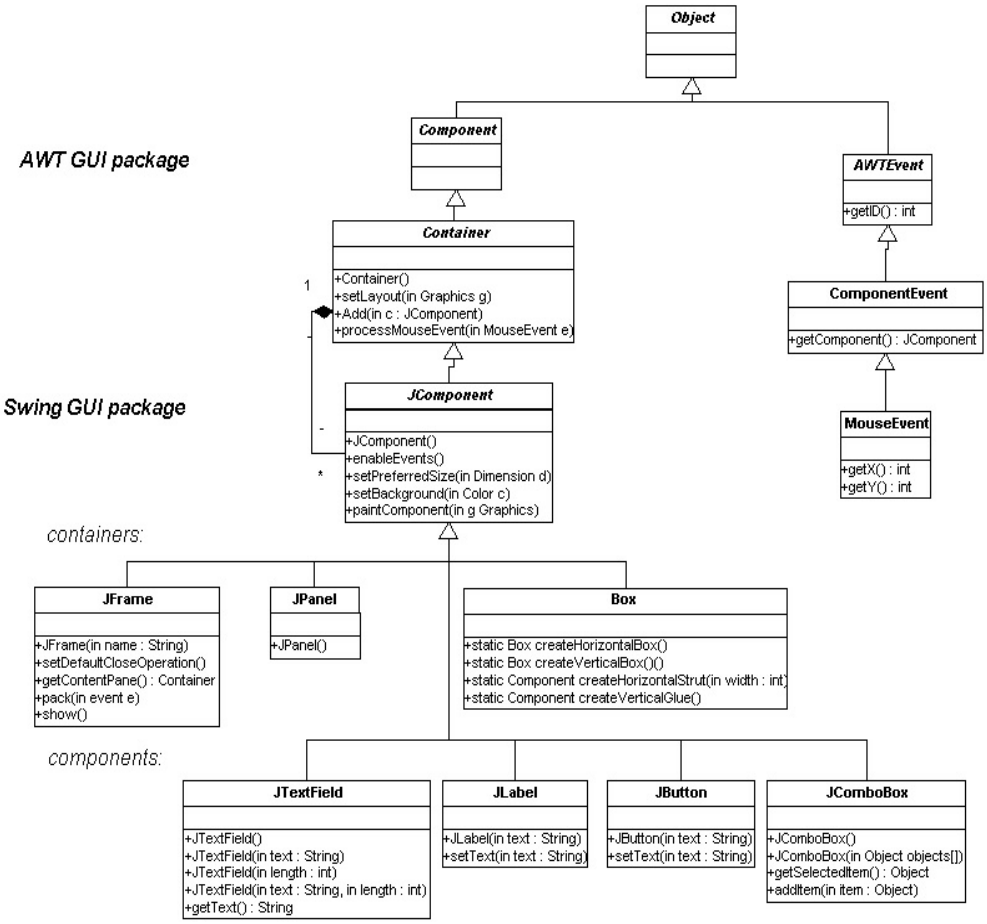


Abb. 123 Modell der Klasse JComponent
Quelle: Parallel Computing Laboratory, Vrije Universiteit Brussel[23]

Model-View-Controller-Prinzip

Unter dem Model-View-Controller-Prinzip versteht man das Konzept, drei verschiedene Bestandteile eines grafischen Elements zu unterscheiden:

- das *Modell*, welches den Zustand und die Daten eines Dialogelements abspeichert,
- der *View*, welcher für die grafische Darstellung der Komponente verantwortlich ist und
- der *Controller*, der Tastatur- und Mausereignisse empfängt und die erforderlichen Veränderungen von Model und View bewirkt.

[23] http://parallel.vub.ac.be/documentation/java/swing/Swing_GUI_Class_model.jpg

Die bei Swing-Dialogelementen genutzte Variante *Model-Delegate-Prinzip* fasst aus Komplexitätsgründen die Funktionalität von View und Controller in einer Benutzerschnittstelle *Delegate* zusammen.

Neben den oben beschriebenen positiven Eigenschaften von Swing besteht ein erheblicher Nachteil: Swing-Anwendungen sind wegen ihrer plattformunabhängigen, selbst erstellenden Komponenten ressourcenhungrig und verlangsamen bei nicht ausreichender Rechnerleistung eine Anwendung. Aufgrund der immer weiter steigenden Rechnerleistungen ist dieser Nachteil jedoch vernachlässigbar.

A.2.3 JFreeChart

Bei JFreeChart[24] handelt es sich um eine frei verfügbare Java Diagramm Bibliothek, die, wie von den Entwicklern beschrieben, folgende besonderen Merkmale besitzt:

- eine konsistente und gut dokumentierte Programmierschnittstelle (API), welche eine große Menge von Diagrammtypen unterstützt,
- ein einfaches, flexibles Design, welches einfach zu erweitern ist und auf server- und clientseitige Anwendungen zielt,
- eine große Menge von Ausgabeformaten für die erzeugten Diagramme: neben den gängigen Bilddateiformaten (JPG, PNG) und Vektorgrafikformaten (PDF, EPS, SVG) auch Java-Swing Komponenten, welche für eine spätere Implementierung wichtig sind,
- es handelt sich um eine freie „open-source“ Software, welche nach der GNU Lesser General Public Licence (LGPL)[25] vertrieben werden kann.

Für die in Abschnitt 3.5.6 beschriebene Darstellung der Fehlerindikatoren Avg1, Avg2 und Understanding ist das Packet **`org.jfree.chart`** der JFreeChart-Bibliothek von besonderem Interesse. Es werden hier Objekte der Klasse **`JFreeChart`**, welche die eigentlichen Diagramme repräsentieren, und Instanzen der Klasse **`ChartPanel`**, Java-Swing Komponenten, für die Anzeige eines JFreeChart-Objekts, benötigt[26]. JFreeChart benötigt das JDK 1.3 oder höher.

[24] http://www.jfree.org/jfreechart/
[25] http://www.gnu.org/licenses/lgpl.html
[26] Eine komplette Übersicht der JFreeChart API gibt es unter http://www.jfree.org/jfreechart/api/javadoc/index.html

A.3 Entwurf

Im diesem Unterkapitel werden die im Rahmen dieser Arbeit selbst entworfenen Teile der Architektur genauer erläutert. Zuerst liegt das Augenmerk auf der Komponente *Simulation*, die direkt und nur auf Java aufsetzt und in der die eigentlichen Simulationen ablaufen. Der zweite Abschnitt beschreibt den Entwurf der *grafischen Benutzeroberfläche*, die die Schnittstelle zwischen den Simulationsklassen und dem Benutzer darstellt. Im letzten Abschnitt wird das Zusammenwirken aller Entwurfsteile, als Übergang zur im nächsten Kapitel beschriebenen Implementierung, kurz erläutert.

A.3.1 Simulation

Der Entwurf der Simulation leitet sich nahezu ausschließlich aus dem Modell dieser Arbeit und den Grunddatentypen von Java ab. Es existiert eine Klasse *Simulation*, die sowohl als Schnittstelle zur grafischen Benutzeroberfläche als auch zum Halten und Steuern mehrerer Simulationsumgebungen bestimmt ist. Es soll gewährleistet sein, dass ein Speichern oder Laden dieser Klasse mit allen in ihr enthaltenen Attributen ausreicht, um eine Simulation zu sichern bzw. wiederherzustellen.

Innerhalb der Klasse Simulation befinden sich beliebig viele Instanzen der Klasse *Environment*, wobei jede für einer der parallel ablaufenden Simulationsumgebungen steht. Die Methoden, die für den Ablauf einer Simulation in einer Umgebung verantwortlich sind und die in der Modellbeschreibung in Unterkapitel 3.5 beschrieben werden, befinden sich in der Klasse Environment. Zusätzlich dazu befinden sich in der Klasse Simulation die Klasse *EventManager*, die für die Ereignisverwaltung der Simulationen zuständig ist. Dabei existieren zwei verschiedene Arten von Ereignissen: Die erste Art kodiert Auswirkungen auf eine Simulationsumgebung und ändert dort Parameter zur Laufzeit. Die zweite definiert welche Daten zu welchen Zeitpunkten während eines Simulationslaufs gesammelt werden. Das Sammeln eines Datums wird ähnlich deklarierbar sein wie die bereits in der Modellbeschreibung erläuterten Ereignisse. Die so erhaltenen Datensätze werden in der Klasse *Database*, die ebenfalls Bestandteil der Simulationsklasse ist, gehalten. Sie stellt die Funktionalitäten einer einfachen Datenbank zur Verfügung.

Die Klasse *Agent* als Bestandteil einer Umgebung beinhaltet neben den benötigten und in Unterkapitel 3.5 beschriebenen Algorithmen die Bestandteile seines neuronalen Netzes, welches in Klassen zweier weiterer Typen aufgeteilt wird. So besteht es aus einer Menge *n* von Instanzen des Typs *Layer*, die eine Schicht des neuronalen Netzes repräsentiert und einer Menge *n-1* von Instanzen der Klasse

Connection, die Verbindungen der neuronalen Netze, die stets zwei Schichten miteinander verknüpfen. Eine Schicht besteht folglich aus einer Menge von Neuronen, Instanzen der Klasse *Soma*, die nur aus primitiven Datentypen zum Halten reeller Werte besteht, während eine Verbindung aus einer Menge von Gewichtungen besteht. Diese Klasse *Axon* besteht ebenfalls nur aus primitiven Datentypen, sodass ab diesem Punkt keine weitere Abstraktion mehr notwendig erscheint.

A.3.2 Grafische Benutzeroberfläche

Der Entwurf einer grafischen Benutzeroberfläche sieht vor, dass das zu erstellende Werkzeug die in der Anforderungsdefinition aufgelisteten Benutzerinteraktionen ermöglicht. Zudem soll durch eine klare strukturelle Aufteilung der Oberfläche in verschiedene Funktionalitätsbereiche eine Erweiterungsmöglichkeit um zusätzliche Interaktionen in zukünftigen Versionen garantiert werden.

In erster Instanz teilt sich der Entwurf der Benutzerschnittstelle des Werkzeugs in ein *Hauptfenster* und ein *Analysefenster* auf, welches vom Hauptfenster aus anzusteuern ist. Innerhalb des Analysefensters lässt sich neben einigen kleineren Fenstern, die an dieser Stelle im einzelnen nicht erläutert werden, ebenfalls das *Agentenfenster* öffnen, in dem alle Agenten einer Simulationsumgebung separat zu betrachten und die Lexika der Gemeinschaft gezeigt werden.

Hauptfenster

Der Entwurf des Hauptfensters ist in Abbildung 124 dargestellt, jede einzelne Komponente bietet dem Benutzer eine Menge von zusammenhängenden Interaktionsmöglichkeiten:

Innerhalb der Menüleiste gibt es ein Dateimenü, welches dem Benutzer die Möglichkeit bietet, eine Simulation zu erstellen, zu laden oder zu speichern und ein Informationsmenü, durch welches Informationen über die aktuelle Version des Werkzeugs abgerufen werden können. Unterhalb der Menüleiste befindet sich der Konfigurations- und Steuerbereich des Werkzeugs, welcher dem Anwender die Möglichkeit bietet, sowohl konfigurierende Parameter einzugeben als auch einen Simulationsdurchlauf zu steuern. Zu den konfigurierenden Parametern, welche für alle erstellten Simulationsumgebungen gelten, zählen die Eingabe der Anzahl von Simulationsschritten und das Einlesen einer Menge von Eingabemustern und die mit ihnen verbundenen Häufigkeiten. Innerhalb dieses Bereiches wird ebenfalls angezeigt, ob eine Menge von Eingabemustern und eventuell Häufigkeiten bereits eingelesen wurden und es wird zusätzlich die Möglichkeit geben, die eingelesenen Daten grafisch zu betrachten. Ebenfalls befindet sich hier eine Schaltfläche, die das Analysefenster aufruft. Die Steuerung eines Simulationsdurchlaufs, also das

Starten, Pausieren und Beenden einer Simulation, wird durch drei verschiedene Schaltflächen geregelt.

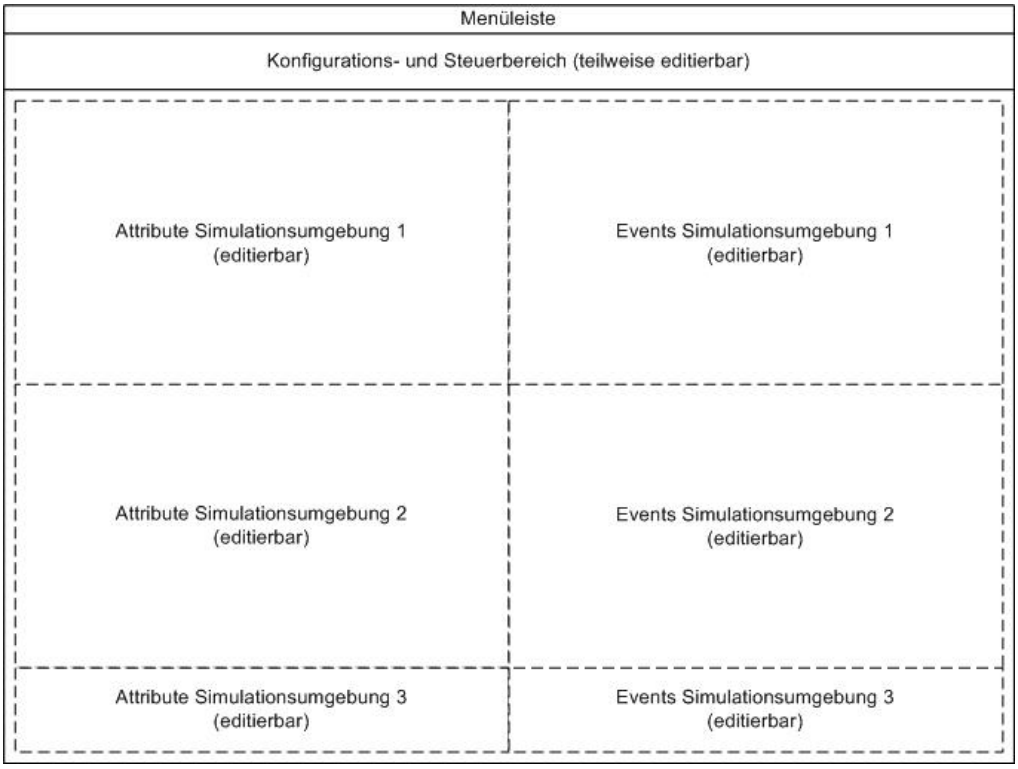


Abb. 124 Entwurf des Hauptfensters

Des Weiteren sieht der Entwurf der grafischen Benutzeroberfläche vor, die nach dem Erstellen erzeugten Simulationsumgebungen innerhalb eines scrollbaren Rahmens einzeln darzustellen. Die Darstellung umfasst dabei im linken Teil eine Auflistung aller Attribute einer Simulationsumgebung, sodass Änderungen an den Attributen zur Laufzeit des Simulationstools vorgenommen werden können. Im rechten Teil befindet sich der für die Ereignisbehandlung zuständige Bereich, in dem neue Events erzeugt und aktuell existierende und bereits ausgelöste Events aufgelistet werden.

Analysefenster

Innerhalb des Analysefensters werden alle gesammelten Datenreihen der Simulationsdurchläufe innerhalb eines scrollbaren Rahmens aufgelistet, wie es in Abbildung 125 gezeigt wird. Da je nach Datentyp andere Formen der Veranschaulichung der einzelnen Daten in Frage kommen, variiert die jeweilige Darstellungsform leicht. Teilweise sind die gesammelten Daten direkt ersichtlich, teilweise können oder müssen zum Betrachten der Simulationsdaten neue, kleinere Fenster geöffnet werden, beispielsweise mit Simulationsgraphen oder

Kreuztabellen. Da diese jedoch relativ simpel gehalten sind, werden sie hier nicht separat aufgeführt. Das einzige innerhalb des Analysefensters aufrufbare komplexere Fenster ist das Agentenfenster, das im nächsten Abschnitt erläutert wird.

Abb. 125 Entwurf des Analysefensters

Agentenfenster

Der Entwurf des Agentenfensters, welches zur genauen Betrachtung der Agenten einer Simulationsumgebung dient, teilt das Fenster, wie in Abbildung 126 gezeigt, in zwei Bereiche auf.

Innerhalb des oberen Auswahlbereichs kann der Benutzer zum einen das Eingabemuster, zu welchem er die Wörter und Abzeichnungen der Agenten betrachten möchte, auswählen. Neben dem Index wird eine graphische Darstellung des Eingabemusters als Information gegeben. Zum anderen kann der Benutzer in diesem Bereich zwischen verschiedenen Darstellungsformen der Wörter und Abzeichnungen der Agenten wählen.

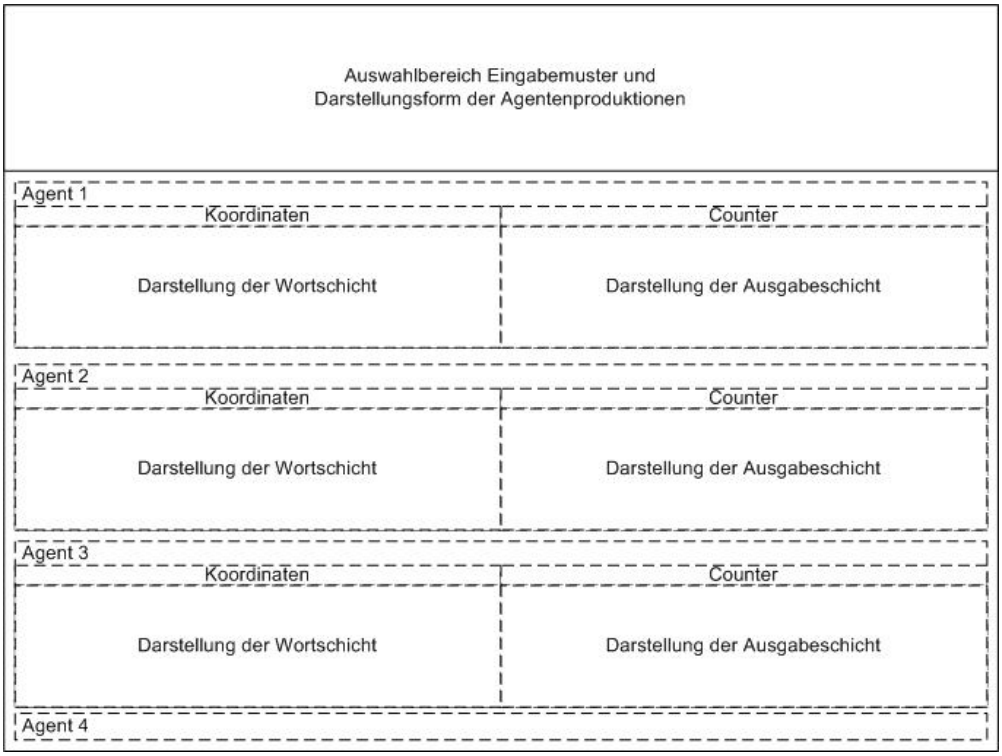


Abb. 126 Entwurf des Agentenfensters

Zur Veranschaulichung der Wörter kann der Benutzer zwischen

- den exakten, reellen Aktivierungswerten
- den auf zwei Nachkommastellen gerundeten Aktivierungswerten
- einer Abbildung: Aktivierungswert zu Buchstabenfolge
- einer grafischen Darstellung der Aktivierungswerte
- einer grafischen Darstellung der Aktivierungswerte nach der diskreten Kosinustransformation

der einzelnen Neuronen der Wortschicht wählen. Zur Darstellung der Deduktion eines Musters kann sich der Nutzer zwischen

- den exakten, reellen Aktivierungswerten
- den auf zwei Nachkommastellen gerundeten Aktivierungswerten
- einer grafischen Darstellung der Aktivierungswerte

der Neuronen der Ausgabeschicht entscheiden. Die Auflistung der Agenten erfolgt innerhalb eines scrollbaren Rahmens. Je nach ausgewählter Darstellungsform werden für jeden Agenten sein Wort und seine Abzeichnung des ausgewählten Eingabemusters ausgegeben.

A.3.3 Gesamtentwurf

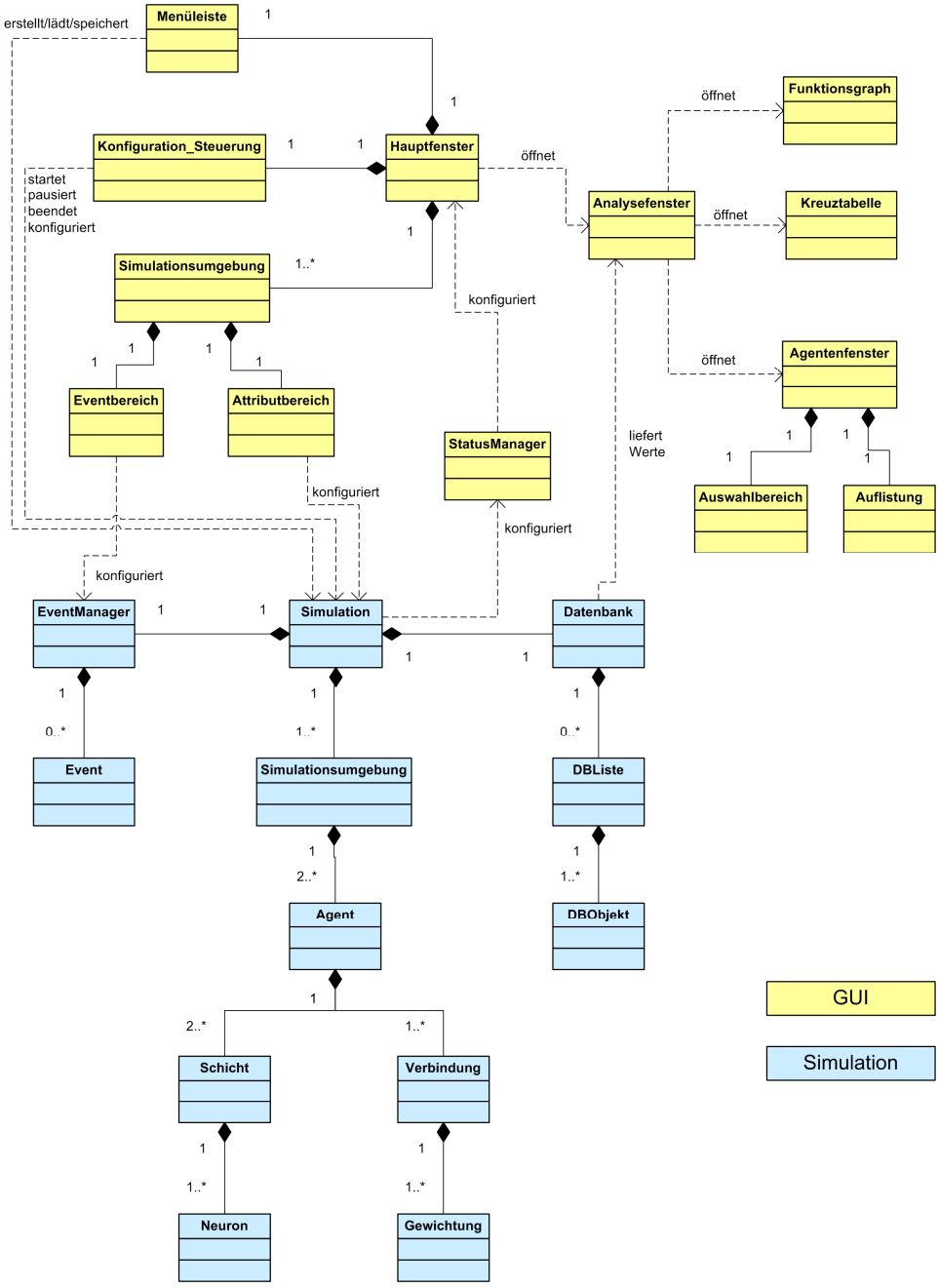


Abb. 127 Gesamtentwurf

Abbildung 127 zeigt das Zusammenwirken der beschriebenen Komponenten des Gesamtentwurfs. Es wird nochmals deutlich, dass die Klasse Simulation die zentrale Schnittstelle zwischen der grafischen Benutzeroberfläche und dem Simulationsteil des Tools ist.

B. DiaLex – Die Implementierung

In diesem Kapitel wird die Implementierung des im vorherigen Unterkapitel erläuterten Entwurfs unter Berücksichtigung der angestrebten Architektur beschrieben. Die strikte Trennung zwischen Simulation und grafischer Benutzeroberfläche des Entwurfs teilt auch dieses Kapitel in zwei Teile.

Es ist erkennbar, dass die im Gesamtentwurf verwendeten Klassen durch die in DiaLex tatsächlich genutzten Klassennamen ersetzt wurden. Die Struktur des Entwurfs wurde in der Realisierung gänzlich übernommen. Auch die in Unterkapitel A.2 beschriebene Architektur findet sich in der gesamten Implementierung wieder. Die grafische Benutzeroberfläche dient als auslösende, steuernde und konfigurierende Schicht für die Simulation, deren zentrale Schnittstelle die Klasse **`Simulation`** ist.

Die Implementierung erfolgte in einer Eclipse[27] Umgebung unter Nutzung eines SVN-Servers an der Universität Koblenz-Landau. Erste Versuche zur Realisierung von neuronalen Netzen und die darauf aufbauende Implementierung des Modells von Hutchins und Hazlehurst im Rahmen der vorangegangenen Diplomarbeit (vgl. [FK07]) bilden die Basis für die hier dargestellte Implementierung des Modells von DiaLex. Da Teile der in diesem Kapitel aufgeführten Klassen bereits Bestandteile der aus [FK07] hervorgegangenen Implementierung LexLearn waren und für die Ziele dieser Arbeit erweitert wurden, befinden sich in der Implementierungsbeschreibung Textpassagen, die aus [FK07] nicht oder nur leicht verändert übernommen wurden. Dies trifft im Simulationsteil besonders auf die Klasse **`Agent`** zu, die die elementaren Funktionen eines neuronalen Netzes implementiert. Da sie jedoch zum Verständnis dieser Arbeit notwendig sind, kann auf eine detaillierte Beschreibung an dieser Stelle nicht verzichtet werden.

Die Beschreibung des Aufbaus des Simulationstools DiaLex erfolgt sowohl textuell als auch mit Hilfe von UML-Diagrammen und elementaren Auszügen aus dem Quellcode.

[27] Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. URL: http://www.eclipse.org

B.1 Simulation

Die Implementierung der Simulation unterteilt sich, wie bereits im Entwurf der Implementierung ersichtlich, in die drei Hauptebenen *Agent*, *Environment* und *Simulation*.

B.1.1 Die Klasse Agent

Die Klasse **`Agent`** beinhaltet sämtliche dem Agenten zugerechneten Attribute und Methoden. Sie besteht nicht nur aus einem neuronalen Netz, dem Hirn eines Agenten, sondern ebenfalls aus einer Menge von Methoden, die für die Evolution des Netzes während eines Simulationsdurchlaufs benötigt werden. So enthält sie Methoden für eine zufällige initiale Belegung des neuronalen Netzes, den Forward Pass Algorithmus, die Fehlerberechnung für das Backpropagation of Error Verfahren, eine Methode, die anhand dieser Fehlerberechnung die Kantengewichtungen verändert, eine Methode zur Abschwächung der Kantengewichtungen als eine Form des Vergessens und eine Reihe kleinerer, meist für die Ein- und Ausgabe benötigter Methoden, die jedoch im Laufe dieses Unterkapitels noch genauer erläutert werden.

Der Aufbau des neuronalen Netzes

Das neuronale Netz wird in der Klasse **`Agent`** in zwei *Arrays* gehalten. Zum einen existiert ein Array aus Instanzen der Klasse **`Layer`** namens **`layers`** der variablen Länge n, welches, beginnend bei der Eingabeschicht an Position *0*, über die verborgenen Schichten von Position *1* bis *n-1*, bis hin zur Ausgabeschicht, die sich an Position *n* des Arrays befindet, die *Schichten* des neuronalen Netzes beinhaltet. Die Verbindungen zwischen den einzelnen Schichten werden in einem weiteren Array namens **`con`** gehalten, das aus Instanzen der Klasse **`Connection`** besteht. Beide Klassen, **`Layer`** und **`Connection,`** mit ihren zugehörigen Unterklassen werden im Laufe dieses Abschnitts an späterer Stelle detailliert beschrieben.

Des Weiteren existieren zwei Arrays des Typs **`int`**, in welchen die Längen und Breiten der einzelnen Schichten aufgelistet werden. Folglich heißen sie **`layerlengths`** und **`layerwidths`**. Eine **`String`** Variable namens **`name`** beinhaltet den Namen des Agenten, die Variablen **`learningRate`** und **`momentum`** des Typs **`double`** halten die Lernrate und das Momentum des Netzes. Eine **`int`** Variable mit dem Namen **`wordPosition`** bestimmt die Position der Wortschicht im neuronalen Netz. Hinzu kommen zwei Zählvariablen **`speakerCount`** und **`listenerCount`** des Typs **`int`**, die die Anzahl der Kommunikationsversuche eines jeweiligen Agenten in der Rolle des Sprechers

bzw. des Zuhörers mitzählen. Somit kann bei einer genaueren Untersuchung eines Simulationsdurchlaufs beispielsweise ein ungelernter Agent erkannt werden.

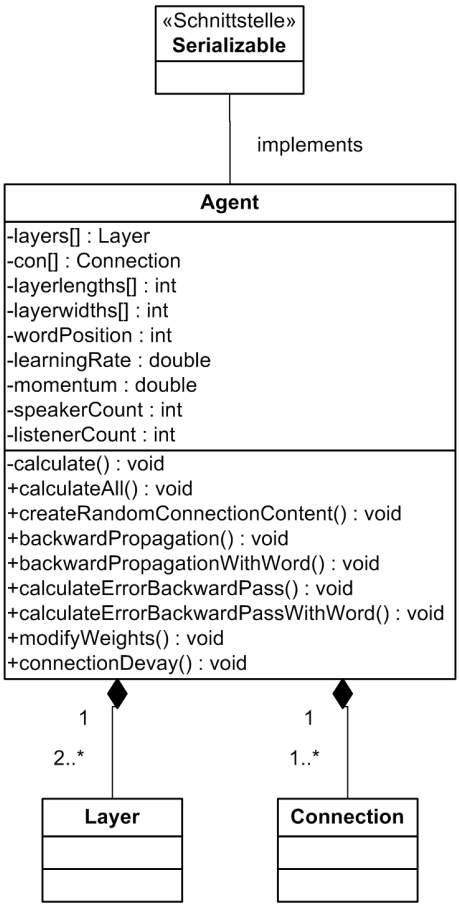


Abb. 128 Klassendiagramm der Klasse Agent

Die Klasse Layer

Eine Schicht des neuronalen Netzes wird durch die Klasse **`Layer`** repräsentiert. Sie besteht aus einer Zeichenkette **`name`** mit dem Namen der Schicht, einem zweidimensionalen Array **`field`** des Typs **`Soma`**, welches die Neuronenschicht darstellt, einem Biasneuron **`bias`** des Typs **`SomaBias`** und einem zweidimensionalen Array des Typs **`double`** der Verbindungen des Biasneurons zu den einzelnen Neuronen der Schicht, welches **`biasCon`** heißt.

Darüber hinaus enthält die Klasse **`Layer`** einige kleinere Methoden, von denen die wichtigsten im Folgenden aufgelistet und kurz beschrieben werden:

`void createRandomBiasConContent()`
In dieser Methode werden die Verbindungsgewichtungen vom Biasneuron zu den Neuronen der Schicht mit reellen Zufallswerten zwischen -0.5 und +0.5 belegt.

`double[][] getValues()`
`double[][] getActivations()`
`double[][] getErrors()`
Diese drei Methoden arbeiten ähnlich, **`getValues()`** liefert ein zweidimensionales Array des Typs double, welches die **`value`** Werte der Soma enthält, aus denen das **`field`** Array besteht. **`getActivations()`** liefert entsprechend die Aktivierungen der **`Soma`** und **`getErrors()`** die Fehlerwerte (vgl. Unterkapitel 3.3).

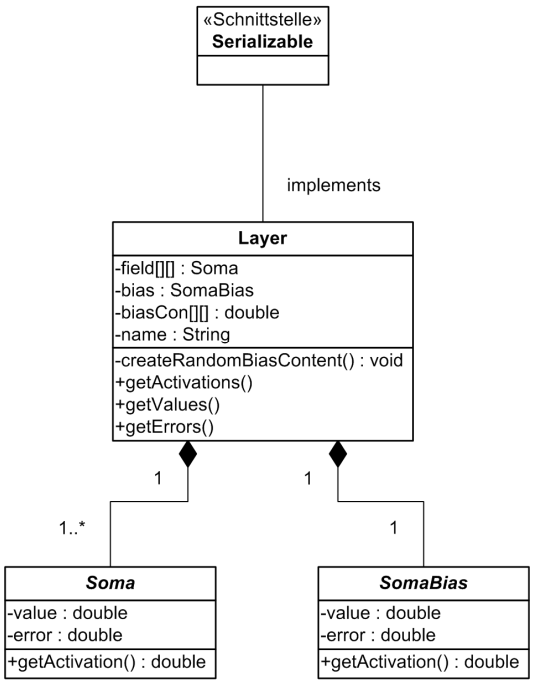


Abb. 129 Klassendiagramm der Klasse Layer

Die Soma-Klassen

Die Klasse **`Soma`** und ihre Derivate **`SomaBias`**, **`SomaInput`**, **`SomaHidden`** und **`SomaOutput`** repräsentieren die Zellkörper der Neuronen eines neuronalen Netzes. **`Soma`** steht dabei für ein Neuron und beinhaltet lediglich zwei Variablen des Typs **`double`**, **`value`**, den aktuellen Wert der aufsummierten Größen der eingehenden Verbindungen des Neurons und **`error`**, im dem der Fehlerwert des

Neurons gespeichert wird (vgl. Abschnitt 3.3.4). Neben den obligatorischen Methoden

- **`getValue()`**
- **`setValue(double value)`**
- **`getError()`**
- **`setError(double error)`**

enthält die Klasse nur eine weitere Methode namens

- **`getActivation()`,**

die jedoch als **`abstract`** deklariert ist und somit in den abgeleiteten Klassen definiert werden muss.

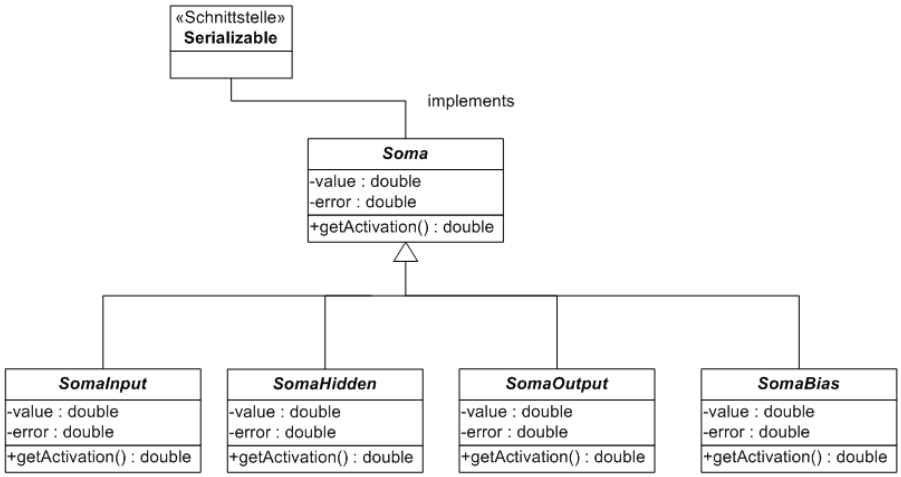


Abb. 130 Klassendiagramm der Klasse Soma

Die notwendige Unterscheidung der einzelnen **`Soma`**-Varianten liegt darin begründet, dass die Aktivierungsfunktion eines Soma von seiner Position im neuronalen Netz abhängig ist. So wird die abstrakte Methode **`getActivation()`** in jeder abgeleiteten Klasse verschieden implementiert:

- **`SomaInput`**
  Die Klasse **`SomaInput`** symbolisiert ein Neuron der Eingabeschicht. Laut Definition der neuronalen Netze (vgl. Unterkapitel 3.3) entspricht die Aktivierungsfunktion hier der Identitätsfunktion, d. h. die Eingabewerte werden unverändert weitergereicht. Folglich liefert **`getActivation()`** hier den Inhalt der Variable **`value`**.

- **`SomaHidden und SomaOutput`**
  Die Klassen **`SomaHidden`** und **`SomaOutput`** stehen für Neuronen der verborgenen und der Ausgabeschicht. Ihre Aktivierungen entsprechen der dem Rückgabewert der sigmoiden Funktion an der Stelle **`value`**. Inhaltlich sind diese beiden Klassen identisch, ihre Unterscheidung dient lediglich der Übersichtlichkeit.

- **`SomaBias`**
  Das Biasneuron gibt stets den Wert 1 zurück. Folglich ist 1 die Belegung des Werts **`value`** und auch der Rückgabewert der Methode **`getActivation()`**.

Die Klasse Connection

Die Verbindungen innerhalb der neuronalen Netze werden durch die Klasse **`Connection`** dargestellt. Hierbei steht *eine* Instanz dieser Klasse für alle Verbindungen zwischen *zwei* Schichten.

Neben dem String **`name`**, der den Namen der **`Connection`** enthält, besteht die Klasse **`Connection`** aus einem vierdimensionalen Array namens **`field`** aus Instanzen der Klasse **`Axon`**, in denen die Eigenschaften der Verbindung gehalten werden. Hierbei stehen die ersten beiden Dimensionen des Arrays für die *Startposition der Verbindung* in der vorderen zweidimensionalen Schicht, während die letzten beiden Dimensionen auf das Ziel-Soma in der hinteren Schicht deuten und entsprechend die *Koordinaten des Ziel-Somas* repräsentieren. Die Verbindung an Position *[a][b][c][d]* verbindet folglich das Soma an Position *[a][b]* der vorderen Schicht mit dem Soma an Position *[c][d]* der hinteren Schicht. Abbildung 132 zeigt dies an dem Beispiel zweier Schichten eines neuronalen Netzes. Die wichtigsten Methoden der Klasse **`Connection`** werden im Folgenden erläutert:

- **`void createRandomContent()`**
  Diese Methode belegt die Verbindungsgewichtungen der **`Connection`** mit Zufallswerten zwischen -0.5 und +0.5 und wird beispielsweise beim Initialisieren eines Agenten ausgeführt.

- **`void decay(double maxDecay)`**
  Diese Methode verändert die Verbindungsgewichtungen zufällig bis zu einer maximalen Abweichung, die in der Variable **`maxDecay`** übergeben wird. Die genaue Funktionsweise wird im Unterabschnitt „*Das Abschwächen der Verbindungsgewichtungen als Störfaktor*“ dieses Abschnitts erklärt.

- **`void show()`**
  Diese Methode gibt die aktuelle Belegung der Verbindungsgewichtungen auf der Konsole aus. Auch wenn sie in DiaLex selbst nicht genutzt wird, ist sie für eine genauere Analyse der Verbindungsgewichtungen unerlässlich.

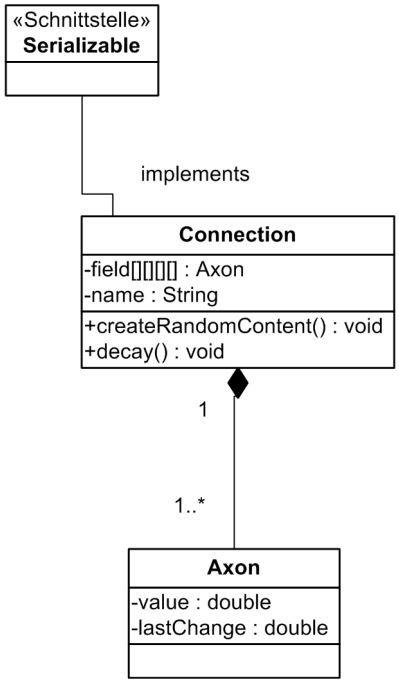


Abb. 131 Klassendiagramm der Klasse Connection

Die Klasse Axon

Das vierdimensionale Array **`field`** der Klasse **`Connection`** besteht aus Instanzen der Klasse **`Axon`**, die die Axone, die Fortsätze der Nervenzellen, als natürliche Vorbilder haben. Eine Instanz der Klasse **`Axon`** symbolisiert dort genau eine Verbindung von einem **`Soma`** einer Schicht *n* zu einem **`Soma`** der Schicht *n+1*.

Es existieren zwei Klassenvariablen, **`value`** vom Typ **`double`**, die den Wert der Verbindungsgewichtung hält, und **`lastChange`**, ebenfalls vom Typ **`double`**, die die letzte Änderung der Verbindungsgewichtung speichert. Zum Abschluss dieses Unterabschnitts zeigt Abbildung 132 die Koordinatenansicht eines einfachen zweidimensionalen neuronalen Netzes, das die Idee der vierdimensionalen Verbindungsarrays verdeutlicht.

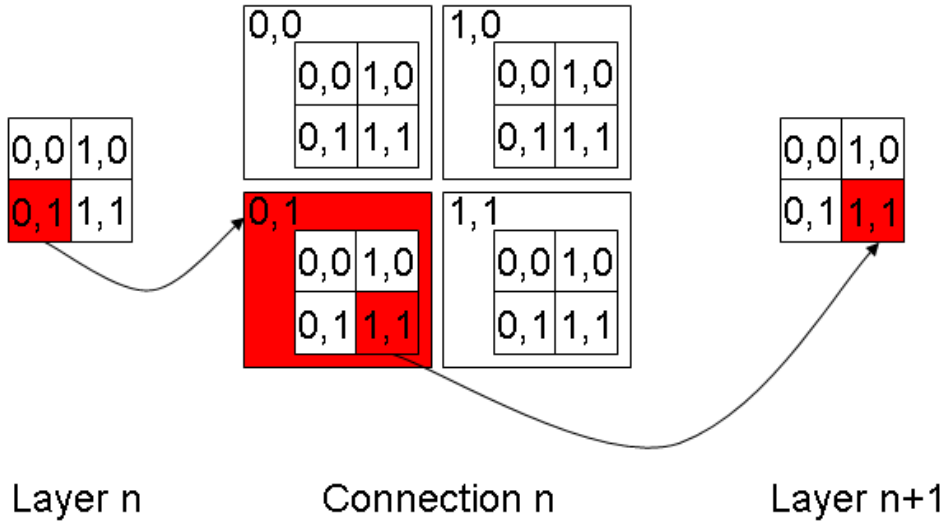


Abb. 132 Koordinatenansicht zweier Schichten eines neuralen Netzes

Die Instanziierung

Die Instanziierung der Klasse **`Agent`** erfolgt durch die Übergabe der Variablen

- **`double learningRate`**
- **`double momentum`**
- **`int[] layerlengths`**
- **`int[] layerwidths`**
- **`int wordPosition`**

innerhalb des Konstruktors. Während die Variablen **`learningRate`**, **`momentum`** und **`wordPosition`** lediglich lokal gespeichert werden, wird die komplette Netzstruktur aus **`layerlenghts`** und **`layerwidths`** abgeleitet.

Die Größen der einzelnen Schichten sind direkt aus den beiden Arrays ersichtlich. Daher wird das Array **`layers`** direkt mit in der Größe angepassten Instanzen der Klasse **`Layer`** gefüllt. Hierbei wird die Position der Schicht berücksichtigt, d. h. die Eingabeschicht besteht aus Objekten der Klasse **`SomaInput`**, die verborgenen Schichten aus Instanzen der Klasse **`SomaHidden`** und die Ausgabeschicht aus Objekten der Klasse **`SomaOutput`**. Der folgende Quellcodeauszug führt die beschriebene Instanziierung aus.

```
for (int i = 0; i<laylengths.length; i++)
{
      if (i==0)
      {
            SomaInput[][] array = new SomaInput[laylengths[i]]
                                                [laywidths[i]];
            for (int a=0;a<laylengths[i];a++)
            {
                  for(int b=0; b<laywidths[i];b++)
                  {
                        array[a][b] = new SomaInput();
                  }
            }
            layers[i] = new Layer(array,"Layer "+i,
                              laylengths[i+1],laywidths[i+1]);
      }
      else{
            if (i== laylengths.length-1)
            {
                  SomaOutput[][] array = new SomaOutput
                                    [laylengths[i]][laywidths[i]];
                  for (int a=0;a<laylengths[i];a++)
                  {
                        for(int b=0; b<laywidths[i];b++)
                        {
                              array[a][b] = new SomaOutput();
                        }
                  }
                  layers[i] = new Layer(array,"Layer "+i);
            }
            else{
                  SomaHidden[][] array = new SomaHidden
                                    [laylengths[i]][laywidths[i]];
                  for (int a=0;a<laylengths[i];a++)
                  {
                        for(int b=0; b<laywidths[i];b++){
                              array[a][b] = new SomaHidden();
                        }
                  }
                  layers[i] = new Layer(array,"Layer "+i,
                                laylengths[i+1],laywidths[i+1]);
            }
      }
}
```

Die Verbindungen werden ebenfalls direkt bei der Instanziierung erzeugt. Die Struktur des vierdimensionalen Arrays richtet sich, wie bereits im vorherigen Abschnitt ausgeführt, ebenfalls nach **`layerlengths`** und **`layerwidths`**.

```
…
for (int j = 0; j<layers.length-1;j++)
{
      con[j] = new Connection(new Axon [laylengths[j]]
             [laywidths[j]][laylengths[j+1]][laywidths[j+1]]);
}
…
```

Soll eine zufällige Belegung der Verbindungsgewichtungen gewährleistet werden, so wird nach der Instanziierung die Methode

- **`void createRandomConnectionContent()`**

ausgeführt, die iterativ alle Instanzen der Klasse **`Connection`** im Array **`con`** durchläuft und dort lokal die Methode **`createRandomContent()`** aufruft.

Die Forward Pass Berechnung

Die Forward Pass Berechnung des neuronalen Netzes wird durch folgende fünf Methoden bewerkstelligt:

- **`void calculate(Layer in, Layer out, Connection con)`**
  Die Methode **`calculate()`** ist die grundlegende Methode der Propagierung von Werten innerhalb der Feedforward Netze. Sie bestimmt anhand der Aktivierungswerte der Schicht **`in`** und der Verbindung **`con`**, die **`in`** und **`out`** verbindet, die Netzeingaben der Ausgabeschicht **`out`**.
  Dabei wird das Produkt aller Gewichtungen der in ein Soma eingehenden Verbindungen mit den Aktivierungswerten der Soma, die die Startpunkte der jeweiligen Verbindungen darstellen, aufsummiert.

- **`void calculateAll()`**
  Diese Methode ruft iterativ, beginnend bei der Eingabeschicht bis hin zur Ausgabeschicht, die Methode **`calculate()`** auf und führt so eine Berechnung aller Werte innerhalb eines kompletten Feedforward Netzes durch.

- **`void calculateAll (double[][] in)`**
  Diese Methode setzt zuerst das Array **`in`** als Eingabewerte der Eingangsschicht und führt danach **`calculateAll()`** aus.

- **`void calculateFromUntil (int start, int end)`**
  Mit **`calculateFromUntil()`** ist eine partielle Berechnung des Forward Pass Algorithmus möglich. Hierbei steht start für die Schicht, in der die Berechnung startet, und end für die letzte zu berechnende Schicht.

- **`void calculateFromUntil (double[][] in, int start, int end)`**
  Analog zu **`calculateAll()`** existiert auch bei **`calculateFromUntil()`** eine Methode, die zuerst die Eingabeschicht neu belegt und danach die Berechnung startet.

Die Fehlerberechnung des Backward-Pass Algorithmus

Die Fehlerberechnung der aktuellen Belegung des neuronalen Netzes erfolgt in der Methode

- **`void calculateErrorBackwardPassWithWord (double[][] target,double[][] word,int location)`**

Die übergebenen Arrays stellen hier die Ziele der Verbesserung des neuronalen Netzes dar. Das Array **`target`** repräsentiert das Zielmuster, das in der letzten Schicht des neuronalen Netzes abgezeichnet werden soll. Das Array **`word`** ist das Wort des Agenten, das der sein Netz optimierende Agent hört und das er als Ziel für seine Wortbildung nimmt. Die Variable indiziert die Position der Wortschicht innerhalb des neuronalen Netzes, da es bei mehreren verborgenen Schichten theoretisch mehrere mögliche Positionen der Wortplatzierung gibt.

a) Output
Der Algorithmus durchläuft das neuronale Netz rückwärts, beginnend bei der Ausgabeschicht. Dort wird für jedes Neuron ein individueller Fehlerwert berechnet. Dieser ergibt sich aus der Differenz zwischen dem Wert der entsprechenden Zelle des Zielmusters und dem Produkt aus dem aktuellen Aktivierungswert mit dem Funktionswert der sigmoiden Ableitung der Netzeingabe des Neurons. Dieser Wert wird daraufhin in der Variable **`error`** der Klasse **`SomaOutput`** gespeichert.

b) Hidden
Von diesem Fehlerwert aus können auch die theoretischen Fehlerwerte der Neuronen der verborgenen Schichten berechnet werden. Beginnend bei der hintersten verborgenen Schicht, durchläuft eine Schleife alle verborgenen Schichten bis hin zur vordersten, die auf die Eingabeschicht folgt. Die Berechnung der Fehlerwerte ist in jeder Schicht identisch, eine notwendige Abweichung ergibt sich aber in der Schicht, die das Wort repräsentiert.

- kein Wort
  Ist die aktuell zu berechnende Schicht ungleich der Wortschicht, so wird in jedem Neuron, ebenfalls im Biasneuron, zunächst die Summe aller Produkte der Stärken aller ausgehenden Verbindungen mit den Fehlerwerten der Neuronen, die das Ziel der Verbindungen darstellen, errechnet. Dieser Wert wird darauf mit der Ableitung der sigmoiden Funktion der Netzeingabe des Neurons multipliziert. Der sich daraus ergebende Wert wird in der Variable **`error`** der Klasse **`SomaHidden`** gespeichert.

- Wort
  In der Wortschicht funktioniert die Fehlerberechnung ähnlich. Die einzige Erweiterung des bereits beschriebenen Algorithmus ist, dass bei der Aufsummierung der Produkte von Verbindungsstärken und Fehlern der nachfolgenden Schicht auch für jedes Neuron die Differenz zwischen dem Zielwert, also dem Wert des Wortes des anderen Agenten, und dem aktuellen Wert des Neurons, mit berücksichtigt wird. Diese Differenz bildet gewissermaßen einen „Extra-Summanden“.

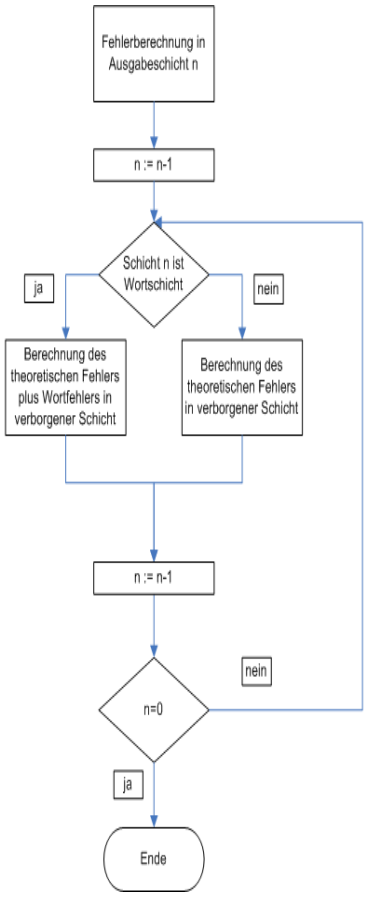


Abb. 133 Flussdiagramm der Fehlerberechnung

Der folgende Auszug aus dem Quellcode zeigt die in Abbildung 133 dargestellte Fehlerberechnung.

```
…
for (int n=getLayers().length-2;n>0;n--)
{
  for (int j=0; j<(getLayers()[n].getField().length);j++)
  {
   for (int k=0; k<getLayers()[n].getField()[j].length;k++)
   {
    for (int l=0;l<getLayers()[n+1].getField().length;l++)
    {
     for (int m=0; m<getLayers()[n+1].getField()[l].length; m++)
     {
      getLayers()[n].getField()[j][k].setError(getLayers()[n]
                .getField()[j][k].getError()+getLayers()[n+1]
                    .getField()[l][m].getError()* getCon()[n]
                         .getField()[j][k][l][m].getValue());
     }
    }
    double wordError = 0;
    if (n==location)
    {
      wordError = word[j][k]-getLayers()[n].getField()[j][k]
                                                  .getActivation();
    }
    getLayers()[n].getField()[j][k].setError
                   ((getLayers()[n].getField()[j][k].getError()
           +wordError)*Tools.sigmoidDerivation(getLayers()[n]
                                .getField()[j][k].getValue()));
   }
  }
  for(int l=0;l<getLayers()[n+1].getField().length;l++)
  {
   for (int m=0; m<getLayers()[n+1].getField()[l].length;m++)
   {
      getLayers()[n].getBias().setError(getLayers()[n]
      .getBias().getError()+getLayers()[n+1].getField()[l][m]
             .getError()* getLayers()[n].getBiasCon()[l][m]);
   }
  }
  getLayers()[n].getBias().setError(getLayers()[n]
                .getBias().getError()*Tools.sigmoidDerivation
                      (getLayers()[n].getBias().getValue()));
}
…
```

Die Gewichtsmodifikation des Backward Pass Algorithmus

Die Gewichtsmodifikation des neuronalen Netzes anhand der vorher berechneten Fehlerwerte geschieht in der Methode

- **`void modifyWeights()`**

In einer Schleife werden alle Instanzen der Klasse **`Connection`** im Array **`con`** durchlaufen, beginnend bei der letzten Stelle des Arrays, welche die Verbindungen zwischen der hintersten verborgenen Schicht und der Ausgabeschicht darstellt, bis hin zur ersten Position, in der die Verbindungen zwischen der Eingabeschicht und der ersten verborgenen Schicht zu finden sind.

Dort wird für jede Verbindung aller Neuronen, ebenfalls dem Bias-Neuron, der vorderen zur hinteren Schicht zuerst die Variable **`deltawij`** des Typs **`double`**, die Differenz zwischen der aktuellen und der gewünschten Verbindungsstärke, bestimmt. Zur Berechnung von **`deltawij`** wird zuerst das Produkt aus Lernrate, dem Fehlerwert des Endneurons einer Verbindung und dem Aktivierungswert des Startneurons einer Verbindung gebildet. Daraufhin wird **`deltawij`** in der Art neu gewichtet, dass die aktuell berechnete Veränderung mit dem Faktor *(1-***`momentum`***)* multipliziert und die Veränderung des vorherigen Simulationsschritts gewichtet mit dem Faktor **`momentum`** hinzuaddiert wird. Zuletzt wird **`deltawij`** zur aktuellen Verbindungsgewichtung addiert. Damit ist die Gewichtsmodifikation abgeschlossen. Der folgende Ausschnitt aus dem Quellcode, eines der Herzstücke von DiaLex, stellt die Gewichtsmodifikation explizit dar.

```
…
for (int h = getCon().length-1; h>=0;h--)
{
 for (int i = 0; i<getCon()[h].getField().length;i++)
 {
  for (int j = 0; j<getCon()[h].getField()[i].length;j++)
  {
   double deltawij = 0;
   for (int k = 0; k<getCon()[h].getField()[i][j].length;k++)
   {
    for(int l=0;l<getCon()[h].getField()[i][j][k].length;l++)
    {
     deltawij=learningRate*getLayers()[h+1].getField()[k][l]
                    .getError()*getLayers()[h].getField()[i][j]
                                                .getActivation();
     deltawij = (1-momentum)*deltawij+momentum*getCon()[h]
                             .getField()[i][j][k][l].getLastChange();
     getCon()[h].getField()[i][j][k][l].setLastChange(deltawij);
     getCon()[h].getField()[i][j][k][l].setValue(getCon()[h]
                    .getField()[i][j][k][l].getValue()+deltawij);

   }
  }
```

```
 }
 double deltawij = 0;
 for(int l=0;l<getLayers()[h].getBiasCon().length;l++)
 {
  for (int m=0; m<getLayers()[h].getBiasCon()[l].length;m++)
  {
   deltawij =  learningRate*getLayers()[h+1].getField()[l][m]
                                                        .getError();
   getLayers()[h].getBiasCon()[l][m] += deltawij;
  }
 }
}
…
```

Das Abschwächen der Verbindungsgewichtungen als Störfaktor

Bereits in der Klasse **`Agent`** befindet sich der erste im Entwurf deklarierte Störfaktor. Es handelt sich um die Abschwächung der Verbindungsgewichtungen der neuronalen Netze. Hierzu wird die Methode

- **`void connectionsDecay(double[] maxDecay)`**

aufgerufen, der ein Array von maximal möglichen Stärken der Abschwächung mitgegeben wird. Somit ist eine verschieden starke Abschwächung in unterschiedlichen Verbindungsschichten des neuronalen Netzes möglich. **`connectionsDecay`** macht dabei nichts weiteres, als innerhalb einer jeden Instanz der Klasse **`Connection`** im Netz der Agenten die Methode

- **`void decay (double maxDecay)`**

mit dem ihr zugehörigen Maximalwert aufzurufen. Diese Methode läuft iterativ durch alle Axone der Verbindungen und verändert die einzelnen Gewichtungen, die in der Variable **`value`** gehalten werden, bestimmt durch den Übergabewert **`maxDecay`** und zwei Zufallsvariablen **`r2`** und **`r3`**. **`maxDecay`** gibt dabei den Maximalwert an, der mit dem durch **`r2`** erzeugten reellen Wert im Intervall [0;1] multipliziert wird und schließlich, bedingt durch den Zufallswert **`r3`**, entweder zu 1 addiert oder von 1 subtrahiert wird.

Multipliziert mit dem so berechneten Faktor bildet der ursprüngliche Gewichtungswert die neue Stärke der Verbindungsgewichtung. Im Folgenden wird der Inhalt der Methode **`decay`** als Auszug aus dem Quellcode dargestellt.

```
…
Random r2 = new Random();
Random r3 = new Random();
for (int i = 0; i<field.length;i++)
{
  for (int j = 0; j<field[i].length;j++)
  {
    for (int k = 0; k<field[i][j].length;k++)
    {
      for (int l = 0; l<field[i][j][k].length;l++)
      {
        if (r3.nextBoolean())
        {
            field[i][j][k][l].setValue(field[i][j][k][l].
            getValue()*(1-((r2.nextDouble()*(maxDecay)))));
        }
        else
        {
            field[i][j][k][l].setValue(field[i][j][k][l].
            getValue()*(1+((r2.nextDouble()*(maxDecay)))));
        }
      }
    }
  }
}
..
```

B.1.2 Die Klasse Environment

Die Klasse **`Environment`** repräsentiert eine Simulationsumgebung und besteht aus einer Reihe von Klassenvariablen, die die notwendigen Attribute einer Umgebung speichern. Im Vergleich zur Vorgängerversion aus LexLearn ist sie erheblich umfangreicher. So wurde sie um die Möglichkeiten der Agentenpositionierung, die Größenangabe der Simulationsumgebung, der Häufigkeit der Inputpattern, den Informationen zum Initialzustand der Agenten, der Sprecher- und Zuhörerwahl, der Kommunikationsart, der Nachbarschaftsgröße und zu allen fünf Störfaktoren erweitert.

- **`int numberOfAgents`**
  Diese Integervariable, die die Klasse **`Environment`** bei der Instanziierung erhält, speichert die Anzahl der Agenten, die sich in der Simulationsumgebung befinden.

- **`Agent[] agents`**
  Dieses Array aus Instanzen der Klasse Agent beinhaltet die Agenten der Umgebung.

- **`double[][][] inputPattern`**
  Dieses dreidimensionale Array des Typs **`double`** beinhaltet die Menge der visuellen Szenen. Hierbei indiziert die erste Dimension die Position der Szene im Array, die zweite und dritte Dimension stehen für die Höhe und Breite des Patterns.

- **`double[] inputPatternFrequency`**
  Dieses Array des Typs **`double`** speichert die relativen Häufigkeiten der visuellen Szenen.

- **`int agentPositioning`**
  Die Variable des Typs **`int`** kodiert die Art und Weise, wie die Agenten in der Simulationsumgebung positioniert werden. Neben einer zufälligen Positionierung können Koordinaten auch manuell gesetzt werden oder aber es kann auf eine Positionierung verzichtet werden.

- **`int enviLength`**
  Die Variable des Typs **`int`** gibt die Länge der Simulationsumgebung an.

- **`int enviWidth`**
  Analog zu **`enviLength`** kodiert diese Variable die Breite der Simulationsumgebung.

- **`int initialState`**
  Die Variable des Typs **`int`** enthält die Auswahl, ob die neuronalen Netze der Agenten mit zufälligen Verbindungsgewichtungen oder aber mit Gewichtungen der Stärke 0 initialisiert werden.

- **`int speakerSelection`**
  Diese Variable des Typs **`int`** kodiert die Sprecherauswahl der Agenten. Die beiden Möglichkeiten sind eine analoge Abfolge oder eine zufällige Auswahl.

- **`int listenerSelection`**
  In dieser Variable wird die Art der Wahl eines Zuhörers gespeichert. Neben der zufälligen Auswahl existieren auch die zufälligen Auswahlen basierend auf der euklidischen Distanz oder der Manhattan-Distanz.

- **`int inputPatternSelection`**
  Die Variable des Typs **`int`** speichert die Art der Wahl der visuellen Szene. Möglichkeiten hier sind die zufällige Auswahl oder eine Wahl basierend auf den relativen Häufigkeiten.

- **`int communicationType`**
  Die Variable des Typs **`int`** kodiert die Art und Weise, wie die Agenten miteinander kommunizieren. Entweder fungiert der erste Agent als Sprecher und der zweite als Zuhörer oder aber beide füllen beide Rollen aus.

- **`int neighbourhoodSize`**
  Die Variable des Typs **`int`** setzt die Größe der Nachbarschaft eines Agenten, in der alle Agenten wie direkte Nachbarn bei der Distanzberechnung behandelt werden.

- **`int[] xPos`**
  Dieses Array des Typs **`int`** speichert die x-Koordinaten zur Positionierung der Agenten.

- **`int[] yPos`**
  In diesem Array des Typs **`int`** werden die y-Koordinaten zur Positionierung der Agenten gespeichert.

- **`double deathProbability`**
  Diese Variable des Typs **`double`** hält die Wahrscheinlichkeit, mit der jeder Agent innerhalb eines Simulationsschritts sterben kann und durch einen neuen untrainierten Agenten ersetzt wird. Sie kodiert somit den ersten Störfaktor.

- **`double[] maxDecays`**
  Dieses Array des Typs **`double`** speichert die maximalen Abschwächungen einer jeden Verbindungsschicht der neuronalen Netze der Agenten. Es enthält somit notwendige Informationen des zweiten Störfaktors.

- **`int maxDecaysActivated`**
  Diese Variable des Typs **`int`** stellt eine Hilfsvariable zum zweiten Störfaktor dar. Wird auf diesen Störfaktor verzichtet, so wird sie auf den Wert 0 gesetzt und somit ein schnellerer Ablauf der Simulation ermöglicht.

- **`double inputPatternNoise`**
  Diese Variable des Typs **`double`** kodiert die Stärke des Rauschens bei der Perzeption der visuellen Szenen und somit den Einfluss des dritten Störfaktors.

- **`double communicationNoise`**
  Diese Variable des Typs **`double`** speichert die Stärke des Rauschens bei der Kommunikation zweier Agenten. Sie stellt den vierten Störfaktor dar.

- **`double thresholdDCT`**
  Diese Variable des Typs **`double`** hält den Grenzwert, ab dem Informationen bei der Komprimierung des Worts eines Agenten mit Hilfe der diskreten Kosinustransformation nach der Quantisierung wegfallen. Somit wird sie zur Berechnung des fünften Störfaktors benötigt.

Bei der Instanziierung der Klasse **`Environment`** werden dem Konstruktor folgende Attribute übergeben, die zumeist im vorigen Teil erläutert wurden und sich in mehrere Untergruppen einteilen lassen:

- **`double[][][] inputPattern`**
- **`double[] inputPatternFrequency`**
- **`int numberOfAgents`**

Mit diesen Variablen werden die elementaren Informationen einer Simulationsumgebung übergeben: Die Menge der visuellen Szenen, ihre relative Häufigkeit und die Anzahl der Agenten.

- **`int agentPositioning`**
- **`int enviLength`**
- **`int enviWidth`**
- **`int[] xPos`**
- **`int[] yPos`**

Die in dieser Gruppe aufgeführten Informationen enthalten sämtliche für die Positionierung der Agenten benötigten Angaben.

- **`int[] nnLengths`**
- **`int[] nnWidths`**
- **`int wordPosition`**
- **`double learningRate`**
- **`double momentum`**
- **`int initialState`**

Mit Hilfe dieser Variablen werden die Agenten bei der Instanziierung der Klasse **`Environment`** initialisiert.

- **`int speakerSelection`**
- **`int listenerSelection`**
- **`int inputPatternSelection`**
- **`int communicationType`**
- **`int neighbourhoodSize`**

Die Angaben dieser Gruppe sind für den korrekten Ablauf der Kommunikation der Agenten notwendig.

- **`double deathProbability`**
- **`double[] maxDecays`**
- **`double inputPatternNoise`**
- **`double communicationNoise`**
- **`double dctThreshold`**

Die letzte Gruppe von Variablen, die bei der Instanziierung übergeben werden, stellen die fünf Störfaktoren dar.

Ebenso beinhaltet die Klasse **`Environment`** eine Menge von Methoden, die sowohl für die Instanziierung der Agenten als auch für den Ablauf der Simulation benötigt werden. Aufgrund der großen Menge von nahezu 40 Methoden können im Folgenden nur die wichtigsten kurz erläutert werden:

- **`int selectAgentRandomly()`**
- **`int selectAgentInOrder(int simStep)`**
- **`int selectAgentProximityEuclidean(int speakerID)`**
- **`int selectAgentProximityManhattan(int speakerID)`**

  Diese vier Methoden werden zur Auswahl des Sprechers und des Zuhörers in einem Simulationsschritt benötigt. Während **`selectAgentRandomly()`** für beide Funktionen verwendet werden kann, beschränkt sich der Einsatz von **`selectAgentInOrder(int simStep)`** auf die Sprecherwahl und der von **`selectAgentProximityEuclidean(int speakerID)`** und **`selectAgentProximityManhattan(int speakerID)`** auf die Wahl des Zuhörers.

- **`int selectInputPatternRandomly()`**
- **`int selectInputPatternFrequency()`**

  Mit Hilfe dieser beiden Funktionen wird eine visuelle Szene zur Kommunikation ausgewählt, entweder ausschließlich durch den Zufall bedingt oder aber zufällig, jedoch unter Berücksichtigung der relativen Häufigkeiten.

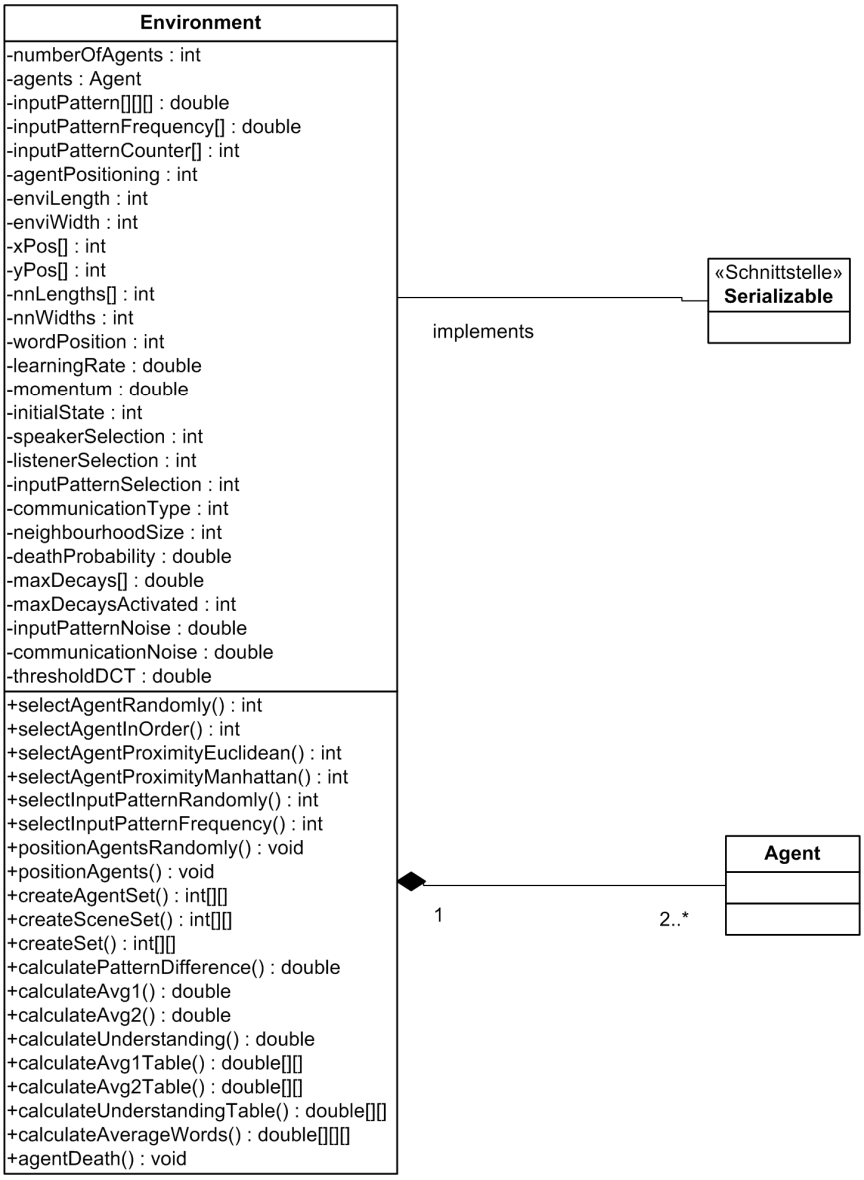


Abb. 134 Klassendiagramm der Klasse Environment

- **`void positionAgentsRandomly()`**
- **`void positionAgents(int[],int[])`**
  Diese beiden Methoden positionieren die Agenten in ihrer Simulationsumgebung. Dies geschieht entweder zufällig oder durch die explizite Übergabe von Koordinaten.

- **`int[][] createAgentSet()`**
- **`int[][] createSceneSet()`**
- **`int[][] createSet(int, int)`**
- **`int[][] createSet(int[])`**

  Die ersten beiden Methoden kreieren die Menge aller Zweitupel, entweder von Agenten oder visuellen Szenen, die beiden letzteren sind Hilfsmethoden dazu. Diese Methoden werden bei der Berechnung der Fehlerindizes benötigt.

- **`double calculatePatternDifference( double[][] in, double[][] out)`**

  Die hier genannte Methode berechnet die Differenz zwischen zwei übergebenen Mustern. Sie wird zur Berechnung der Fehlerindizes benötigt.

- **`double calculateAvg1()`**

  In der Methode **`calculateAvg1()`** wird der Fehlerquotient Avg1 berechnet. Dies geschieht durch Aufsummieren und anschließende Mittelwertsbildung der Differenz der Wortschichten eines Agenten für jeweils zwei visuelle Szenen für alle Tupel der Menge, die in **`createSeceneSet()`** gebildet wurde. Dies wird für jeden Agenten der Gemeinschaft berechnet. Daraus wird der Mittelwert der individuellen Abweichungen der Agenten gebildet.

- **`double calculateAvg2()`**

  Die Methode **`calculateAvg2()`** errechnet den Fehlerquotient Avg2. Dies geschieht durch die Differenzbildung der Inhalte der errechneten Ausgabeschicht zweier Agenten. So wird für jede visuelle Szene und jeden möglichen Tupel zweier Agenten die Differenz aufsummiert und danach der Mittelwert berechnet.

- **`double calculateUnderstanding()`**

  Die Methode **`calculateUnderstanding()`** errechnet den Fehlerquotient Understanding. Dies geschieht durch die Ermittlung des Kommunikationserfolgs von jeweils zwei Agenten. So wird für jede visuelle Szene und jeden Tupel zweier Agenten der Erfolg aufsummiert und danach der Mittelwert berechnet.

- **`double[][] calculateAvg1Table()`**
- **`double[][] calculateAvg2Table()`**
- **`double[][] calculateUnderstandingTable()`**

  Diese Methoden errechnen zu jeder der im Vorigen erläuterten Methoden eine Kreuztabelle um den Vergleich zwischen allen Eingabemustern oder allen Agenten zu ermöglichen.

- **`double[][][] calculateAverageWords()`**
  Die Methode **`calculateAverageWords()`** bildet das gemeinsame Lexikon aller Agenten. Somit wird ein Vergleich dieses Lexikons über den Simulationsverlauf möglich gemacht.

- **`void agentDeath()`**
  Die Methode **`agentDeath()`** simuliert den möglichen Tod eines Agenten. Zuerst wird mit Hilfe eines Zufallsgenerators entschieden, ob dieses Ereignis eintritt und falls ja, wird ein neuer untrainierter Agent mit den entsprechenden Parametern an seiner Stelle im Array **`agents`** erzeugt.

B.1.3 Die Klasse Simulation

Die Klasse **`Simulation`** bildet die oberste der drei Schichten des Simulationsbereichs von DiaLex. Durch die in diesem Abschnitt beschriebenen Komponenten der Klasse ist es möglich, alle in der Modellbeschreibung beschriebenen Arten von Simulationen zu deklarieren und durchzuführen. Mit den in ihr enthaltenen Unterklassen **`Database`** und **`EventManager`** bildet sie die einzige Schnittstelle zur grafischen Oberfläche, die im nächsten Unterkapitel erläutert wird, und grenzt somit klar den Simulationsbereich von anderen Teilen des Programms ab.

Die Hauptklasse Simulation

In der Klasse **`Simulation`** kommt das Singleton Pattern zur Anwendung, damit gewährleistet ist, dass keine weitere Instanz, außer der im Vorhinein statisch angelegten **`instance`**, erzeugt wird. Auf sie kann mit Hilfe der Methode **`getInstance()`** zugegriffen werden. Dadurch ergibt sich, dass es keinen öffentlichen Konstruktor der Klasse **`Simulation`** geben kann. Alle Klassenvariablen müssen folglich explizit gesetzt werden. Abbildung 135 zeigt die Erstellung und den Durchlauf einer Simulation unter besonderer Betrachtung der Klasse **`Simulation`** in Form eines Flussdiagramms.

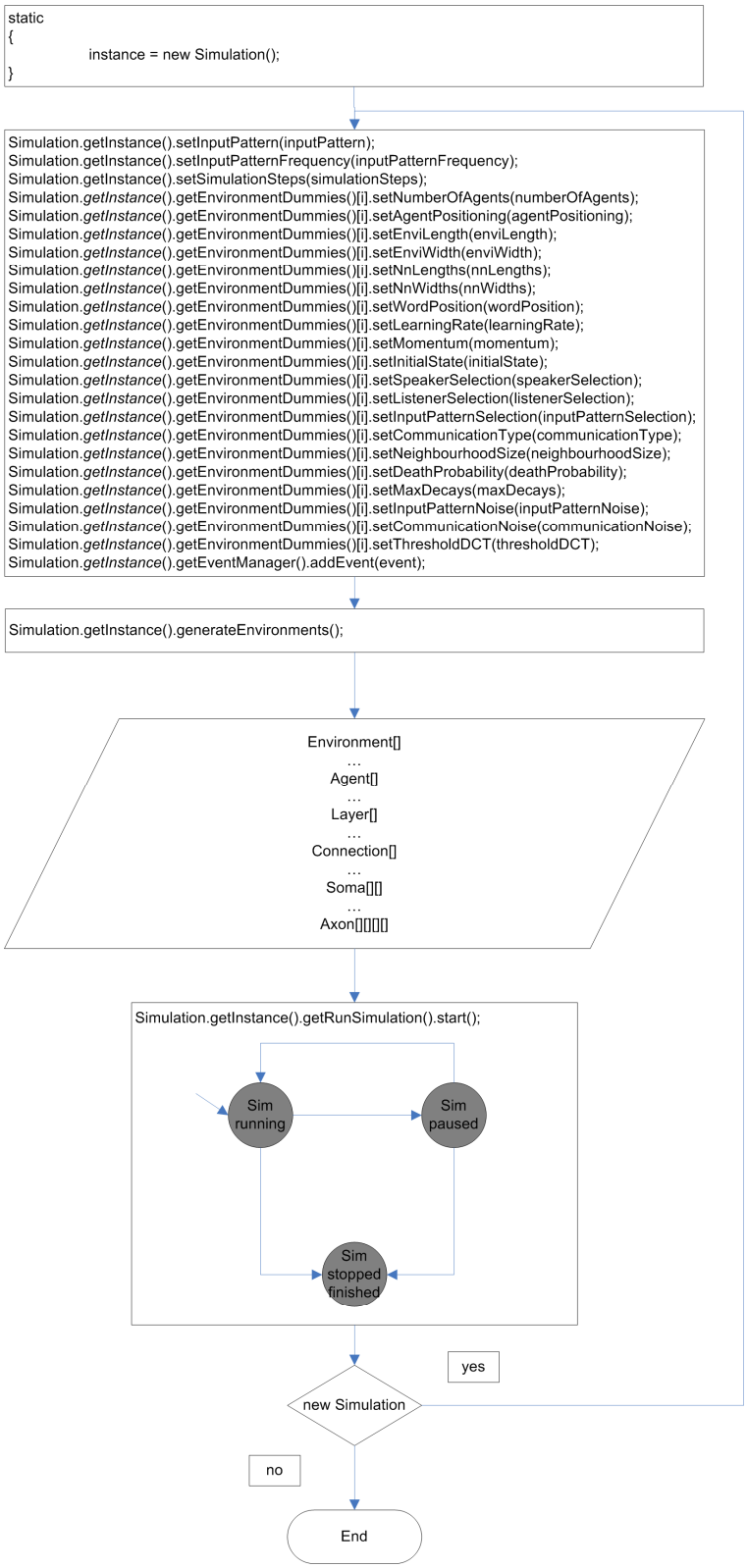


Abb. 135 Flussdiagramm zu Erstellung und Durchlauf einer Simulation

Durch die Notwendigkeit, die Daten mehrerer Simulationsumgebungen mit allen notwendigen Angaben in der Klasse **`Simulation`** zu speichern, ergibt sich eine Reihe von Klassenvariablen, die im Folgenden erläutert werden:

- **`int simulationSteps`**
  Die Variable **`simulationSteps`** speichert die Anzahl der Simulationsschritte.

- **`int actualSimulationStep`**
  Diese Variable beinhaltet während eines Simulationsdurchlaufs den aktuellen Simulationsschritt.

- **`double[][][] inputPattern`**
  Dieses dreidimensionale Array des Typs **`double`** speichert die Sequenz der visuellen Szenen, die den Agenten der Simulationsumgebungen präsentiert wird.

- **`double[] inputPatternFrequency`**
  Dieses Array des Typs **`double`** speichert die relativen Häufigkeiten der visuellen Szenen.

- **`EnvironmentAttributesDummy[] environmentDummies`**
  Das Array der Klassen **`EnvironmentAttributesDummy`** hält alle simulationsumgebungsspezifischen Werte bevor ein Simulationsdurchlauf gestartet wird. Dies ist von Vorteil, da sonst eine Änderung eines Attributs einer Simulationsumgebung entweder zu performance-beeinträchtigenden Umbaumaßnahmen innerhalb der Instanzen der Klasse **`Environment`** oder aber zu inkonsistenten Daten führen könnte. Aus diesem Grund werden die Simulationsumgebungen als Instanzen der Klasse **`Environment`** erst mit dem Start eines Simulationsdurchlaufs erzeugt. Auf eine Auflistung der bereits im Abschnitt B.1.2 aufgeführten und erläuterten Attribute wird an dieser Stelle verzichtet.

- **`Environment[] environments`**
  In diesem Array aus Instanzen der Klasse **`Environment`** werden die Simulationsumgebungen mit allen Bestandteilen gehalten. Sie werden durch die Methode **`generateEnvironments()`** erzeugt.

- **`EventManager eventManager`**
  Die in der Klasse **`Simulation`** enthaltene Instanz der Klasse **`EventManager`** übernimmt die Verwaltung der Ereignisse während eines Simulationsdurchlaufs. Seine genaue Funktionsweise wird im Unterabschnitt „*Die Unterklassen des Bereichs Ereignis*“ detailliert erläutert.

- **`Database db`**
  Die Datenbankklasse übernimmt das Speichern der durch Ereignisse gesammelten Simulationsdaten und liefert diese bei der Auswertung an die grafische Benutzeroberfläche. Ihre genaue Funktionsweise wird im Unterabschnitt „*Die Unterklassen des Bereichs Datenbank*“ gezeigt.

- **`DoSimulation runSimulation`**
  Die benutzeroberflächenspezifischen Teile der Berechnung eines Simulationsdurchlaufs sind in die Klasse **`DoSimulation`** ausgelagert. Der Vorteil dieser Implementierungsvariante ist, dass die Berechnung so in einem eigenen Thread geschieht. Dies hat für den Simulationsteil von DiaLex keine weiteren Konsequenzen, jedoch ist es dank dieser Konstruktion möglich, auch während eines Simulationsdurchlaufs mit der grafischen Benutzeroberfläche zu arbeiten und so beispielsweise einen Simulationsdurchlauf zu pausieren oder abzubrechen.

Die wichtigsten Methoden der Klasse Simulation werden im Folgenden beschrieben.

- **`void generateEnvironments()`**
  Diese Methode wird beim Starten eines Simulationsdurchlaufs ausgeführt. Erst zu diesem Zeitpunkt werden die Instanzen der Klasse **`Environment`** generiert, auf denen danach die Simulation ausgeführt wird. Der Vorteil dieser Variante ist, dass sich Änderungen in den Simulationsparametern so nur auf lokal gespeicherte Variablen auswirken und keine Daten in komplexeren, ineinander verschachtelten Klassen verändert werden müssen.

- **`void simulationStart()`**
  Die Methode **`simulationStart()`**, die beim Starten eines Simulationsdurchlaufs ausgeführt wird, instanziiert die Klasse **`DoSimulation`** und führt dort die Methode **`run()`** in einem neuen Thread aus.

- **`void simulationPause()`**
  Diese Methode hält den Thread, in dem die Simulation durchgeführt wird, bei laufender Berechnung an; sollte sie bereits im Pausezustand sein, so wird die Simulation fortgesetzt.

- **`void simulationStop()`**
  Mit der Methode **`simulationStop()`** werden der Simulationsdurchlauf und damit der zugehörige Thread beendet.

- **`void doSimulation ()`**
  In dieser Methode wird der eigentliche Simulationsdurchlauf ausgeführt. Wie bereits mehrfach erläutert wird sie aus Gründen der Ansprechbarkeit der grafischen Benutzeroberfläche aus der Klasse **`DoSimulation`** aufgerufen. Neben dem regulären Ablauf einer Simulation, also der Auswahl von Agenten und visuellen Szenen und der Durchführung einer Kommunikation übernimmt sie ebenfalls die Positionierung der Agenten, den Einfluss der einzelnen Störfaktoren innerhalb der Agentengemeinschaften und die Ereignisbehandlung. So überprüft sie alle im **`EventManager`** gehaltenen Ereignisse auf Eintritt, verändert die entsprechenden Simulationsparameter zur Laufzeit, und, handelt es sich um ein Ereignis, das Simulationsdaten liefert, so werden diese an die Datenbank zum Speichern übergeben. Aufgrund der Wichtigkeit dieser Methode wird ihre Arbeitsweise im Unterabschnitt *„Die Methode doSimulation“* detailliert erläutert.

- **`static void save (String path)`**
  Die Methode **`save`** speichert den aktuellen Zustand der Klasse **`Simulation`** binär unter dem im Konstruktor übergebenen Dateipfad.

- **`static void load (String path)`**
  Analog zur Methode **`save`** lädt die Methode **`load`** die kompletten Daten einer vorher gespeicherten Simulationsklasse.

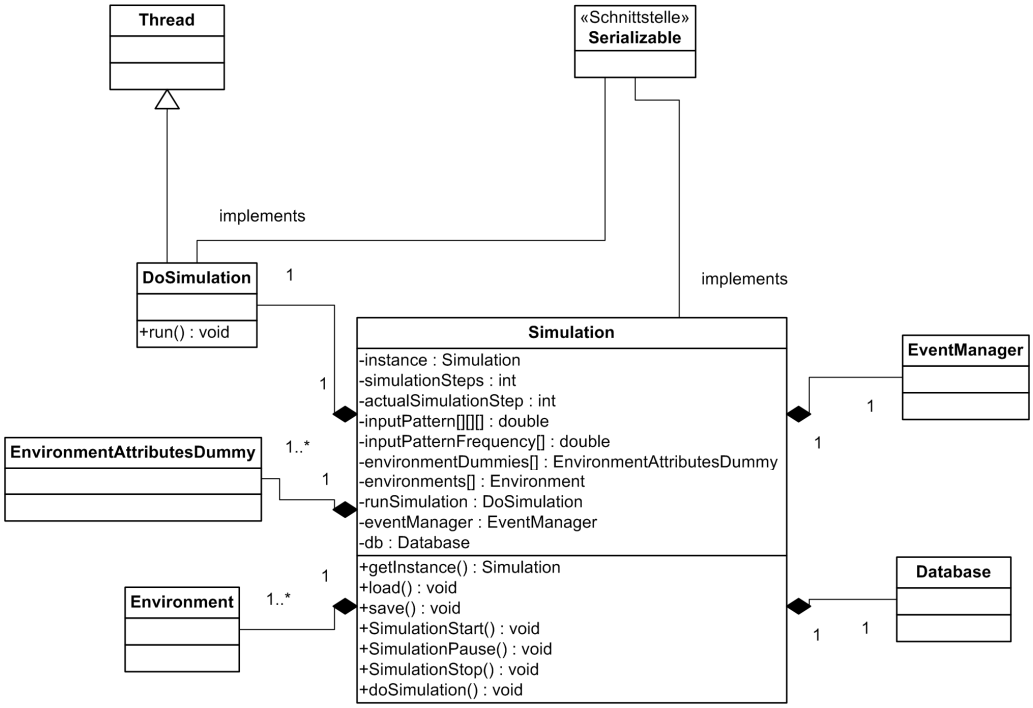


Abb. 136 Klassendiagramm der Klasse Simulation

Die Unterklassen des Bereichs Ereignis

Der Ereignisbereich der Klasse **`Simulation`** gliedert sich in eine Instanz der Klasse **`EventManager`** und beliebig viele in ihr enthaltene Instanzen der Klasse **`Event`**.

Die Klasse EventManager

Die Klasse **`EventManager`** beinhaltet als Klassenvariablen drei Listen des Typs **`ArrayList`**, in denen die Ereignisse als Instanzen der Klasse **`Event`** gespeichert werden. **`startEventList`** ist dabei eine Liste aller vor dem Start eines Simulationsdurchlaufs deklarierten Ereignisse, **`triggeredEventList`** listet alle eingetretenen Ereignisse auf und **`currentEventList`** beinhaltet alle aktuell vorhandenen Ereignisse, die während eines Simulationsdurchlaufs noch auf ihren Eintritt überprüft werden müssen.

Zusätzlich stellt die Klasse einige Methoden zur Verfügung, die für die Ereignisverwaltung unabdingbar sind.

- **`void addEvent(Event c)`**
  Mit dieser Methode wird ein neues Ereignis in die **`startEventList`** hinzugefügt.

- **`void delEvent(Event c)`**
  Diese Methode entfernt ein Ereignis aus der **`startEventList`**.

- **`void simulationStarts()`**
  Diese Methode wird beim Start eines Simulationsdurchlaufs ausgeführt und hat zwei Aufgaben: Zuerst kopiert sie alle Ereignisse der **`startEventList`** in die **`currentEventList`**, danach initialisiert sie die **`triggeredEventList`**.

Die Klasse Event

Die Klasse **`Event`** kodiert genau ein Ereignis, das entweder eine Veränderung einer oder mehrerer Variablen innerhalb einer Simulationsumgebung während der Laufzeit bewirkt, oder aber ein Sammeln von Daten mit anschließender Speicherung in der Datenbank auslösen kann. Hierzu enthält es die nachfolgenden Klassenvariablen:

- **`int environmentNumber`**
  Diese Variable kodiert die Simulationsumgebung, zu der das Ereignis gehört.

- **`String eventChooser`**
  Diese Zeichenkette kodiert die Art der Bedingung, die zur Auslösung des Ereignisses führt.

- **`Object eventObject`**
  Dieses Objekt steht in Beziehung zu dem vorherigen String. Zusammen bilden sie die Eintrittsbedingung.

- **`String effectChooser`**
  Diese Zeichenkette gibt an, was nach Eintritt des Ereignisses ausgeführt werden soll.

- **`Object effectObject`**
  Dieses Objekt gehört zur Variable **`effectChooser`**. Zusammen deklarieren sie, welche Auswirkungen das Ereignis mit sich bringt.

- **`Object extraObject`**
  Da manche Ereignisse zwei Objekte benötigen um einen Effekt zu kodieren, wurde das Ereignis um **`extraObject`** erweitert. Dies geschieht besonders bei wieder eintretenden Ereignissen.

- **`boolean toInfinity`**
  Diese Variable sagt aus, ob ein Ereignis nach Eintritt in eventuell veränderter Form wieder in die Liste der zu überprüfenden Ereignisse aufgenommen werden soll.

- **`Event nextEvent`**
  Mit **`nextEvent`** wurde auf Simulationsebene die Möglichkeit geschaffen, dass ein Ereignis mit seinem Eintritt ein anderes Ereignis erzeugen kann. Diese Fähigkeit wurde jedoch nicht in die grafische Benutzeroberfläche integriert und ist somit als Feature für eventuelle spätere Versionen des Simulationstools zu sehen. Es wird jedoch der Vollständigkeit halber an dieser Stelle erwähnt.

Außer den obligatorischen Getter- und Setter-Methoden enthält die Klasse keine weiteren Methoden. Im Folgenden werden einige Beispiele möglicher Ereigniskodierungen zuerst natürlichsprachig mit anschließender Belegung der Klassenvariablen dargestellt. Sämtliche erwähnten Textstrings sind in der Klasse **`ConstantStrings`** deklariert und werden dem Benutzer in einem Drop-Down-Menü zur Verfügung gestellt.

- In Simulationsumgebung 1 soll im Simulationsschritt 1000 die Nachbarschaftsgröße auf 5 gesetzt werden.

```
environmentNumber       1
eventChooser            simstep==
eventObject             1000
effectChooser           neighbourhoodsize=
effectObject            5
extraObject             null
toInfinity              false
```

- In Simulationsumgebung 2 soll, sobald der Understanding-Indikator einen Wert > 0.75 erreicht, die Zuhörerauswahl über die Manhattan-Distanz ausgewählt werden.

```
environmentNumber       2
eventChooser            understanding>
eventObject             0.75
effectChooser           listenerSelection=
effectObject            2
extraObject             null
toInfinity              false
```

- In Simulationsumgebung 1 soll im Simulationsschritt 1000 der Indikator Avg1 berechnet und in die Datenbank eingetragen werden.

```
environmentNumber       1
eventChooser            simstep==
eventObject             1000
effectChooser           avg1
effectObject            null
extraObject             null
toInfinity              false
```

- In Simulationsumgebung 3 soll im Simulationsschritt 10000 der Indikator Avg2 der Agenten 0,1,2,3 und 4 berechnet und in die Datenbank eingetragen werden. Dies soll sich alle 1000 Schritte wiederholen.

```
environmentNumber       3
eventChooser            simstep==
eventObject             10000
effectChooser           avg2_
effectObject            1000
extraObject             0,1,2,3,4
toInfinity              true
```

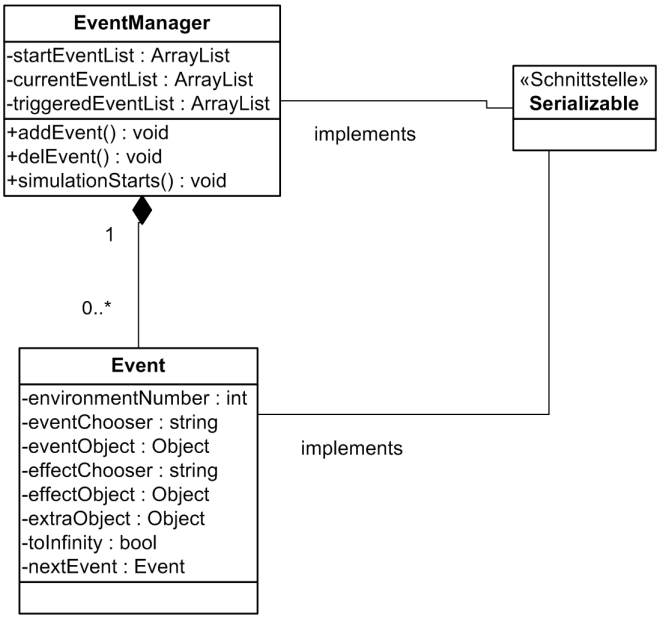


Abb. 137 Klassendiagramm der Event-Klassen

Die Unterklassen des Bereichs Datenbank

Der Datenbankbereich der Klasse **`Simulation`** wird durch die Klassen **`Database`**, **`DbList`**, und **`DbObject`** realisiert. **`Database`** stellt hierbei die Funktionalitäten einer primitiven relationalen Datenbank zur Verfügung, **`DbList`** repräsentiert eine Liste in dieser Datenbank, die mit Objekten des Typs **`DbObject`** befüllt wird.

Die Klasse Database

Die notwendigen Datenbankfunktionen werden in der Klasse **`Database`** zusammengefasst, die wie eine relationale Datenbank funktioniert und angesprochen wird. Aufgrund der relativ geringen Menge an Daten, die in DiaLex gewonnen werden, wurde jedoch von dem Einsatz einer etablierten Datenbank abgesehen. Vielmehr ist die nun bereits vorhandene Schnittstelle als eine Vorbereitung auf eventuelle weitergehende Programmversionen zu sehen. Als Klassenvariable enthält **`Database`** nur eine Liste namens **`dbLists`** des Typs **`ArrayList`**. In ihr werden die Instanzen der Klasse **`DbList`** gehalten. Die Funktionalitäten einer Datenbank werden mit den folgenden Methoden realisiert:

- **`void createList(int envi, String name, String type)`**
  Diese Methode erzeugt eine neue Liste innerhalb der Datenbank. Die übergebenen Variablen definieren die Simulationsumgebung, den Namen der Liste und den Klassennamen der Elemente einer Liste.

- **`void addElement(int envi, String name, int step, Object ob)`**
  Mit dieser Methode wird ein Element zu einer Liste hinzugefügt. **`envi`** und **`name`** erlauben eine Listenzuordnung, **`step`** gibt den Simulationsschritt an und **`ob`** liefert das zu speichernde Objekt.

- **`boolean hasList(int envi, String name)`**
  Diese Methode prüft, ob bereits eine Liste mit den beiden übergebenen Eigenschaften existiert.

- **`boolean isEmpty()`**
  Diese Methode prüft, ob die Datenbank leer ist.

- **`int getNumberOfLists()`**
  Diese Methode liefert die Anzahl der in der Datenbank gehaltenen Listen.

- **`void show()`**
  Eine Auflistung aller Datenbanklisten mit ihren Eigenschaften liefert diese Methode.

- **`DbList getList (int position)`**
  Diese Methode liefert eine Liste der Datenbank, die über ihre Position in **`dbLists`** ausgewählt wird.

- **`DbList getList(int envi,String name)`**
  Mit dieser Methode ist es möglich, eine Liste über ihre zugehörige Simulationsumgebung und ihren Namen zu finden.

- **`ArrayList getLists (int envi)`**
  Diese Methode liefert alle zu einer Simulationsumgebung zugehörigen Listen.

Die Klasse DbList

Die Klasse **`DbList`** stellt eine Datenbankliste dar. Sie enthält als Klassenvariablen die Variable **`envi`** vom Typ **`int`** und die beiden Variablen des Typs **`String`**, **`name`** und **`type`**, die ihr alle im Konstruktor innerhalb der Methode **`createList()`** in der Klasse **`Database`** übergeben werden. Durch sie

ist die Liste eindeutig identifizierbar. Als einzige weitere Klassenvariable existiert eine Liste namens **`dbObjects`** vom Typ **`ArrayList`**. In ihr werden die einzelnen Datenbankobjekte gehalten. In der Klasse werden die folgenden Methoden definiert:

- **`void add(int step, Object ob)`**
  Diese Methode fügt der Liste ein Element hinzu. Sie wird in der Methode **`addElement()`** in **`Database`** aufgerufen.

- **`int size()`**
  Die Anzahl der Elemente einer Liste wird von dieser Methode zurückgegeben.

- **`int[] getSteps()`**
  Diese Methode liefert ein **`array`** des Typs **`int`**, das die Simulationsschrittangaben aller in der Liste enthaltenen Elemente enthält.

- **`Object getElement(int step)`**
  Mit dieser Methode ist es möglich, ein Element der Liste mit Hilfe des zugehörigen Simulationsschritts zu finden.

Die Klasse DbObject

In der Klasse **`DbObject`** werden die einzelnen Daten einer Datenbankliste gespeichert. Dazu benötigt sie die beiden Klassenvariablen **`step`** vom Typ **`int`** und **`ob`** vom Typ **`Object`**. **`step`** speichert dabei den zugehörigen Simulationsschritt zum gespeicherten Datum **`ob`**. Beide Variablen werden direkt bei der Instanziierung der Klasse, was in der Methode **`addElement()`** der Klasse **`Database`** geschieht, übergeben. Aufgrund des Einsatzes der Klasse **`Object`** als Speicherklasse ist es möglich, sämtliche in Java deklarierten Klassen in dieser Datenbank zu speichern. Dies ist in der Praxis ein entscheidender Vorteil, da das Spektrum der gehaltenen Daten von einfachen Werten des Typs **`double`** bis hin zu kompletten Simulationsumgebungen der Klasse **`Environment`** reicht. Das Klassendiagramm des Datenbankbereichs der Simulation stellt den Abschluss dieses Unterabschnitts dar.

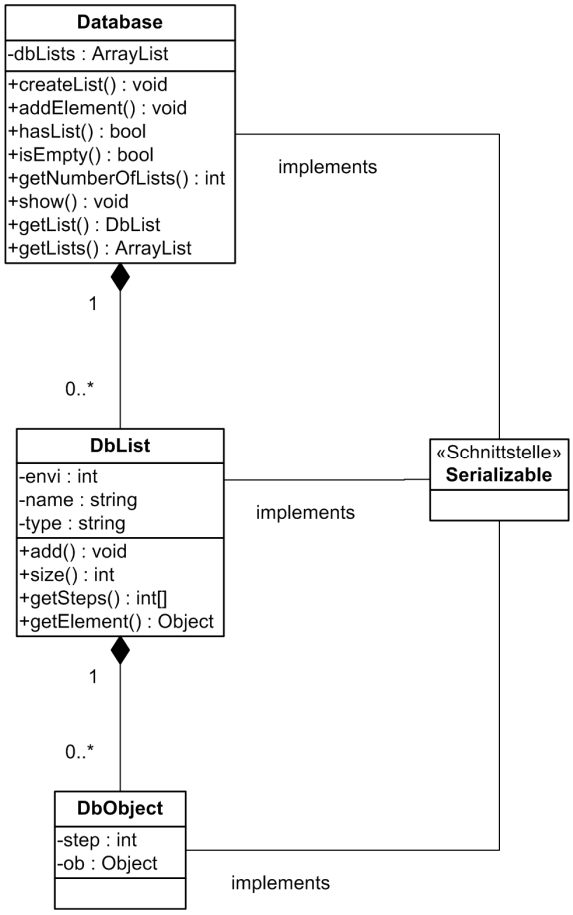


Abb. 138 Klassendiagramm der Datenbank-Klassen

Die Methode doSimulation

Die Methode **`doSimulation`**, die innerhalb des Threads **`runSimulation`** ausgeführt wird, führt die eigentlichen Simulationsberechnungen durch. Sie unterteilt sich grob in drei Abschnitte, die in je einem der folgenden Absätze dargestellt werden:

Aktionen vor dem Start eines Simulationsdurchlaufs

Vor dem Start eines Simulationsdurchlaufs wird zuerst der Status in der Klasse **`StatusManager`** auf „*2*“ gesetzt (vgl. Abschnitt B.2.6). Außerdem wird die Datenbank neu initialisiert und somit alle dort aus vorangegangenen Simulationen befindlichen Daten gelöscht. Weiterhin wird der EventManager mit der Methode **`simulationStarts()`** auf den Start eines Simulationsdurchlaufs vorbereitet. Zuletzt werden die Agenten aller Simulationsumgebungen entsprechend der Vorgaben positioniert.

Aktionen während eines Simulationsdurchlaufs

Der Hauptteil der Methode steht für die Aktionen während eines Simulationsdurchlaufs. Da sich alle nun folgenden Anweisungen in einer Schleife befinden, werden sie zu jedem Simulationsschritt einmal ausgeführt.

Zuerst wird der **`ProgressMonitorThread`** des Hauptfensters mit dem aktuellen Simulationsschritt aktualisiert. Alle nun folgenden Schritte werden für jede Simulationsumgebungen durchgeführt.

Der erste Teil gebührt der Ereignisverwaltung. Alle definierten Ereignisse, die sich in der **`currentEventList`** der Klasse **`EventManager`** befinden, werden auf ihr Eintreten überprüft. Dies wird exemplarisch am Beispiel der Bedingung „*simstep==*“ an Hand eines Quellcodeauszugs gezeigt:

```
…
if (ev.getEventChooser().equals(ConstantStrings.EVENTSIMSTEPS))
{
  int step = Integer.parseInt(ev.getEventObject().toString());
  if(actualSimulationStep==step)
  {
    action = true;
  }
}
…
```

Ist die Variable **`action`** des Typs **`boolean`** nach der Überprüfung mit dem Wert **`true`** belegt, so wird der Effektteil des Ereignisses abgearbeitet. Als Beispiele für beide Arten von Ereignissen wird neben dem simplen Simulationsereignis der Veränderung der Sterbewahrscheinlichkeit der Agenten

```
…
if (ev.getEffectChooser().equals(
                        ConstantStrings.EFFECTDEATHPROBABILITY))
{
  double dp = new Double(ev.getEffectObject().
                                      toString()).doubleValue();
  environments[currentEnvironment].setDeathProbability(dp);
}
…
```

auch die Berechnung des Avg1 Indikators mit Datenbankbehandlung angeführt:

```
…
if(ev.getEffectChooser().equals(ConstantStrings.EFFECTAVG1))
{
  if (ev.getExtraObject()==null)
  {
    if (!db.hasList(currentEnvironment,ev.getEffectChooser()))
    {
      db.createList(currentEnvironment,ev.getEffectChooser(),
                                                    "double");
    }
    db.addElement(currentEnvironment,ev.getEffectChooser(),
      actualSimulationStep,environments[currentEnvironment].
                                              calculateAvg1());
  }
  else
  {
    int[] agli = (int[]) ev.getExtraObject();
    String agliString = Tools.showIntArray(agli);
    String dbString = ev.getEffectChooser()+"_"+agliString;
    if (!db.hasList(currentEnvironment,dbString))
    {
      db.createList(currentEnvironment,dbString,"double");
    }

db.addElement(currentEnvironment,dbString,actualSimulationStep,
          environments[currentEnvironment].
          calculateAvg1PatternAgentSelection(agli));
  }
  if(ev.isToInfinity())
  {
    int newstep = Integer.parseInt(ev.getEventObject().

toString())+Integer.parseInt(ev.getEffectObject().toString());
    ev.setEventObject(new Integer(newstep));
  }
}
…
```

Es ist ersichtlich, dass in der Datenbank der errechnete Wert abgelegt wird. Außerdem wird das aktuell abgearbeitete Ereignis in die **`triggeredEventList`** der Klasse **`EventManager`** kopiert und, falls eine Wiederholung eines Ereignisses gewünscht ist, ein aktualisiertes neues Ereignis erzeugt, das in die **`currentEventList`** kopiert wird.

Als Nächstes wird der Störfaktor der Abschwächung der neuronalen Netze abgearbeitet. Dazu wird, falls eine Abschwächung definiert wurde, die Methode **`connectionsDecay(double maxDecay)`** der Agenten aufgerufen. Da diese Methode bereits im Abschnitt B.1.1 dieser Arbeit erläutert wurde, wird sie hier nicht noch einmal angeführt.

Danach wird der zweite Störfaktor berechnet. Ist eine Agentensterblichkeit vom Benutzer definiert worden, so wird die Methode **`agentDeath(int agentID)`** in der Klasse **`Environment`** für jeden Agenten aufgerufen, die über den Tod eines Agenten entscheidet und ihn bei Bedarf neu initialisiert. Die Methode wurde bereits in Abschnitt B.1.2 erläutert.

Als nächste Aktionen werden die visuelle Eingabeszene, der Sprecher und der Zuhörer der nachfolgenden Kommunikation auf die durch den Benutzer gewünschte Art ausgesucht. Dies geschieht in der angegebenen Reihenfolge, da beispielsweise eine Zuhörerauswahl durch die Entfernung zum Sprecher bestimmt werden kann.

Bevor jedoch eine Kommunikation stattfindet, kommt der Einfluss des dritten Störfaktors zum Tragen. Wurde ein Rauschen während der Perzeption der Eingabeszene definiert, so werden die Eingabemuster, bevor sie den Agenten präsentiert werden, mit der Methode **`noise (double[][] in, double value)`** der Klasse **`Tools`** verrauscht. Jeder Eingabewert wird dabei mit einer vorher definierten Wahrscheinlichkeit durch einen Zufallswert ersetzt. Ihre Funktionsweise wird an dieser Stelle anhand des Quellcodes gezeigt:

```
…
double[][] out = new double[in.length][in[0].length];
Random r = new Random();
Random s = new Random();

for (int i=0; i<out.length;i++)
{
  for(int j=0; j<out[i].length;j++)
  {
    if(r.nextDouble()<value)
    {
      out[i][j] = s.nextDouble();
    }
    else
    {
      out[i][j] = in[i][j];
    }
  }
}
…
```

Daraufhin folgen die Berechnungen innerhalb der neuronalen Netze der Agenten. Dies geschieht über den Aufruf der Methode **`calculateAll(double[][] in)`** in der Klasse **`Agent`**. Die so errechneten Wortmuster beider Agenten werden, falls dies die Simulation so vorsieht, mit der gleichen Methode wie die Eingabemuster verrauscht. Dies ist der vierte Störfaktor.

Danach kommt der fünfte und letzte Störfaktor zum Tragen. Die Wortmuster werden gegebenenfalls in der Klasse **`DCT`** mit Hilfe der diskreten Kosinustransformation umgeformt, quantifiziert, dequantifiziert und wieder in ihre Ursprungsform zurückverwandelt. Dies geschieht mit dem Aufruf der Methode **`completeDCTRound (double[][] in, double threshold, int choice)`**, der neben dem Wortpattern der Quantisierungsschwellenwert übergeben wird. **`choice`** wählt zwischen einer linearen und einer exponentiellen Quantisierungsmatrix aus. Im Quellcode ist sie jedoch fest auf die lineare Variante gesetzt. Der nachfolgende Ausschnitt aus dem Quellcode der Klasse **`DCT`** zeigt die Funktionsweise der Methode **`forwardDCT (double[][] input)`**, die ein zwei-dimensionales Array des Typs **`double`** mit Hilfe der diskreten Kosinustransformation umwandelt.

```
…
double[][] output = new double[input.length][input[0].length];

for (int i=0; i< input.length;i++)
{
  for (int j=0; j<input[i].length;j++)
  {
    double cI;
    double cJ;
    if(i==0)
    {
      cI = (1/Math.sqrt(2));
    }
    else
    {
      cI = 1;
    }
    if (j==0)
    {
      cJ = (1/Math.sqrt(2));
    }
    else
    {
      cJ = 1;
    }
```

```
    for (int k=0; k<input.length;k++)
    {
      for(int l=0; l<input[k].length;l++)
      {
        output[i][j] += input[k][l] * Math.cos(
                  (Math.PI/input.length)*i*(k+0.5))*Math.cos(
                  (Math.PI/input[k].length)*j*(l+0.5));
      }
    }
    output[i][j] *= (2.0/(input.length*input[i].length))*cI*cJ;
 }
}
…
```

Zuletzt wird zwischen den beiden Kommunikationsformen unterschieden. Entweder optimiert nur der Zuhörer, basierend auf der eventuell verzerrten verbalen Repräsentation des Sprechers, sein neuronales Netz, oder aber beide optimieren ihre Netze, basierend auf der verbalen Repräsentation des Gegenübers. Damit ist die Abarbeitung eines Simulationsschritts abgeschlossen.

Aktionen nach einem Simulationsdurchlauf

Nach einem erfolgreichen Simulationsdurchlauf wird lediglich der Status in der Klasse **`StatusManager`** auf „*5*“ gesetzt. Eine Beschreibung dieser Klasse befindet sich in Abschnitt B.2.6 dieser Arbeit.

B.2 Grafische Benutzeroberfläche

Das Design der grafischen Benutzeroberfläche in DiaLex ist auf die Erfüllung zweier Hauptziele ausgelegt. Zum einen ist der strukturelle Aufbau sehr einfach gestaltet, um einen mit neuronalen Netzen unerfahrenen Benutzer bzw. einem Benutzer ohne Programmierkenntnissen einen barrierefreien Einstieg zu gewährleisten. Zum anderen garantiert die Oberfläche eine größtmögliche Ausschöpfung aller Funktionalitäten der darunter liegenden Simulationsklassen, um auch komplexe Simulationen erstellen zu können.

In diesem Unterkapitel werden die wichtigsten Klassen der grafischen Benutzeroberfläche erläutert. Die Ausführungen beschränken sich auf die Klassen MainWindow und AnalysisWindow als die beiden Hauptfenster des Simulationstools, die drei wichtigsten Darstellungsfenster der Simulationsergebnisse, das EnvironmentAnalysisFrame, das AvgTableFrame und das ChartFrame sowie die Klasse StatusManager. MainWindow und

EnvironmentAnalysisFrame kommen in vereinfachten Formen bereits in der vorangegangenen Arbeit aus [FK07] vor, sodass Teile dieser Abschnitte an die Ausführungen dieser Arbeit angelehnt sind. Die genannten Klassen werden aus der Sicht der Software-Entwicklung dargestellt, nicht aus der Sicht eines Benutzers. Das dafür erstelle Benutzerhandbuch stellt das achte Kapitel dieser Arbeit dar.

B.2.1 Die Klasse MainWindow

Das Hauptfenster in DiaLex ist eine Instanz der Klasse **`MainWindow`**, welche aus der Klasse **`JFrame`** abgeleitet ist. Die Klasse **`JFrame`** bildet die Hauptfensterklasse innerhalb der Swing-Bibliotheken, deren Hauptfenster im Unterschied zum AWT nur eine einzige Hauptkomponente **`JRootPane`**, die alle anderen Komponenten aufnimmt, besitzt.

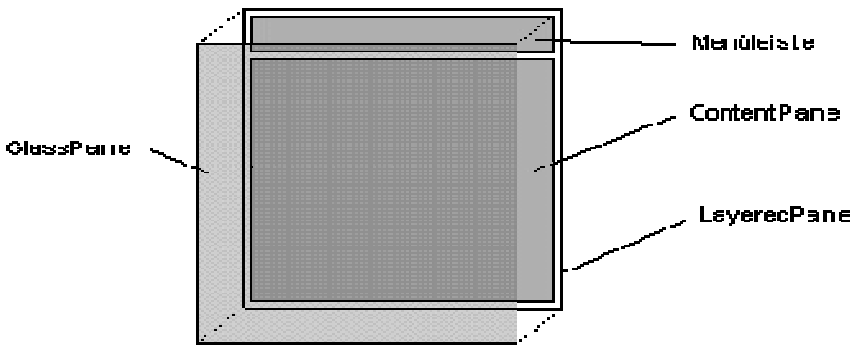


Abb. 139 Struktur einer RootPane
Quelle: [Kru00], S. 755, Abb. 36.2

Während die über der LayeredPane liegende GlassPane in der Regel nicht zur Grafikausgabe genutzt wird, enthält eine aus **`JLayerdPane`** abgeleitete LayeredPane die Menüleiste (abgeleitet aus **`JMenuBar`**) und Dialogelemente innerhalb der ContentPane (abgeleitet aus **`Container`**). Abbildung 140 zeigt das zugehörige Klassendiagramm der Klasse **`MainWindow`**.

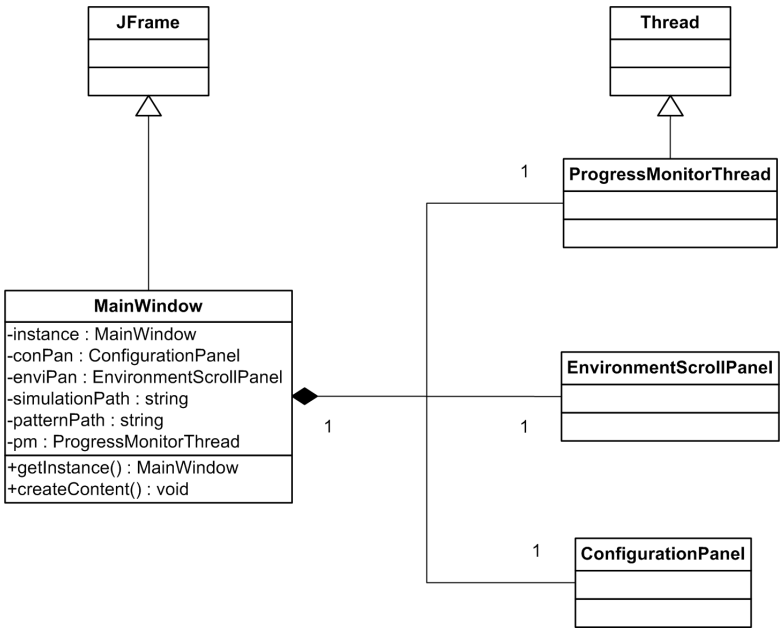


Abb. 140 Klassendiagramm der Klasse MainWindow

Mit Hilfe des Singelton-Patterns kann pro Programmstart immer nur eine Instanz der Klasse **`MainWindow`** erzeugt werden, die mit Hilfe der Methode **`getInstance()`** aufgerufen werden kann. Die ContentPane dieser Instanz besteht aus zwei Komponenten, die jeweils aus **`JPanel`** abgeleitet sind:

- **`ConfigurationPanel conPan`**
  Dieses Panel stellt alle Dialogelemente zur globalen Konfiguration einer Simulation zur Verfügung.

- **`EnvironmentScrollPanel enviPan`**
  In diesem scrollbaren JPanel werden die Attribute und Ereignisse aller erstellten Simulationsumgebungen aufgelistet.

Die Komponenten werden mit Hilfe des *Layoutmanagers* **`GridBagLayout`** innerhalb der ContentPane ausgerichtet, was im Folgenden am Beispiel der **`conPan`** gezeigt wird:

```
…
Container cp = getContentPane();
cp.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
conPan = new ConfigurationPanel();
…
cp.add(conPan,c);
…
```

Durch das Belegen einzelner Attribute der **`GridBagConstraints c`**, wie z. B. x-/y- Koordinate, Höhe und Breite, wird die Lage und Ausrichtung einer Komponente innerhalb der ContentPane festgelegt, sodass der Aufbau des Hauptfensters dem in Abschnitt A.3.2 beschriebenen Entwurfsschema entspricht. Neben den einzelnen Instanzen der Komponenten des Hauptfensters, werden in der Klasse **`MainWindow`** weitere Attribute gehalten:

- **`String simulationPath`**
  In diesem String wird nach dem Abspeichern oder Laden einer Simulation der Verzeichnispfad gehalten, um dem Benutzer bei weiteren Speicher- und Ladevorgängen die Suche des Verzeichnisses zu vereinfachen.

- **`String patternPath`**
  In diesem String wird analog zu **`simulationPath`** der Pfad der eingelesenen visuellen Szenen gespeichert.

- **`ProgressMonitorThread pm`**
  Dieser **`ProgressMonitorThread`** wird beim Start eines Simulationsdurchlaufs sichtbar und zeigt den Fortschritt der Simulation an.

Die wichtigsten Methoden der Klasse **`MainWindow`** sind:

- **`MainWindow getInstance()`**
  Sie gibt die aktuelle Instanz der Klasse **`MainWindow`** zurück.

- **`void createContent(int numberOfEnvironments)`**
  Sie erstellt den kompletten Inhalt der ContentPane des Hauptfensters in Abhängigkeit von der Anzahl der Simulationsinstanzen.

Im Folgenden werden die einzelnen Komponenten des Hauptfensters, die Menüleiste, das Konfigurationspanel und das EnvironmentScrollPanel genauer betrachtet.

Menüleiste

Die Menüleiste in DiaLex wird durch eine Instanz der Klasse **`myMenuBar`** repräsentiert und ist ebenfalls mit Hilfe des Singletonpatterns erstellt worden, sodass nur eine einmalige Instanziierung möglich ist.

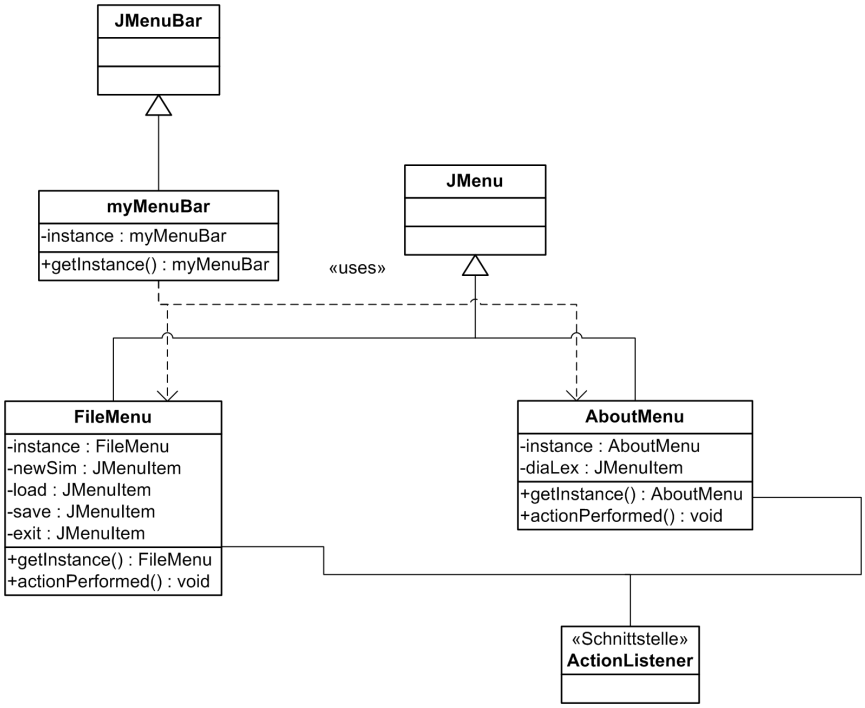


Abb. 141 Klassendiagramm der Klasse myMenuBar

Wie in Abbildung 141 ersichtlich, besteht die Menüleiste aus einem Datei- und Infomenü. Das Dateimenü besteht aus vier verschiedenen Elementen des Typs **`JMenuItem`**. Durch das Implementieren der **`ActionListener`** Schnittstelle und das Überschreiben der Methode **`actionPerformed(ActionEvent event)`** kann jedes Element des Dateimenüs seine folgende Aufgabe erfüllen:

- **`JMenuItem newSim`**
  Dieses Item ist für das Erstellen eines **`NewSimulationDialogs`** verantwortlich und ermöglicht dem Benutzer, eine neue Simulation zu konfigurieren.

- **`JMenuItem load`**
  Das Menüelement **`load`** ist für das Laden einer bereits gespeicherten Simulation zuständig. Eine Instanz der Klasse **`JFileChooser`** übernimmt hierbei die Auswahl eines Dateipfades **`path`** mit Hilfe eines Dialogs. Über **`Simulation.load(path)`**werden alle Attribute der ausgewählten Simulation geladen und das Hauptfenster entsprechend dieser neu gezeichnet, das heisst, dass sich das EnvironmentScrollPanel anhand der in der Simulation abgespeicherten Simulationsumgebungen und deren einzelnen Attributen und Ereignisse neu aufbaut.

- **`JMenuItem save`**
  Dieses Item ermöglicht das Speichern einer Simulation. Wie schon beim Laden einer Simulation übernimmt ein Objekt der Klasse **`JFileChooser`** die Auswahl des Dateipfades, in dem eine Simulation abgespeichert werden soll. Als Dateinamen gibt dieser Dialog **`*.sim`** vor. Dies kann jedoch vom Benutzer beliebig geändert werden. Nach sukzessivem Einlesen der einzelnen Textfelder innerhalb der EnvironmentScrollPanel werden alle Attribute und Ereignisse der einzelnen Simulationsumgebungen innerhalb der Instanz der Klasse **`Simulation`** gesetzt. Durch den Aufruf der Methode **`Simulation.save(path)`** wird die Simulation binär abgespeichert.

- **`JMenuItem exit`**
  Das Menüelement **`exit`** beendet durch den Aufruf von **`System.exit(0)`** das Programm.

- **`JMenuItem diaLex`**
  Dieses Infomenü bietet eine Übersicht über die aktuelle Version von DiaLex.

ConfigurationPanel

Innerhalb der ConfigurationPanel befinden sich wesentliche Konfigurations- und Steuerungselemente für eine erstellte Simulation. Wie in Abbildung 142 ersichtlich, besteht dieser Teil des Hauptfensters aus einem Textfeld, sieben verschiedenen Schaltflächen und zwei Auswahlboxen. Über das **`JTextField simStepsText`** wird die Anzahl der Simulationsschritte manuell eingegeben.

Mit Hilfe des **`JButton selectInputPatternButton`** kann über ein Dateidialogfenster eine CSV-Datei als Menge visueller Eingabemuster eingelesen werden. Dabei wird unter Berücksichtigung des eingelesenen Dateipfades eine Instanz der Klasse **`CSVParser`** des Paketes **`tools`** erzeugt und mit Hilfe der Methode **`readFile()`** eine Menge von Eingabemustern in der aktuellen Simulation als dreidimensionales Array von Double-Werten gesetzt.

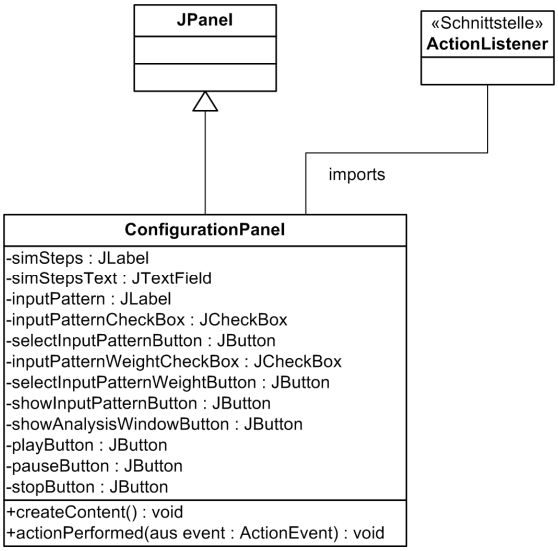


Abb. 142 Klassendiagramm der Klasse ConfiguarationPanel

```
…
String path = choose.getSelectedFile().getPath();
…
CSVParser parse = new CSVParser(path);
try   {
      Simulation.getInstance().setInputPattern(parse.readFile());
      …
      inputPatternCheckBox.setSelected(true);
      selectInputPatternWeightButton.setEnabled(true);
      showInputPatternButton.setEnabled(true);
      …
      }
catch (Exception e) {
      JOptionPane.showMessageDialog(null,
      ConstantStrings.ERRORCSVPARSER, ConstantStrings.ERROR,
      JOptionPane.ERROR_MESSAGE);
}
…
```

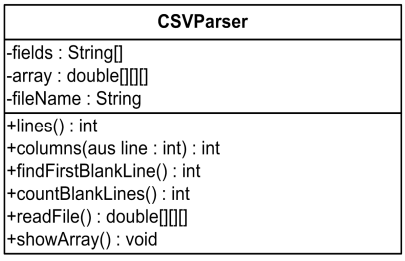


Abb. 143 Klassendiagramm der Klasse CSVParser

Damit eine CSV-Datei erfolgreich geparst werden kann, müssen die einzelnen Eingabemuster durch eine Leerzeile getrennt und von gleicher Länge bzw. Breite sein. Innerhalb der Muster werden die einzelnen Felder, welche aus positiven, reellen Zahlen bestehen, durch „ ; “ oder „ , “ getrennt. Nach einem erfolgreichen Einlesen wird die **`JCheckBox inputPatternCheckBox`** aktiviert und der **`JButton selectInputPatternWeightButton`** freigeschaltet, über den in gleicher Weise relative Häufigkeitsgewichtungen für die eingelesenen visuellen Eingabemuster ebenfalls über eine CSV-Datei eingelesen werden können. War dieses erfolgreich, so wird die **`JCheckBox inputPatternWeightCheckBox`** ebenfalls aktiviert.

Außerdem wird nach dem erfolgreichen Einlesen visueller Eingabemuster der **`JButton showInputPatternButton`** aktiviert und durch die Erzeugung einer Instanz der Klasse **`ShowPatternDialog`** wird eine visuelle Übersicht auf die geladene Menge von Eingabemustern erzeugt. Jeder **`ShowPatternDialog`** besitzt ein Objekt der Klasse **`PatternScrollPanel`**, um eine Analyse größerer Mengen von Eingabemustern zu ermöglichen. Dieses scrollbare JPanel ist wiederum ein Container für eine Instanz der Klasse **`PatternPanel`**, welche alle Eingabemuster in Form von **`SinglePatternPanel`**-Objekten beinhaltet. Wurden relative Häufigkeiten für die Eingabemuster eingelesen, so werden diese ebenfalls in Form von Instanzen der Klasse **`JLabel`** unterhalb der Eingabemuster angezeigt.

```
…
for (int i=0;i<inputPattern.length;i++)
{
 singPat[i] = new
SinglePatternPanel(inputPattern[i],squaresize,i);
 add(singPat[i]);

 if (Simulation.getInstance().getInputPatternFrequency()!=null)
 {
   add(new JLabel(ConstantStrings.INPUTPATTERNFREQUENCY
   +Simulation.getInstance().getInputPatternFrequency()[i]));
 }
}
…
```

Das Objekt **`SinglePatternPanel`** besitzt jeweils ein zweidimensionales Array mit den reellen Zahlenwerten des Eingabemusters, eine ganzzahlige Angabe **`squaresize`** über die Größe der zu zeichnenden Quadrate und den jeweiligen Index **`i`** eines Musters als Attribute, welche es zu einer zweidimensionalen Darstellung der Muster in Form von Quadraten nutzt. Das eigentliche Zeichnen dieser Werte erfolgt mit Hilfe der Klasse **`GraphicPattern`**, innerhalb derer eine Überschreibung der aus dem AWT-Toolkit bekannten Methode **`paint()`** erfolgt.

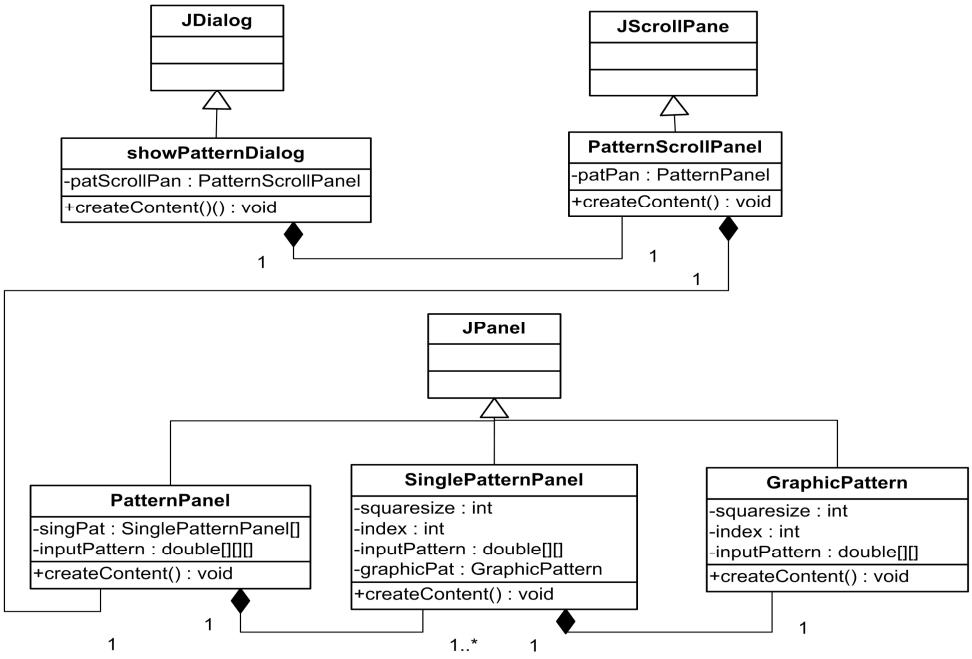


Abb. 144 Klassendiagramm der Klasse ShowPatternDialog

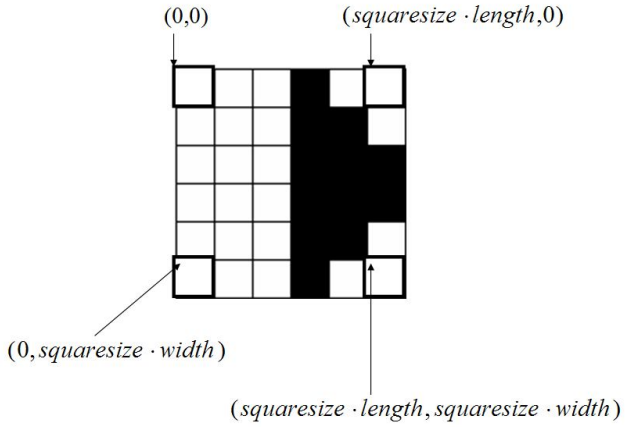


Abb. 145 Darstellung eines Eingabemusters

Um das Eingabemuster in seiner Länge und Breite korrekt darzustellen, werden die einzelnen reellen Werte in zwei Schleifen durchlaufen und die Anfangskoordinaten der einzelnen Quadrate, wie in Abbildung 145 dargestellt, festgelegt. Der Farbton, welches jedes einzelne Quadrat annimmt, hängt von dem reellen Zahlenwert des Eingabemusters an der darzustellenden Position ab und wird mit Hilfe der statischen Methode **`Color getColor(double inputvalue)`** der Klasse **`Tools`** ermittelt.

```
…
g.setColor(Tools.getColor(inputPattern[i][j]));
…
```

Diese statische Methode nutzt den Konstruktor der Klasse **`Color`**, indem sie bei Übergabe eines Eingabemusterwertes von 1.0 alle RGB-Werte auf 0.0 setzt, welches der Definition der Farbe Schwarz entspricht. Dementsprechend werden alle Grautöne bis hin zur Farbe Weiß, welche dem Eingabemusterwert von 0.0 bzw. den RGB-Werten von 1.0 entspricht, erzeugt. Mit Hilfe der Klasse **`SinglePatternPanel`** erhält jedes Eingabemuster einen Rahmen mit seinem jeweiligen Index und wird in einem Containerobjekt der Klasse **`PatternPanel`** aufgenommen. Weiterhin enthält die ConfigurationPanel den **`JButton showAnalysisWindowButton`**, der das Analysefenster öffnet. Dies wird im nachfolgenden Abschnitt genauer beschrieben.

Zuletzt besitzt eine ConfigurationPanel Steuerungselemente in Form eines Start-, Pause- und Stoppbuttons, welche eine Simulation jeweils starten, pausieren, fortsetzen und beenden. Nachdem eine Simulation vollständig konfiguriert wurde ist nur der Startbutton auf aktiv gesetzt. Die Verwaltung des eigentlichen Status der Simulation geschieht in der Klasse **`StatusManager`**, die in Abschnitt B.2.6 genauer erläutert wird. Durch Anklicken des Startbuttons wird zunächst geprüft, ob alle Parameter der erstellten Simulation korrekt eingegeben sind, d. h. es wird geprüft ob:

- alle Parameter syntaktisch und gemäß ihres Typs korrekt eingegeben sind,
- eine nicht leere Menge von Eingabemustern eingelesen wurde,
- die Struktur der Menge der eingelesenen Eingabemuster auf die angegebene Struktur der neuronalen Netze passt.

Für den Fall, dass eine dieser Prüfungen fehlschlägt, öffnet sich ein Fehlerdialog, der auf die Art des bestehenden Fehlers hinweist. Es wird dann kein Simulationsdurchlauf gestartet. Falls alle Daten korrekt eingegeben wurden, wird die eigentliche Simulation erzeugt und gestartet, was zur Folge hat, dass der Startknopf inaktiv, Pause- und Stoppbutton aktiv werden. Um eine Simulation zu erzeugen, müssen sowohl das Simulationsschritttextfeld und die geparste Menge der Eingabemuster samt eventuellen Häufigkeiten als auch die Textfelder mit den Konfigurationsdaten und Events der einzelnen Simulationsumgebungen (siehe Unterabschnitt „*EnvironmentScrollPanel*“) eingelesen werden, sodass die entsprechenden Werte an die Instanz der Klasse **`Simulation`** weitergegeben werden können. Bevor eine Simulation ausgeführt werden kann, müssen zuerst die einzelnen Simulationsumgebungen generiert werden.

```
…
Simulation.getInstance().generateEnvironments();
…
```

Das Starten einer Simulation erfolgt durch den Aufruf der Methode

```
…
Simulation.getInstance().simulationStart();
…
```

innerhalb eines eigenen Threads. Ein laufender Thread kann durch Anklicken des Pausebuttons angehalten werden und es kann eine Analyse der einzelnen Simulationsumgebungen durchgeführt werden. Durch erneutes Anklicken der Schaltfläche wird die Simulation fortgeführt. Durch Anklicken des Stoppbuttons oder nach erfolgreichem Durchlauf durch alle Simulationsschritte wird der aktuelle Thread und die damit verbundene aktuell laufende Simulation beendet und sowohl der Pause- als auch der Stoppbutton wieder auf inaktiv gesetzt.

EnvironmentScrollPanel

Die Aufgabe der EnvironmentScrollPanel ist es, sowohl eine Übersicht über die erstellten Simulationsumgebungen zu geben, als auch eine Veränderbarkeit dieser im Einzelnen zu gewährleisten. Abbildung 146 zeigt den Aufbau dieses JScrollPanel in Form eines Klassendiagramms.

Jede einzelne Simulationsumgebung wird mit Hilfe eines Objekts der Klasse **`EnvironmentPanel`** dargestellt und in ein Array innerhalb einer Instanz der Container-Klasse **`EnvironmentListPanel`** abgelegt. Damit beliebig viele Simulationsumgebungen erstellt und betrachtet werden können, wird dieses EnvironmentListPanel einem scrollbaren Objekt der Klasse **`EnvironmentScrollPanel`** zugeordnet. Jede Simulationsumgebung wird durch ihre Attribute und Ereignisse definiert. Daher besteht ein EnvironmentPanel aus einer Instanz der Klasse **`EnvironmentAttributes`**, in dem sämtliche Simulationsangaben zu den Gebieten Simulationsumgebung, Agenten, Kommunikation und Störfaktoren und den Events angezeigt und verändert werden können, und aus einer Instanz der Klasse **`EnvironmentEventsScrollPanel`**, die als ScrollPane fungiert und in einem Objekt der Klasse **`EnvironmentEvents`** alle Ereignisse auflistet, die jeweils durch eine Instanz der Klasse **`EnvironmentEvent`** dargestellt werden.

Neben der Möglichkeit die vor einem Simulationsdurchlauf definierten Ereignisse zu verändern und neue mit dem **`AddEventDialog`** hinzuzufügen, können an dieser Stelle ebenfalls während eines Simulationsdurchlaufs alle noch abzuarbeitenden Ereignisse und alle bereits ausgelösten Ereignisse aufgelistet werden. Der Benutzer kann so jede Umgebung individuell konfigurieren. Die syntaktische und strukturelle Prüfung der einzelnen Eingaben erfolgt, wie bereits erwähnt, nach dem Drücken des Startbuttons.

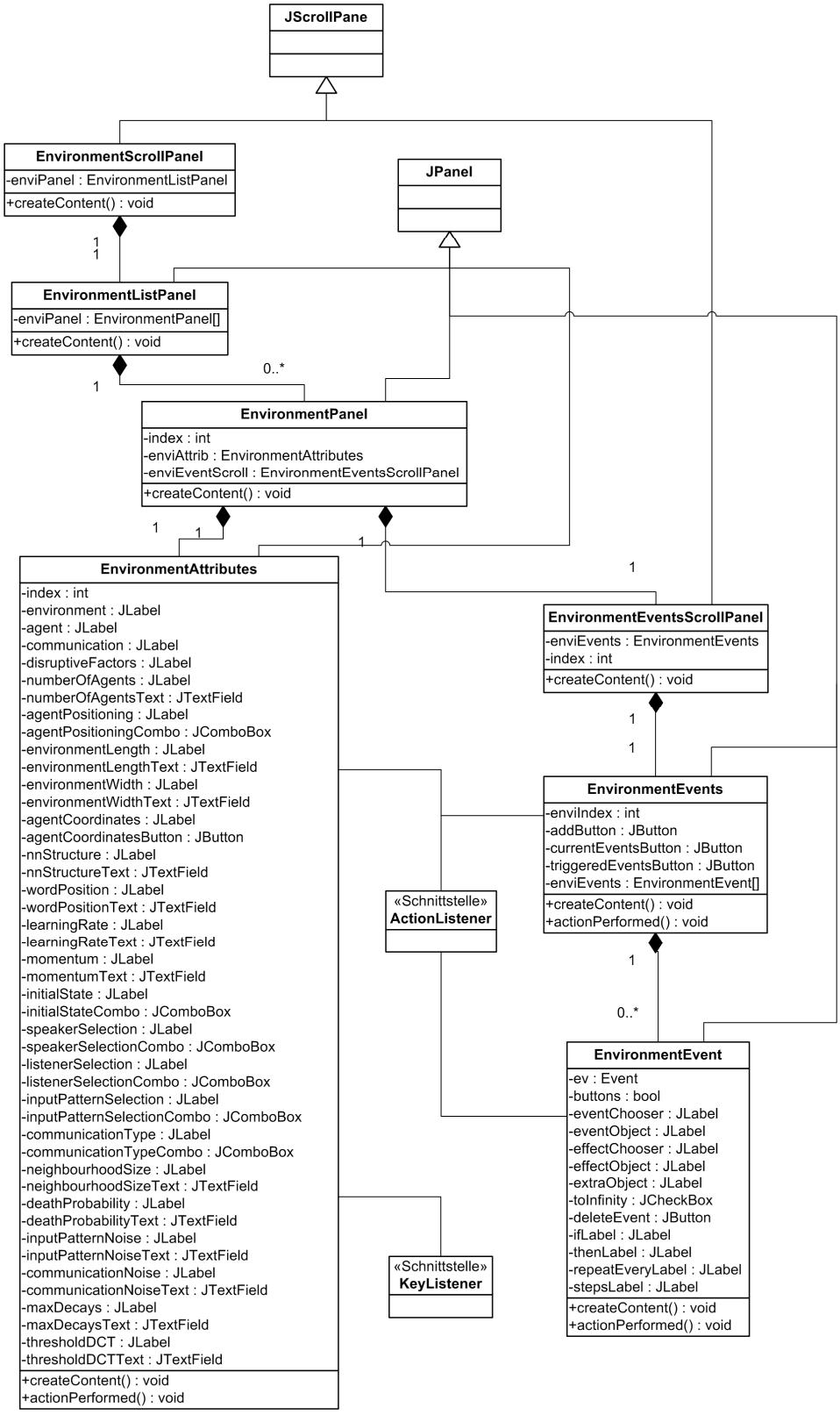


Abb. 146 Klassendiagramm der Klasse EnvironmentScrollPanel

B.2.2 Die Klasse AnalysisWindow

Das Analysefenster stellt das zweite Hauptfenster in DiaLex dar und ist für die Auflistung der während einer Simulation durch Ereignisse gesammelten und in der Datenbank (vgl. Abschnitt B.1.3) gehaltenen Datensätze konzipiert. Für diese Aufgabe ist es in vier ineinander verschachtelte Klassen aufgeteilt. Die Hauptklasse **`AnalysisWindow`** ist wie das Hauptfenster mit dem Singleton-Pattern erstellt worden und wird dadurch nur einmal beim Programmstart instanziiert. Es enthält nur eine Unterklasse:

- **`AnalysisListScrollPanel analistscroll`**
  Das ScrollPanel, in dem die aufgelisteten Simulationsergebnisse scrollbar aufgelistet werden.

Die wichtigsten Methoden der Klasse sind im Folgenden aufgelistet:

- **`AnalysisWindow getInstance()`**
  Sie gibt die aktuelle Instanz der Klasse **`AnalysisWindow`** zurück.

- **`void createContent()`**
  Diese Methode generiert den Inhalt des Analysefensters.

AnalysisListScrollPanel

Durch die Klasse **`AnalysisListScrollPanel`** und den in ihr enthaltenen Klassen wird der Fensterinhalt des Analysefensters erzeugt. Dies geschieht durch den Aufruf der Methode **`createContent()`**. Das so erzeugte ScrollPanel füllt den gesamten Fensterinhalt des Analysefensters aus und erzeugt bei Bedarf die nötigen Scrollbars am rechten und unteren Rand des Fensters. **`AnalysisListScrollPanel`** besteht aus einem Objekt des Typs **`AnalysisListsPanel`**, das für jede Datenreihe eine Instanz der Klasse **`AnalysisListPanel`** erstellt. Diese werden mit Hilfe eines GridLayouts im Analysefenster aufgelistet:

```
…
GridLayout  myGridLayout = new GridLayout(0,1);
myGridLayout.setVgap(5);
setLayout(myGridLayout);

anaPanel = new AnalysisListPanel[numberOfAnalysisLists];
for (int i=0;i<numberOfAnalysisLists;i++)
{
      anaPanel[i] = new AnalysisListPanel(i);
      add(anaPanel[i]);
}
…
```

Innerhalb eines **`AnalysisListPanel`** werden die für jede Wertereihe gesammelten Werte benutzergerecht aufbereitet und darstellt. So werden beispielsweise einfache Wertereihen des Typs **`double`** sowohl direkt im Analysefenster aufgelistet und können bei Bedarf in einem ChartPanel (vgl. Abschnitt B.2.4) betrachtet werden.

Im Gegensatz dazu können komplexere Datensätze wie zum Beispiel Schnappschüsse ganzer Simulationsumgebungen nicht ohne weiteres aufgelistet werden. Vielmehr wird in diesem Fall eine Liste von Schaltflächen erzeugt, die für jeden Schnappschuss eine Instanz der Klasse **`EnvironmentAnalysisFrame`** erzeugen und sichtbar machen. (vgl. Abschnitt B.2.3) Die nachfolgende Abbildung zeigt das Klassendiagramm des Analysefensters und die in ihm enthaltenen Bestandteile.

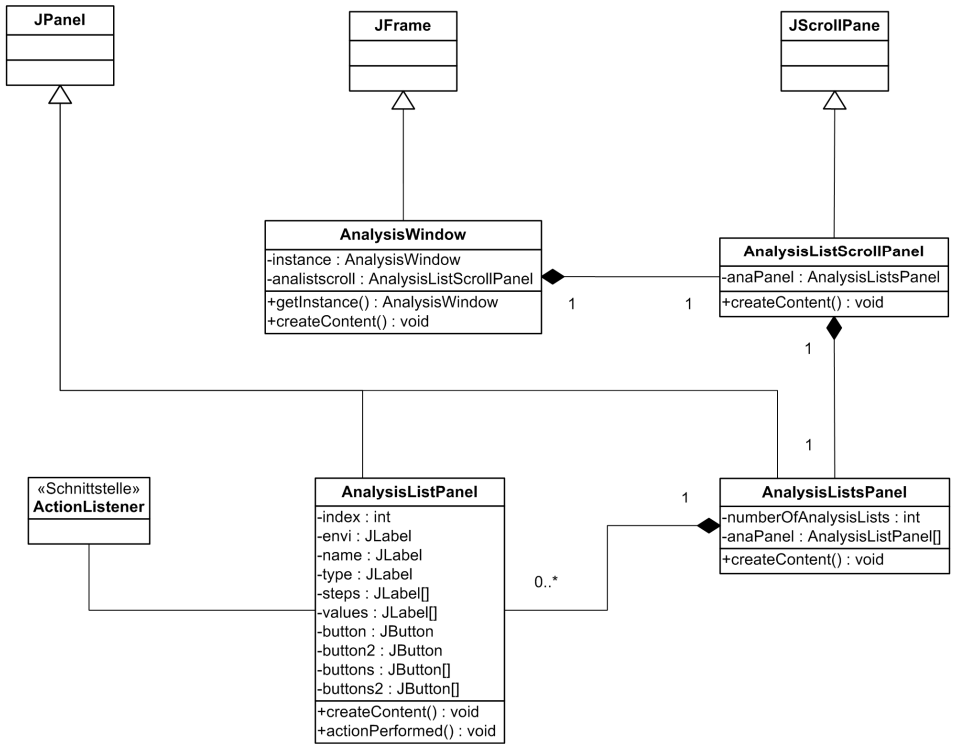


Abb. 147 Klassendiagramm der Klasse AnalysisWindow

B.2.3 Die Klasse EnvironmentAnalysisFrame

In DiaLex ist jedes Agentenfenster eine Instanz der Klasse **`EnvironmentAnalysisFrame`**, welche in Abbildung 148 als Klassendiagramm dargestellt ist.

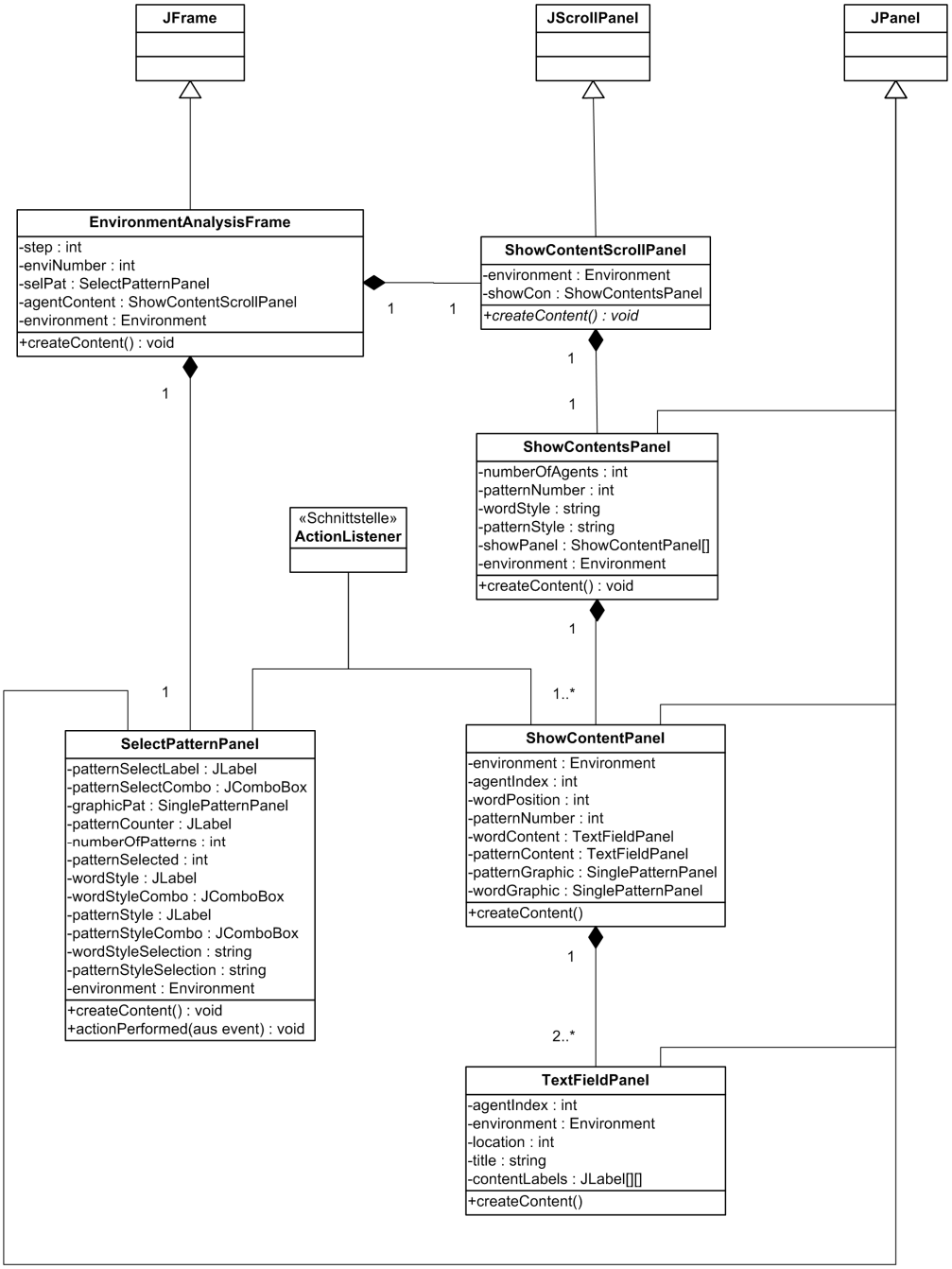


Abb. 148 Klassendiagramm der Klasse EnvironmentAnalysisFrame

Ein Objekt der Klasse **`EnvironmentAnalysisFrame`** enthält dabei folgende Attribute:

- **`Environment envi`**
  Dies ist der Schnappschuss der Simulationsumgebung, die das Frame darstellen soll. Es stammt aus der Datenbank und wird beim Aufruf des Fensters übergeben.

- **`int step`**
  Diese Variable hält den zur Simulationsumgebung zugehörigen Simulationsschritt.

- **`int enviNumber`**
  Diese Variable hält die Nummer der zu diesem Schnappschuss zugehörigen Simulationsumgebung.

- **`SelectPatternPanel selPat`**
  Dies ist das für die Konfiguration der Analyse notwendige Objekt der Klasse **`SelectPatternPanel.`**

- **`ShowContentScrollPanel agentContent`**
  Dieses ScrollPanel enthält die visuelle Auflistung einer Simulationsumgebung in Form einer Instanz der Klasse **`ShowContentScrollPanel.`**

Durch den Aufruf der Methode **`createContent()`** wird das Fenster mit den in ihm enthaltenen zwei Unterklassen gezeichnet. Sie werden im Folgenden genauer erläutert.

SelectPatternPanel

Eine Instanz der Klasse **`SelectPatternPanel`** bietet Benutzern zum einen die Möglichkeit, ein spezielles Eingabemuster auszuwählen, um die Wörter bzw. Deduktionen der einzelnen Agenten zu diesem Muster zu untersuchen. Zum anderen kann der Benutzer zwischen verschiedenen Darstellungsformen der Wörter bzw. Abzeichnungen wählen.

Die wichtigsten Attribute und deren Funktionalitäten im Überblick:

- **`JComboBox patternSelectCombo`**
  Diese Combobox ermöglicht die Auswahl eines Eingabemusters aus der Menge der visuellen Szenen.

- **`SinglePatternPanel graphPat`**
  Dieses bereits in Abschnitt B.2.1 erläuterte Panel bietet eine grafische Darstellung des in der Combobox **`patternSelectCombo`** aktuell ausgewählten Eingabemusters. Über die **`ActionListener`** Schnittstelle wird ein neu ausgewähltes Eingabemuster direkt neu gezeichnet.

- **`JComboBox wordStyleCombo`**
  Diese Combobox dient der Auswahl von verschiedenen Darstellungsmöglichkeiten der Wörter der Agenten. Sie umfasst die Punkte „standard“, „rounded“, „literal“, „dct graphical“ und „graphical“. Je nach Auswahl werden über die Schnittstelle **`ActionListener`** die Wörter der Agenten innerhalb des ShowContentScrollPanel neu dargestellt (siehe Abbildung 149).

- **`JComboBox patternStyleCombo`**
  Dies ist eine Combobox zur Auswahl von verschiedenen Darstellungsmöglichkeiten der Deduktionen oder Abzeichnungen der Agenten. Sie umfasst die Punkte „standard“, „rounded“ und „graphical“. Eine Darstellung als literale oder einer mit Hilfe der diskreten Kosinustransformation umgewandelten Form erscheint hier nicht sinnvoll. Je nach Auswahl werden über die Schnittstelle **`ActionListener`** die Deduktionen der Agenten innerhalb des ShowContentScrollPanel neu dargestellt (vgl. Abb. 149).

Abbildung 149 gibt eine Übersicht über die verschiedenen Darstellungsmöglichkeiten der Wörter bzw. Deduktionen einzelner Agenten.

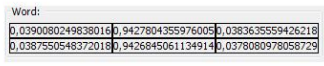
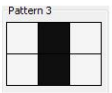
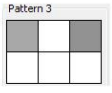
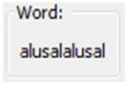
Combobox	Beschreibung	Screenshot
standard	Aktivierungswerte der Neuronen ungerundet	
rounded	Aktivierungswerte der Neuronen auf zwei Nachkommastellen gerundet	
graphical	Aktivierungswerte der Neuronen grafisch dargestellt	
dct graphical	Aktivierungswerte der Neuronen grafisch nach einer diskreten Kosinustransformation dargestellt	
literal	Aktivierungswerte der Neuronen als Silben dargestellt	

Abb. 149 Darstellungsformen der Wörter und Deduktionen

ShowContentScrollPanel

Die Klasse **`ShowContentScrollPanel`** repräsentiert die zweite Komponente des Analysefensters und ist für die Auflistung der Wörter und Deduktionen für ein bestimmtes Eingabemuster aller Agenten innerhalb einer Simulationsumgebung verantwortlich. Ein Objekt der Klasse **`ShowContentScrollPanel`** gibt einen Rahmen für eine Instanz der Klasse **`ShowContentsPanel`** vor, welche die Wörter und Deduktionen eines jedes Agenten in Form eines Array von **`ShowContentPanel`** Objekten enthält. Neben diesem Array besitzt die Klasse **`ShowContentsPanel`** als weitere Attribute einen Verweis auf den Schnappschuss der Simulationsumgebung, die Anzahl an Agenten, die Nummer des ausgewählten Eingabemusters und Angaben über die Darstellungsform der Wörter bzw. Deduktionen in Form des Strings **`wordStyle`** bzw. **`patternStyle`**, welche durch den Konstruktor der Klasse in

```
…
public ShowContentsPanel(     Environment environment,
      int patternNumber, String wordStyle, String patternStyle)
      {
            this.environment = environment;
            this.numberOfAgents = environment.getNumberOfAgents();
            this.patternNumber = patternNumber;
            this.wordStyle=wordStyle;
            this.patternStyle=patternStyle;
            createContent(patternNumber,wordStyle,patternStyle);
      }
…
```

gesetzt werden. Durch den Aufruf der Methode **`createContent( patternNumber,wordStyle,patternStyle)`** innerhalb des Konstruktors wird dieses Panel gezeichnet. Jeder Agent einer Simulationsumgebung erhält dabei ein Objekt der Klasse **`ShowContentPanel`**, welches mit Hilfe des Layoutmanagers **`GridLayout`** innerhalb von ShowContentsPanels angeordnet wird:

```
…
GridLayout myGridLayout = new GridLayout(0,1);
myGridLayout.setVgap(20);
setLayout(myGridLayout);

showPanel = new ShowContentPanel[numberOfAgents];
for (int i=0;i<numberOfAgents;i++)
{
 showPanel[i] = new ShowContentPanel(index,i);
 showPanel[i].createContent(patternNumber,wordStyle,patternStyle);
 add(showPanel[i]);
}
…
```

Der Inhalt des ShowContentPanel richtet sich dabei nach der aktuell ausgewählten visuellen Szene und der Darstellungsform des Wortes bzw. der Deduktion. Während die Koordinaten eines Agenten in der Simulationsumgebung und die Anzahl der Interaktionen als Sprecher und Zuhörer stets gleich angezeigt werden, richten sich die Darstellungen des Wortes und der Deduktion nach den Wünschen des Benutzers. Wurde „standard“, „rounded“ oder „literal“ als Darstellungsform für das Wort eines Agenten bzw. „standard“ oder „rounded“ als Darstellungsform für die Deduktion ausgewählt, erfolgt deren Darstellung in Form zweier Objekte der Klasse **`TextFieldPanel`**.

```
…
wordContent = new   TextFieldPanel(enviIndex,agentIndex,
      Simulation.getInstance().getWordpositions()[enviIndex],
      ConstantStrings.WORD);
wordContent.createContent(patternNumber,wordStyle);
…

…
patternContent = new TextFieldPanel(enviIndex,agentIndex,
      Simulation.getInstance().getEnvironments()[enviIndex]
      .getAgents()[agentIndex].getLayers().length-1,
      ConstantStrings.DRAWING);
patternContent.createContent(patternNumber,patternStyle);
…
```

Ein Objekt der Klasse **`TextFieldPanel`** besteht aus einem zweidimensionalen Array von JTextfeldern, in denen die Aktivierungen der Neuronen der Wortschicht bzw. der Ausgabeschicht innerhalb dieser Textfelder ausgegeben werden (siehe Abbildung 148).

Falls für die Darstellungsform des Wortes „literal“ ausgewählt wurde, erfolgt die Darstellung innerhalb eines Strings, indem die Aktivierungen der Neuronen der Wortschicht innerhalb einer Schleife, mit Hilfe der statischen Methode **`getSyllable(double in)`** der Klasse **`Tools`**, auf konkrete Silben abgebildet werden. Eine Veränderung dieser Abbildung auf Silben kann der Benutzer in der aktuellen Version nur im Quellcode durchführen. [28]

[28] Da die literale Darstellungsform leicht den Eindruck einer verwendeten Schriftsprache der Agenten erwecken kann, wird in den Simulationsauswertungen in Kapitel 4 bewusst auf diese Form der Wortrepräsentation verzichtet. Dennoch bleibt sie dem Nutzer als eine weitere Möglichkeit der Wortdarstellung in DiaLex erhalten.

```
…
public static String getSyllable (double in)
{
      String out = new String("");

      if (in < 0.1)  out += "al"; else
      if (in < 0.2)  out += "as"; else
      if (in < 0.3)  out += "es"; else
      if (in < 0.4)  out += "el"; else
      if (in < 0.5)  out += "il"; else
      if (in < 0.6)  out += "is"; else
      if (in < 0.7)  out += "os"; else
      if (in < 0.8)  out += "ol"; else
      if (in < 0.9)  out += "ul"; else
      if (in < 1.0)  out += "us";

      return out;
}
…
```

Wählt ein Benutzer als Darstellungsform der Deduktion „graphical“ oder „dct graphical“, so erfolgt die Darstellung in Form der in Abschnitt B.2.1 beschriebenen grafischen Zeichnung der Aktivierungen der Ausgabeschicht innerhalb der Klasse **`ShowContentPanel`**.

```
…
patternGraphic = newSinglePatternPanel(pattern,squaresize,
                                       patternNumber);
…
```

Das zu zeichnende Pattern nach einer Umwandlung mit der diskreten Kosinustransformation wird dabei mit Hilfe der Methode **`double[][] forwardDCT(double[][] input)`** der Klasse **`DCT`** im Paket **`Tools`** umgewandelt.

```
…
pattern = DCT.forwardDCT (pattern);
…
```

Ihre Funktionsweise wird im Abschnitt 3.5.5 dieser Arbeit gezeigt.

B.2.4 Die Klasse ChartFrame

Zur Darstellung zweidimensionaler Graphen der gesammelten Simulationsdaten benutzt DiaLex die vordefinierte Klasse **`ChartPanel`** des Frameworks JFreeChart, die in ein JFrame eingesetzt wird. Ein ChartFrame besteht aus den nachfolgenden Bestandteilen:

- **`JComboBox addCombo`**
  Diese Combobox ermöglicht das Hinzufügen von weiteren Datenreihen als Funktionen im ChartPanel.

- **`JComboBox removeCombo`**
  Diese Combobox ermöglicht das Entfernen bereits angezeigter Datenreihen aus dem ChartPanel.

- **`ChartPanel chart`**
  Diese Instanz der vorgefertigten Klasse ChartPanel zeichnet die ausgewählten Datenreihen in ein variabel angepasstes Koordinatensystem.

Bei der Instanziierung der Klasse wird die Klassenmethode **`createContent()`**ausgeführt, die den gesamten Fensterinhalt berechnet und zeichnet. Um sämtliche darstellbaren Datenreihen der Auswahl innerhalb der ComboBoxen hinzuzufügen, enthält die Methode das folgende Stück Quellcode, das die Datenreihen befüllt auf die beiden ComboBoxen verteilt:

```
…
for (int i=0;i<Simulation.getInstance()
                              .getDb().getNumberOfLists();i++)
{
  DbList list = Simulation.getInstance().getDb().getList(i);
  if(list.getType().equals("double"))
  {
    boolean isAlreadyIn = false;
    for (int j=0; j<chart.getChart().getXYPlot()
                                    .getDataset().getSeriesCount();
                                    j++)
    {
      if (chart.getChart().getXYPlot().getDataset()
                    .getSeriesName(j).equals("E:"+list.getEnvi()+
                                             "
                                        "+list.getName()))
      {
        isAlreadyIn=true;
        break;
      }
    }
```

```
    if (!isAlreadyIn)
    {
      addCombo.addItem("E:"+list.getEnvi()+" "+list.getName());
    }
    else
    {
      removeCombo.addItem("E:"+list.getEnvi()+" "+list.getName());
    }
  }
}
…
```

Bereits im ChartPanel dargestellte Datenreihen werden so der ComboBox **`removeCombo`** zugeführt, während alle anderen Datenreihen des Typs **`double`** aus der Datenbank in der ComboBox **`addCombo`** sichtbar sind. Da beide ComboBoxen den **`ActionListener`** implementierten, führt eine Auswahl einer Datenreihe in einer der beiden Boxen stets zu einem aktualisierten Neuzeichnen, das gewünschte Veränderungen darstellt.

Das Objekt der Klasse **`ChartPanel`** wird einem ChartFrame bereits bei der Initialisierung in der Klasse **`AnalysisListPanel`** des Analysefensters aus einer Datenbankliste erzeugt und im Konstruktor der Klasse **`ChartFrame`** übergeben.

```
…
XYSeries xy = new XYSeries("E:"+list.getEnvi()+
                                          " "+list.getName());
for(int a=0;a<list.size();a++)
{
  DbObject dbo = (DbObject) ((ArrayList)
                                      list.getDbObjects()).get(a);
  xy.add(dbo.getStep(),(Double) dbo.getOb());
}
XYSeriesCollection xyc = new XYSeriesCollection(xy);
JFreeChart jfc = ChartFactory.createXYLineChart(" ", // Title
                     "step", // x-axis Label
                     "value", // y-axis Label
                     xyc ,// Dataset
                     PlotOrientation.VERTICAL, //Plot Orientation
                     true, // Show Legend
                     true, // Use tooltips
                     false // Configure chart to generate URLs?
                  );

XYPlot plot = jfc.getXYPlot();
ValueAxis xaxis = plot.getDomainAxis();
```

```
ValueAxis yaxis = plot.getRangeAxis();
xaxis.setMinimumAxisValue(0);
yaxis.setMinimumAxisValue(-0.05);
yaxis.setMaximumAxisValue(1.05);
ChartPanel cp = new ChartPanel(jfc);
ChartFrame pd = new ChartFrame(cp);
…
```

Es zeigt sich eine stark ineinander verschachtelte Struktur der Klassen **`ChartFrame`**, **`ChartPanel`**, **`JFreeChart`**, **`XYSeriesCollection`**, und **`XYSeries`**, die in der nachfolgenden Abbildung noch einmal verdeutlicht wird.

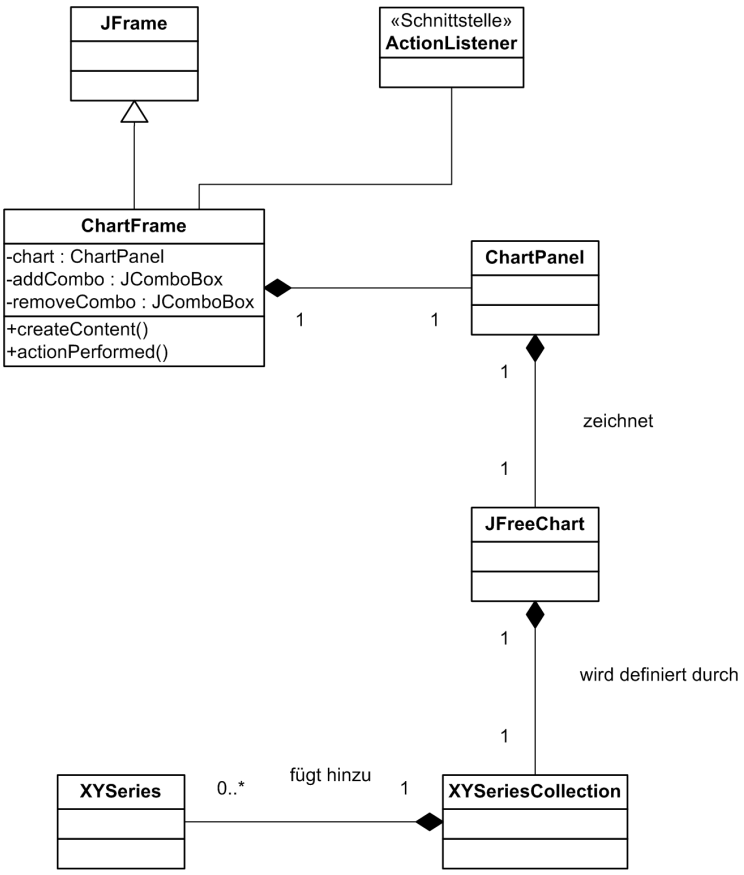


Abb. 150 Klassendiagramm der Klasse ChartFrame

Die Klasse **`XYSeries`** stellt dabei eine Reihe von X-Y-Koordinaten zur Verfügung, die mit mehreren anderen in einer **`XYSeriesCollection`** zusammengefasst wird. **`JFreeChart`** kreiert einen Graphen aus der Menge der Datenreihen, der in **`ChartPanel`** in ein Panel eingepasst wird, um in Fenstern dargestellt werden zu können. Schließlich wird dieses Panel in **`ChartFrame`** Bestandteil eines Fensters.

B.2.5 Die Klasse AvgTableFrame

Die Klasse **`AvgTableFrame`** übernimmt die Darstellung zweidimensionaler Kreuztabellen, die während eines Simulationsdurchlaufs berechnet und gesammelt werden. Die Klasse enthält die folgenden Attribute:

- **`double[][] array`**
  Dieses zweidimensionale Array des Typs **`double`** enthält die darzustellenden Werte der Kreuztabelle. Es wird bei der Initialisierung der Klasse übergeben.

- **`String name`**
  Diese Zeichenkette speichert den Namen der gesammelten Werte und wird ebenfalls bei der Instanziierung der Klasse übergeben.

- **`int step`**
  Analog zu den vorherigen Attributen speichert diese Variable den zugehörigen Simulationsschritt.

- **`int env`**
  Diese Variable speichert entsprechend die zugehörige Simulationsumgebung.

- **`JLabel[][] labels`**
  Dieses zweidimensionale Array von Labels wird in der Methode **`createContent()`** während der Initialisierung der Klasse erzeugt und mit den Werten aus **`array`** befüllt. Dies wird im Folgenden gezeigt:

```
…
for(int i=0;i<array.length;i++)
{
  for(int j=0; j<array[i].length;j++)
  {
    labels[i][j] = new JLabel(""+rounded.format(array[i][j]));
    c.gridx=j+1;
    c.gridy=i+1;
    labels[i][j].setPreferredSize(dim);

labels[i][j].setForeground(Tools.getRainbowColor(array[i][j]));
    add(labels[i][j],c);
  }
}
…
```

Es wird sich der Methode **`getRainbowColor(double in)`** aus der Klasse **`Tools`** bedient. Diese errechnet zu einem Wert aus dem Definitionsbereich *D=[0;1]* eine im Farbspektrum zugehörige Farbe und gibt diese zurück. Somit sind Auffälligkeiten innerhalb einer Kreuztabelle für den Benutzer leichter zu erkennen. Im Folgenden wird der Quellcode der Methode vorgestellt:

```
…
float r=(float) ((0.5*Math.cos(2.0*Math.PI*in) + 0.5));
float g=(float) ((0.5*Math.cos(2.0*Math.PI*(in + 1.0/3.0))+0.5));
float b=(float) ((0.5*Math.cos(2.0*Math.PI*(in - 1.0/3.0))+0.5));

Color color = new Color(r,g,b);
…
```

Objekte der Klasse **`AvgTableFrame`** werden ebenfalls in der Klasse **`AnalysisListPanel`** des Analysefensters mit den Daten eines Elements einer Datenbankliste instanziiert, wie im nachfolgenden Quellcodeauszug gezeigt wird.

```
…
double[][] arr = (double[][]) list.getElement(
                                    new Integer(step).intValue());
AvgTableFrame ad = new AvgTableFrame(arr,list.getName(),
                     new Integer(step).intValue(),list.getEnvi());
…
```

B.2.6 Die Klasse StatusManager

Die Klasse **`StatusManager`** ist eine relativ simpel aufgebaute Klasse, die nicht direkt der grafischen Benutzeroberfläche zugeordnet werden kann, da sie keinerlei grafische Darstellung oder Sichtbarkeit besitzt. Da sie jedoch nicht für die Durchführung von Simulationen notwendig ist, darf sie auch nicht dem Simulationsbereich zugeordnet werden. Diese mit dem Singleton-Design-Pattern erzeugte Klasse übernimmt vielmehr die Verwaltung und koordiniert die Auswirkungen des Simulationsstatus auf die grafische Benutzeroberfläche. Der Simulationsstatus wird in der einzigen Klassenvariable **`int status`** gespeichert. Während der Erstellung und Durchführung von Simulationen setzt die Klasse **`Simulation`** an verschiedenen Stellen einen neuen Status und **`StatusManager`** aktiviert bzw. deaktiviert Schaltflächen der Benutzeroberfläche. Im Folgenden werden die verschiedenen Status einer Simulation aufgelistet:

0	Die Simulation ist nicht fertig konfiguriert.
1	Die Simulation ist fertig konfiguriert und kann gestartet werden.
2	Die Simulation läuft.
3	Die Simulation pausiert.
4	Die Simulation wurde nach einer Pause fortgesetzt.
5	Die Simulation wurde abgebrochen oder beendet.
6	Die Simulation wurde aus einer Datei geladen.

Abb. 151 Die Status einer Simulation

Entsprechend des Status werden im Hauptfenster mindestens 28 und je nach Anzahl der Simulationsumgebungen und Ereignisse leicht über 100 Veränderungen durchgeführt, die an dieser Stelle aus Platzgründen nicht angeführt werden.

C. DiaLex - User Guide

Starting DiaLex you will find the *main window* as it is shown in figure 152.

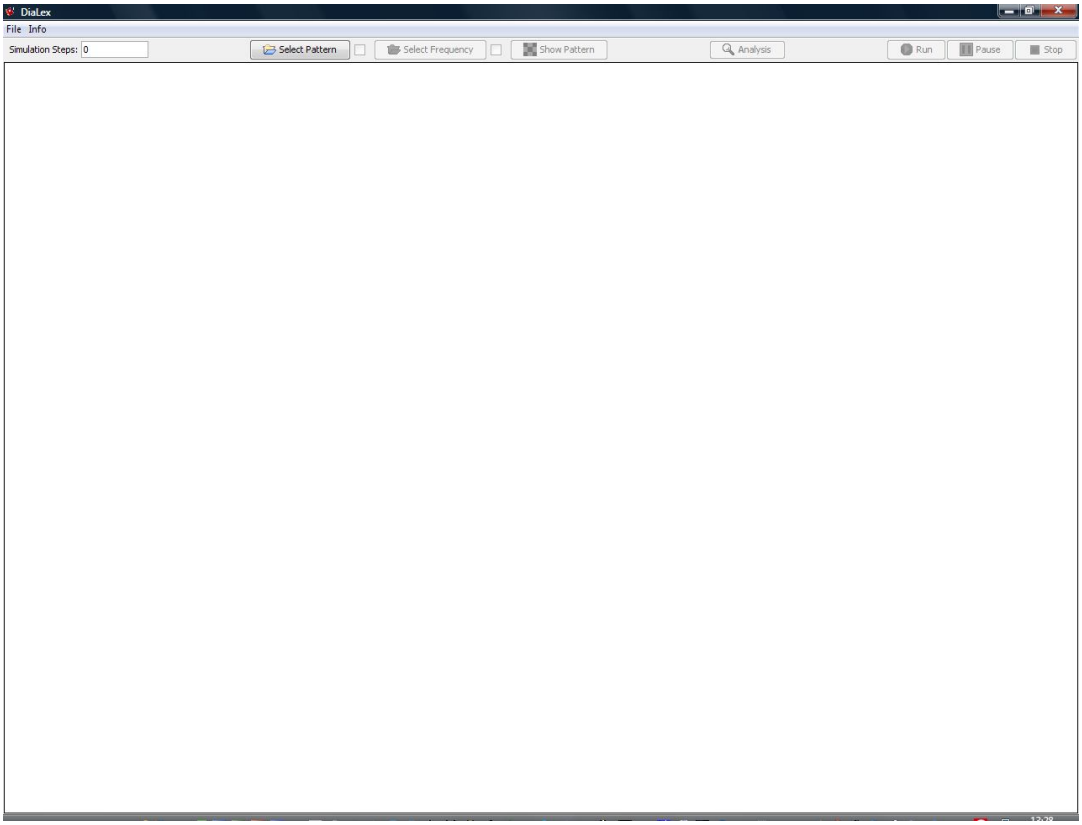


Figure 152: Main Window

DiaLex's main window is divided into six main components which are enumerated in figure 153 and explained in the following.

1. The first component which is placed in the upper left corner is the menu bar, consisting of a *file menu* and an *info menu*. Inside the file menu, you will find the items "New", "Load", "Save as" and "Exit", allowing you to create new, load already existing or save actual simulations and to exit the program. The info menu contains the item "About DiaLex", showing the splash screen that also appears during the program start.

2. The second component is the *simulation steps* text field. Here the number of simulation steps has to be entered by the user.

3. The third component concerns the simulation's *input pattern*. It can be selected after pushing the "Select Pattern" button and examined after clicking the "Show Pattern" button. The checkbox right of the "Select Pattern" button signalizes whether an input pattern is successfully loaded or not. If it is the "Select Frequency" Button will appear activated. Similarly to the input pattern the according frequencies can be added to the simulation and the checkbox right of the "Select Frequency" button indicates if they were loaded successfully. The frequencies will also be shown in the "Show Pattern" dialog. (c.f. figure 167)

4. The fourth component is the "Analysis" button that opens the *analysis window*. It is only active after having run a simulation, while it is in pause or after a simulation was loaded.

5. The fifth component controls the simulation run. Here the buttons "Start", "Pause" and "Stop" can be found, starting, pausing or continuing and stopping simulations.

6. The sixth component which will be empty when starting DiaLex is the environments panel, listing all simulated environments including their attributes and events.

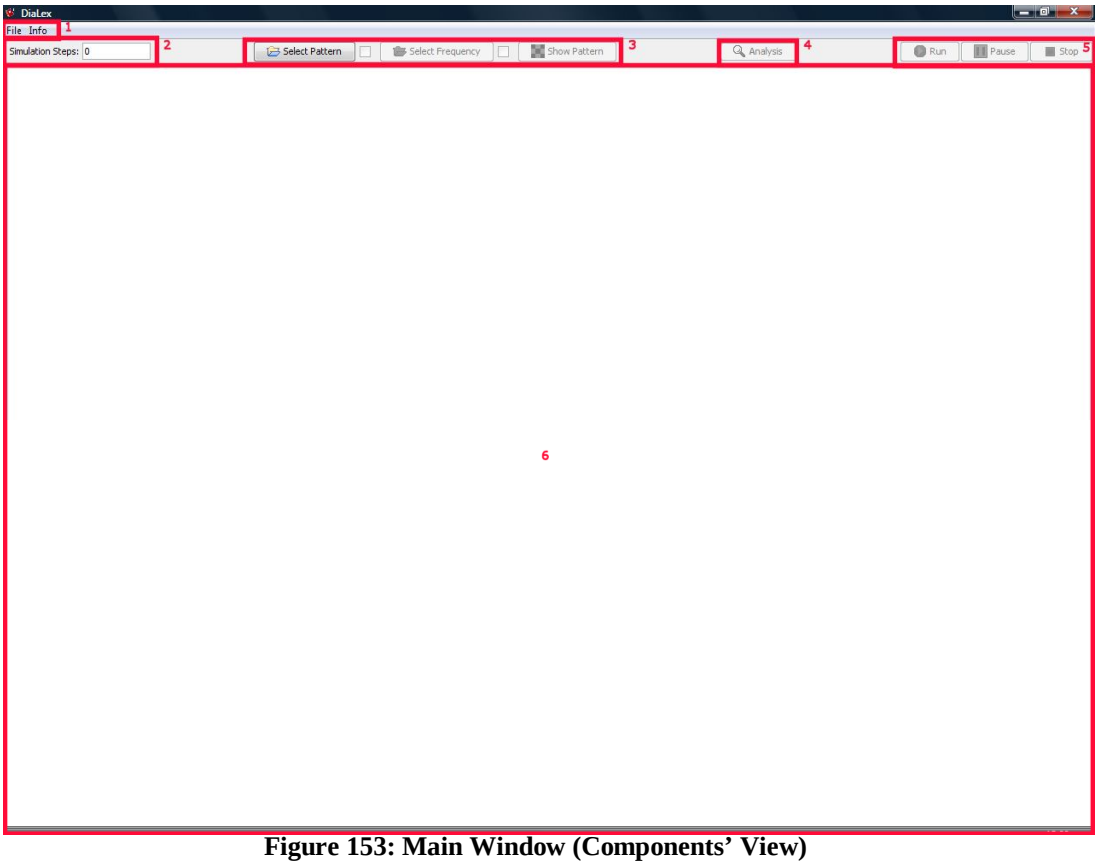


Figure 153: Main Window (Components' View)

C.1 How to create a new Simulation

New simulations can be built using the *new simulation dialog* which can be found in the *file menu*. (cf. figure 154)

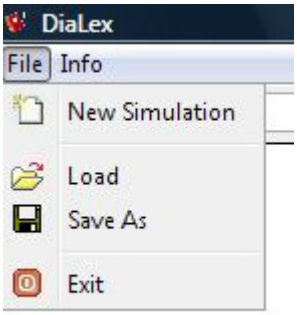


Figure 154: File Menu (Detail)

Within the appearing dialog screen which is shown in figure 155 the number of simulation environments and standard *attributes* and *events* for all environments have to be entered.

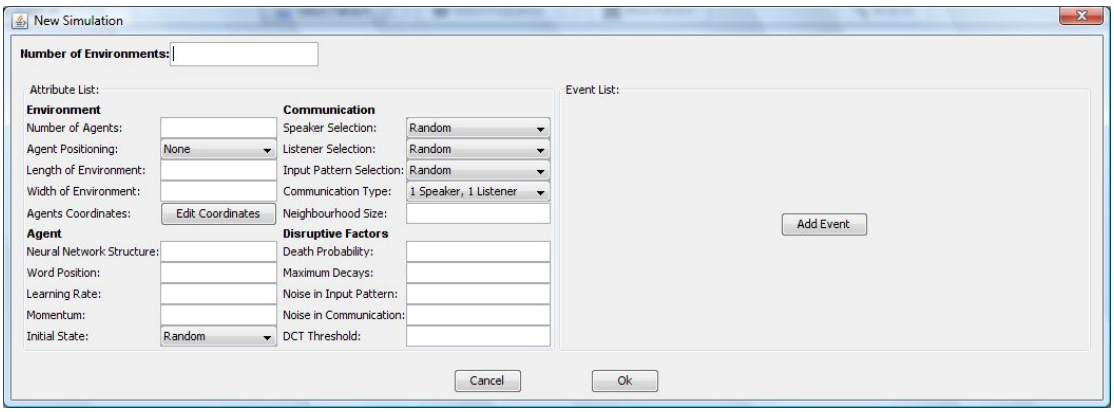


Figure 155: New Simulation Dialog at Beginning

The attributes are divided into four main groups which will be explained in the following:

Environment:

- **Number of Agents**
  This integer number determines the number of agents in each simulation environment.

- **Agent Positioning**
  This drop down menu is used to choose the way how agents are positioned in the environment. The options are “None”, “Random” and “Coordinates”. So agents are either not positioned at all, randomly within the extensions of the environment or explicitly by coordinates which can be entered in the *Edit Coordinates Dialog.*

- **Length of Environment**
  Here the length of the environment’s first dimension can be declared.

- **Width of Environment**
  This integer value stands for the second dimension of each environment’s extent.

- **Agents’ Coordinates**
  Pushing this button opens the Edit Coordinates dialog which is shown in figure 156. Each agent’s coordinates can be changed here.

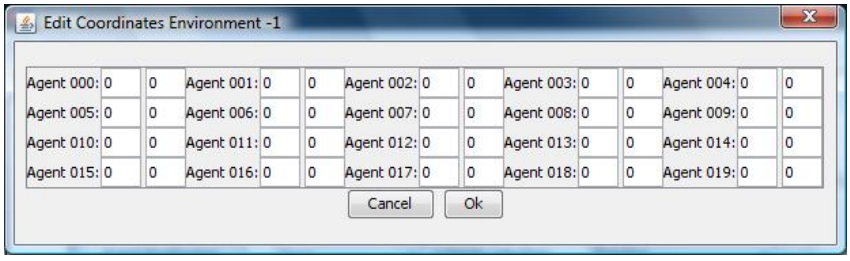


Figure 156: Edit Coordinates Dialog

Agent:

- **Neural Network Structure**
  The network structure is declared in the following way: Layers are always two-dimensional and are separated by “,”. The two dimensions of each layer are separated by the character “x”. No separator is necessary behind the last layer, as it is shown for one example in figure 161.

- **Word Position**
  The word position indicates the word representation layer. Layer numbers start from 0 which stands for the input layer.

- **Learning Rate**
  This real number in the interval [0;1] declares the neural network's learning rate.

- **Momentum**
  The neural network's momentum is determined by this real number in the interval [0;1].

- **Initial State**
  In this drop down menu it can be selected whether the neural network's connections of each agent are initialized with random real values in the interval [-0.5; 0.5] or if they are all set to 0.0. Accordingly the options in this drop down menu are "Random" and "All 0".

Communication:

- **Speaker Selection**
  The way to choose the speaker agent for a communication attempt is determined in this drop down menu. The two choices are "Random" and "In Order". Hence, agents are either chosen randomly or in order, from the first to the last agent, repeated permanently in a loop.

- **Listener Selection**
  For the listener selection there are three different choices: "Random", "Euclidean Distance" and "Manhattan Distance". Accordingly, the listener agent is either selected randomly, or by probabilities determined by either its Euclidean or Manhattan distance from the speaker's position.

- **Input Pattern Selection**
  The input pattern for a communication can either be chosen randomly or randomly based on its frequency. The choices in this drop down menu are "Random" and "Frequency".

- **Communication Type**
  This drop down menu determines wether there is one speaker and one listener in a communication or if both agents act as speaker and listener at the same time. As only the listener optimizes its neural network to match the speaker's verbal representation, either one or both agents have to optimize their networks after a communication.

- **Neighbourhood Size**
  This integer value declares the neighbourhood size. All agents that are positioned within the herein declared distance to the speaker are treated as direct neighbours with a distance of 1.0 to the speaker during listener selection.

Disruptive Factors:

- **Death Probability**
  The real value entered here determines the probability with what every agent "dies" in every simulation step. The value must be in the interval [0;1]. A dead agent will be replaced by a newly created one. The new agent's neural network is determined by the "Initial State" drop down menu.

- **Maximum Decays**
  This set of real numbers in the interval [0;1] is declared similarly to the agent's neural network. A maximum decay has to be determined for every connection between two layers in the neural network. The real numbers are separated by ",". For a three-layer network like "2x2,1x2,2x2" a possible valid set of decays could be "0.1,0.05".

- **Noise in Input Pattern**
  The noise in input patterns is configured through this real number in the interval [0;1]. It indicates the probability with what a value in the input pattern is replaced by a random value in the interval [0;1].

- **Noise in Communication**
  The noise in a communication is configured in a similar way. The only difference is that a value in a verbal representation of the speaker is replaced by a random value.

- **DCT Threshold**
  This real number in the interval [0;1] declares the threshold used in the quantisation of the word pattern after it has been transformed by the discrete cosine transformation. Quantified values lower than this threshold will be eliminated during this calculation.

Note: It is not necessary to determine all the listed attributes. But it will not be possible to complete the New Simulation Dialog without having declared the following attributes: "Number of Environments", "Number of Agents", "Neural Network Structure", "Word Position", "Learning Rate" and "Momentum". All the other settings can either be kept in its standard selection or empty.

In the right half of the window the standard events for all environments can be declared by pushing the "Add Event" button. A new dialog shown in figure 157 will appear. Events are divided into an *if-part* and a *then-part*, declaring the *constraint* and the *effect*. Possible constraints are concerning either the simulation step or the calculated avg1, avg2 or understanding values for the whole agent community.

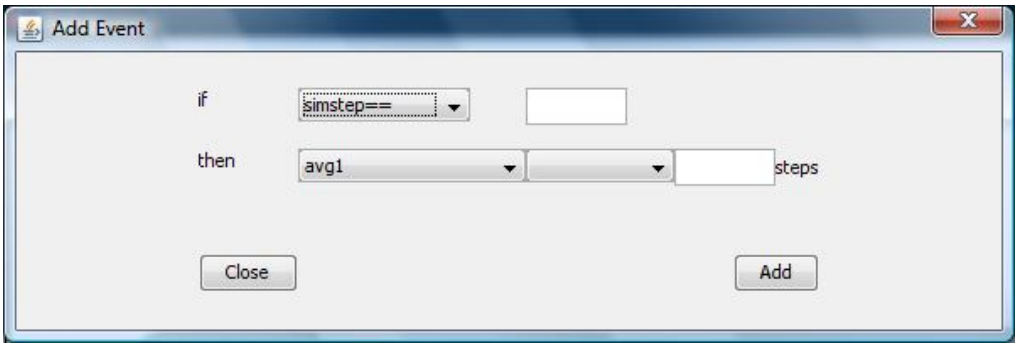


Figure 157: Add Event Dialog at Beginning

In DiaLex there are two different kinds of effects: Effects concerning the simulation run itself and effects that collect simulation data. The first type of effect is easier to define, an example is shown in figure 158. In this example, the listener selection will be changed to Manhattan distance based agent selection if the calculated value for understanding is bigger than 0.9.

Figure 158: Add Event Dialog Example 1

In the second case it might be essential to collect data continuously. That is why the possibility to collect data after every certain amount of steps was introduced. Figure 159 shows an example calculating the avg1 value every 1000 steps starting from step 0. The calculated data will be stored in a database and be held there for analysis after a simulation run.

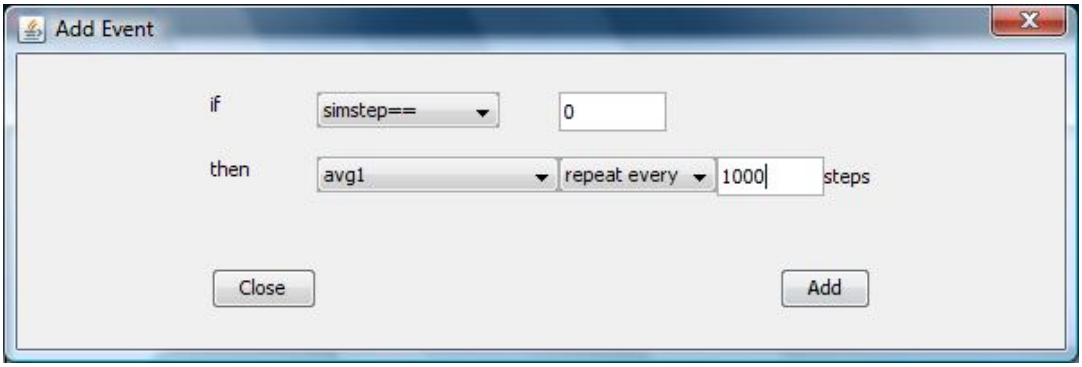


Figure 159: Add Event Dialog Example 2

A special case calculating the indicators *avg1*, *avg2* and *understanding* is the possibility to calculate them only for a subset of input patterns in case of avg1 and more important, for a subset of agents in case of avg2 and understanding. For that case effects with the ending "_" can be selected. In the now appearing field the numbers of the selected patterns or agents can be entered. Figure 160 shows an example for the continuous calculation of the indicator understanding for the agents 0, 1, 2, 3 and 4.

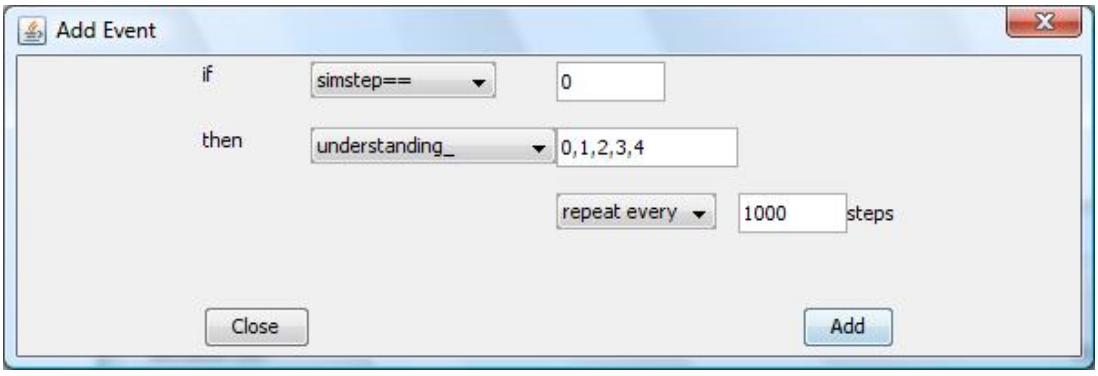


Figure 160: Add Event Dialog Example 3

At this point the New Simulation Wizard is completed. Figure 161 shows an example of how the newly created simulations may look like.

The values declared here will be the standard values in all environments, but can be changed afterwards in the Environment Panel. Confirming the dialog, the main window looks similar to the example in figure 162. Now, the attributes and events in each environment can be changed independently from all the other environments.

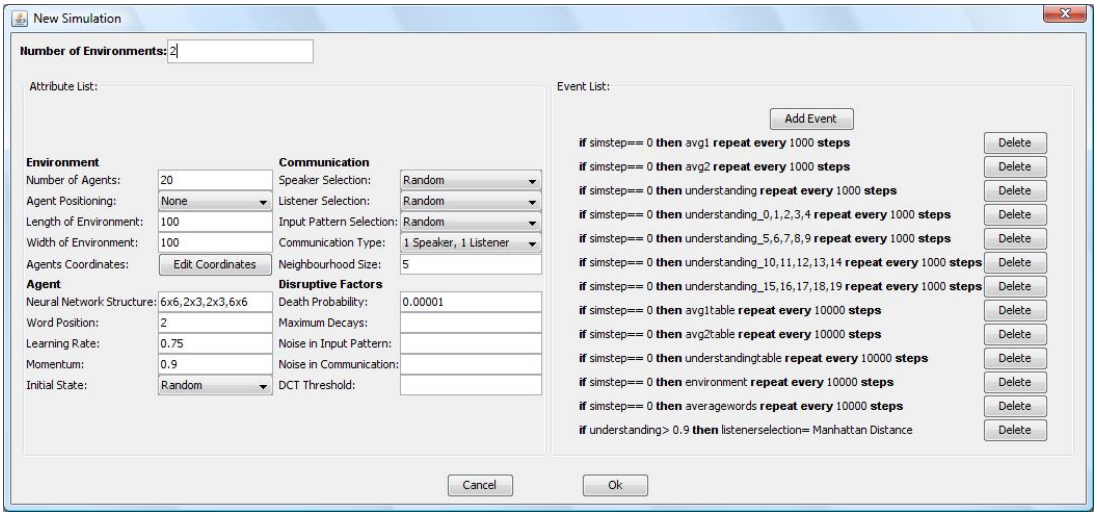


Figure 161: New Simulation Dialog completed

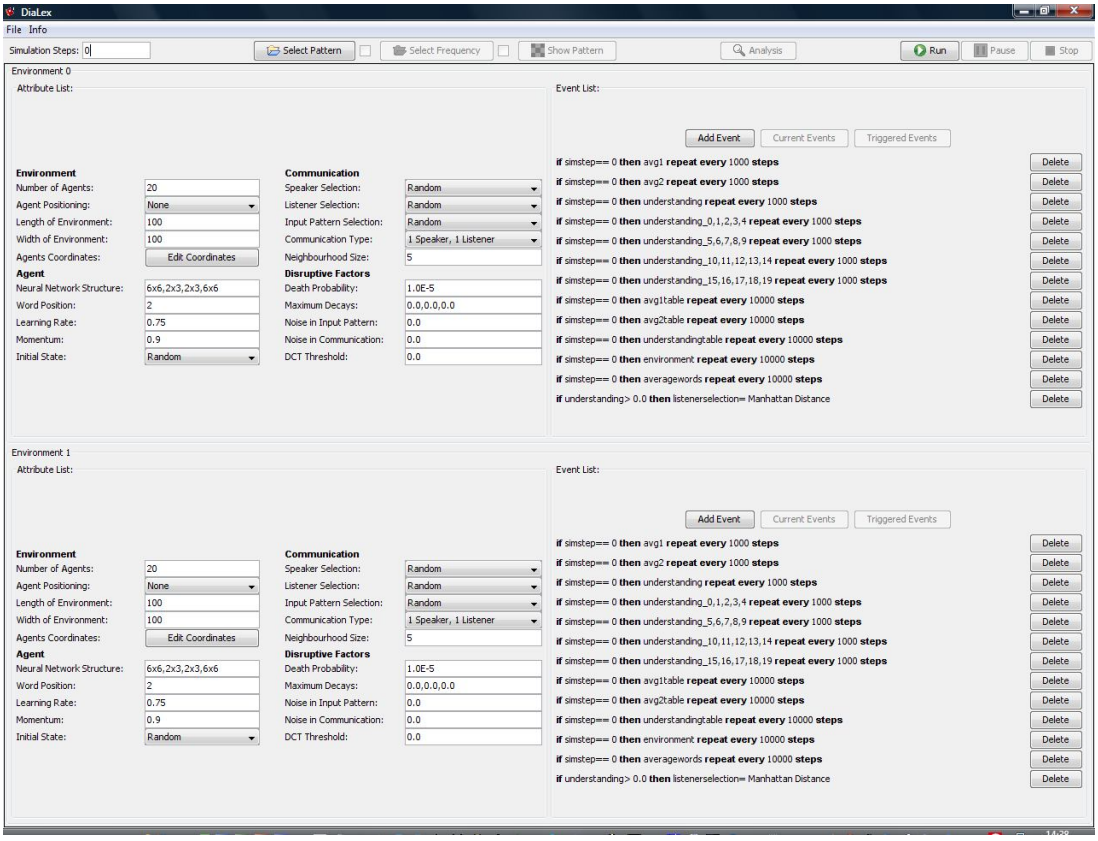


Figure 162: Main Window with Environments

The second necessary step to make a simulation run is the definition of a sequence of input patterns. Therefore a CSV-file can be selected after pressing the “Select Pattern” button. The structure of a CSV-file must be as follows: (cf. figure 163)

- Patterns are separated by a blank line, before the first and after the last pattern there must not be a blank line.
- One pattern is a two dimensional array of double values, separated by “;” in a line, representing the horizontal dimension (no “;” at the end of a line) and by a new line in the vertical dimension.

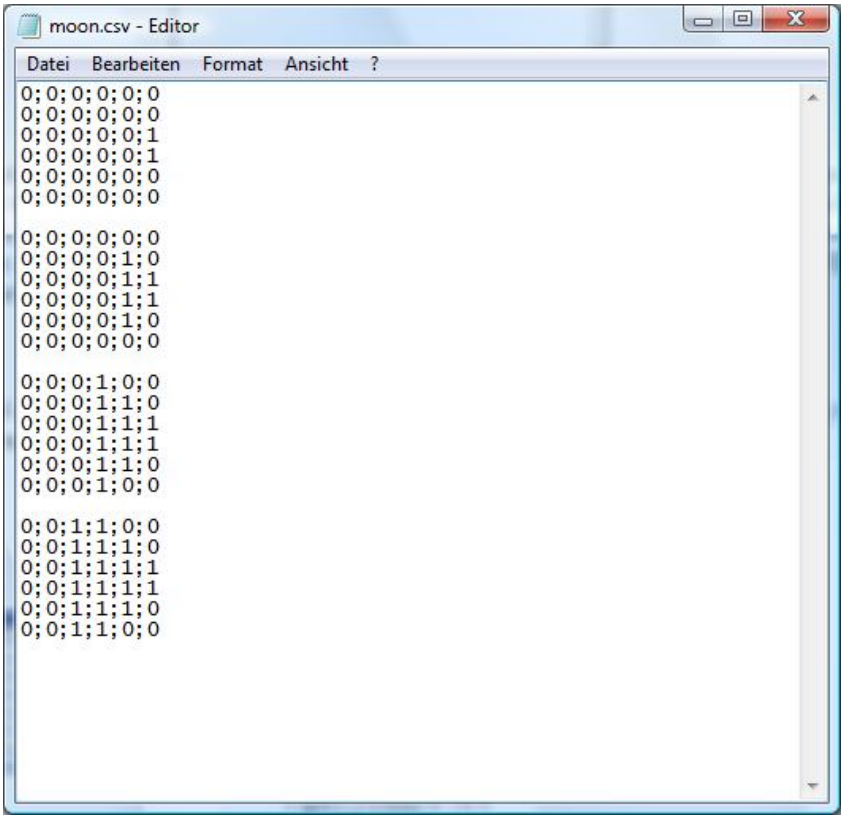


Figure 163: A CSV File for the Input Pattern

Having selected a CSV File, the checkbox right of the button becomes activated (cf. figure 164) and both the “Select Frequency” and the “Show Pattern” button are activated.

Figure 164: Input Pattern Selected (Detail)

Now it is possible (Note: This step is not necessary!) to select pattern frequencies for the set of input patterns. Therefore a second CSV File can be selected after pressing the “Select Frequency” button. This file’s structure is easy:

- Frequencies are separated by a new line, before the first and after the last pattern there must not be a blank line.

Figure 165 shows an example of a frequency CSV File. A frequency can be any real number.

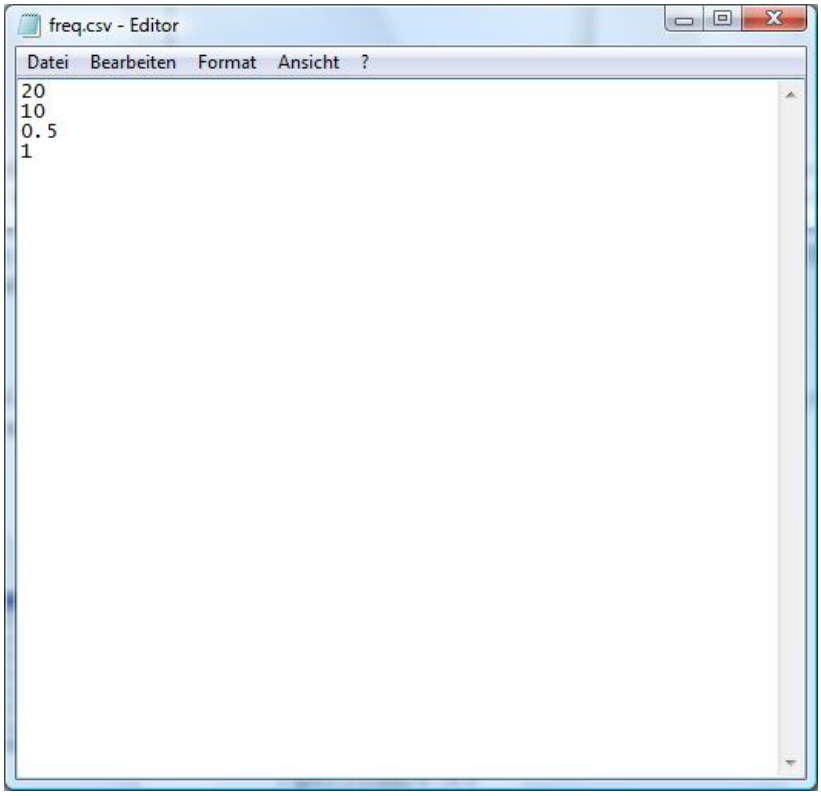


Figure 165: A CSV File for the Input Pattern Frequency

As soon as a pattern frequency was selected the checkbox right of this button also becomes activated, indicating that the frequency file was read successfully and that the number of frequencies fits the sequence of input patterns. (cf. figure 166)

Figure 166: Frequency Selected (Detail)

Now the sequence of input patterns can be examined after clicking the “Show Pattern” button. Then a new window pops up showing the input pattern. (cf. figure 167)

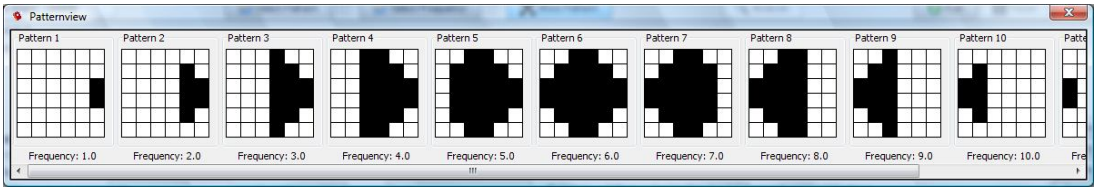


Figure 167: Input Pattern View

Finally, only the number of simulation steps has to be entered in the simulation steps field. Now the simulation is completely created and a simulation run can be started.

C.2 Running a Simulation

Simulations can be started by pressing the “Start” button. Then the agents’ networks will be created and all the simulation environments will be simulated simultaneously. While running a simulation, a progress monitor like shown in figure 168 will appear in the middle of the screen and will be updated every 100 simulation steps. Simulations can be paused pushing the “Pause” button once, and continued pressing it a second time. To manually stop a simulation, push the “Stop” button.

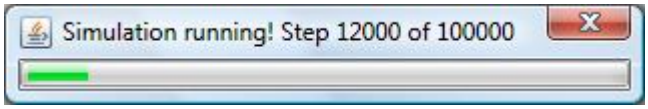


Figure 168: Progress Monitor

Being in the pause state or after a simulation run has finished it’s possible to open the analysis window by clicking the “Analysis” button.

C.3 Simulation Analysis

In the analysis window all the collected simulation data will appear in form of a list like it is shown in figure 169.

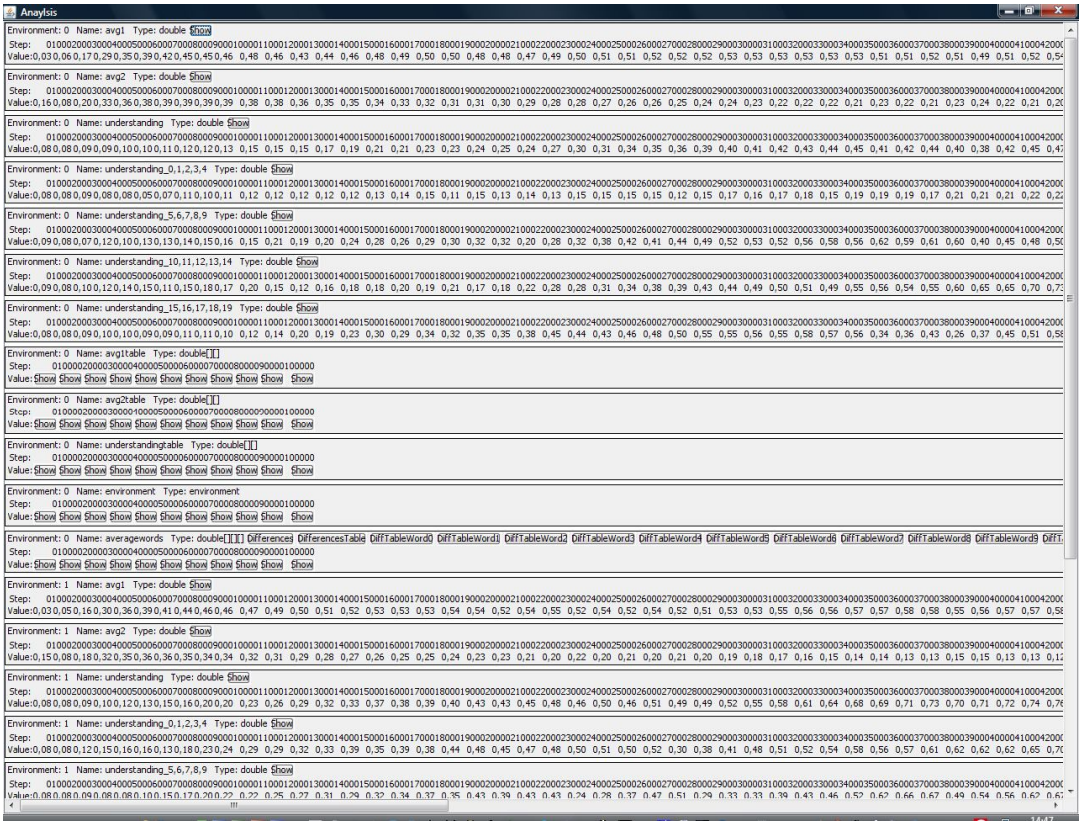


Figure 169: Analysis Window

Simulation data held in a list of real values like avg1, avg2 and understanding are directly shown in lists in the analysis window, combining the calculated value with its appropriate simulation step. After having selected the “Show” button they are also obversable in a chart window as it is shown in figure 170. All available data rows can be added to one chart window by selecting them in the “add” combo box. Removing them is possible by selecting them in the “remove” combo box. The names of each data row are built by first their belonging environment, abreviated with an E and the name of the effect. The x-axis always shows the simulation steps while the y-axis represents the calculated values that are always inside the interval [0;1].

Data held in cross tabulations like avg1table, avg2table and understandingtable can be looked at after selecting the “Show” button below the appropriate simulation step within the list. An example is given in figure 171 showing the understandingtable of simulation step 2000000 of environment 0. To make group recognition easier for the human eye the values are coloured allocating the values from 0 to 1 to the colour spectrum. In case of an avg1table the patterns build the cross tabulation’s axis, if it is an avg2table or an understandingtable the axis are built by the environment’s agents, always showing the similarity between either

both patterns or both agents verbal representations. Cross tabulations can also be exported to a CSV file by pushing the “e” button on the keyboard.

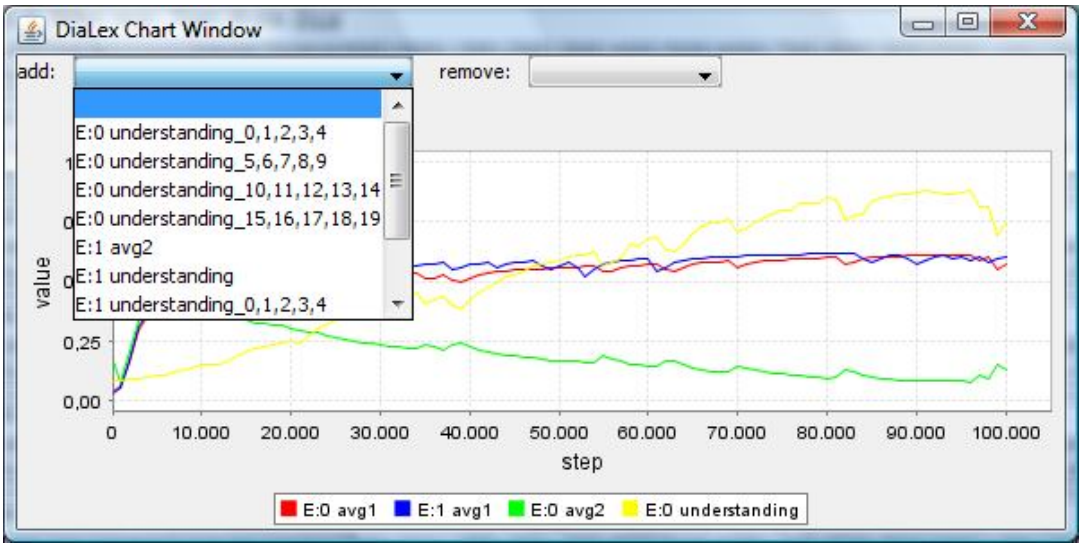
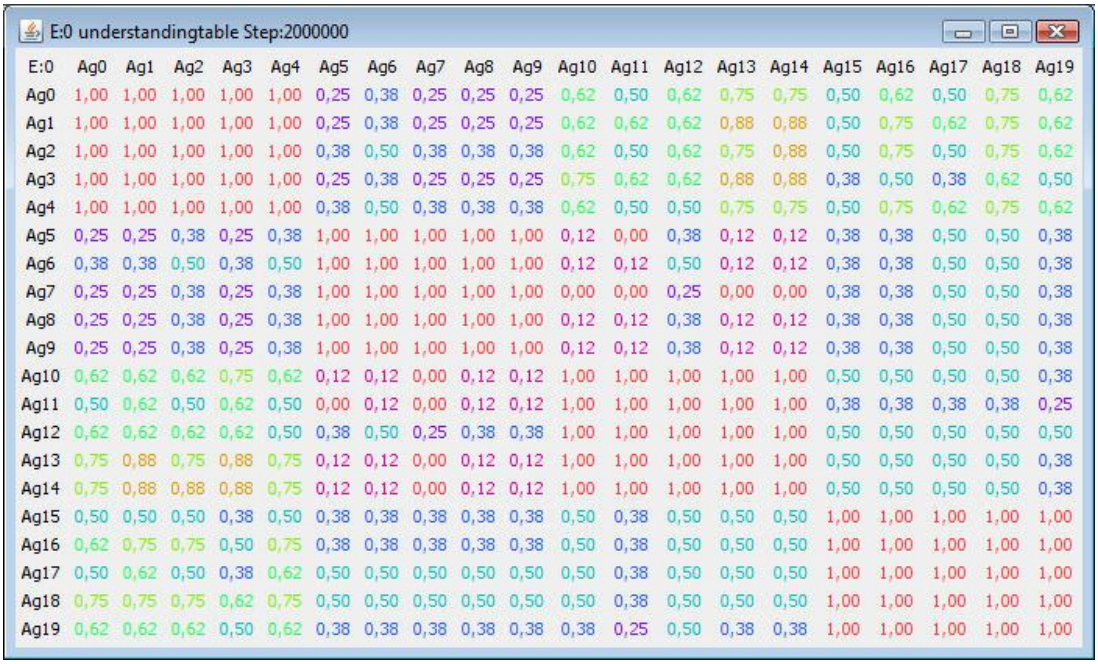


Figure 170: Chart Window



E:0	Ag0	Ag1	Ag2	Ag3	Ag4	Ag5	Ag6	Ag7	Ag8	Ag9	Ag10	Ag11	Ag12	Ag13	Ag14	Ag15	Ag16	Ag17	Ag18	Ag19
Ag0	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,62	0,50	0,62	0,75	0,75	0,50	0,62	0,50	0,75	0,62
Ag1	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,62	0,62	0,62	0,88	0,88	0,50	0,75	0,62	0,75	0,62
Ag2	1,00	1,00	1,00	1,00	1,00	0,38	0,50	0,38	0,38	0,38	0,62	0,50	0,62	0,75	0,88	0,50	0,75	0,50	0,75	0,62
Ag3	1,00	1,00	1,00	1,00	1,00	0,25	0,38	0,25	0,25	0,25	0,75	0,62	0,62	0,88	0,88	0,38	0,50	0,38	0,62	0,50
Ag4	1,00	1,00	1,00	1,00	1,00	0,38	0,50	0,38	0,38	0,38	0,62	0,50	0,50	0,75	0,75	0,50	0,75	0,62	0,75	0,62
Ag5	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,00	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag6	0,38	0,38	0,50	0,38	0,50	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,50	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag7	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,25	0,00	0,00	0,38	0,38	0,50	0,50	0,38
Ag8	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag9	0,25	0,25	0,38	0,25	0,38	1,00	1,00	1,00	1,00	1,00	0,12	0,12	0,38	0,12	0,12	0,38	0,38	0,50	0,50	0,38
Ag10	0,62	0,62	0,62	0,75	0,62	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag11	0,50	0,62	0,50	0,62	0,50	0,00	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,38	0,38	0,38	0,38	0,25
Ag12	0,62	0,62	0,62	0,62	0,50	0,38	0,50	0,25	0,38	0,38	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,50
Ag13	0,75	0,88	0,75	0,88	0,75	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag14	0,75	0,88	0,88	0,88	0,75	0,12	0,12	0,00	0,12	0,12	1,00	1,00	1,00	1,00	1,00	0,50	0,50	0,50	0,50	0,38
Ag15	0,50	0,50	0,50	0,38	0,50	0,38	0,38	0,38	0,38	0,38	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag16	0,62	0,75	0,75	0,50	0,75	0,38	0,38	0,38	0,38	0,38	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag17	0,50	0,62	0,50	0,38	0,62	0,50	0,50	0,50	0,50	0,50	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag18	0,75	0,75	0,75	0,62	0,75	0,50	0,50	0,50	0,50	0,50	0,50	0,38	0,50	0,50	0,50	1,00	1,00	1,00	1,00	1,00
Ag19	0,62	0,62	0,62	0,50	0,62	0,38	0,38	0,38	0,38	0,38	0,38	0,25	0,50	0,38	0,38	1,00	1,00	1,00	1,00	1,00

Figure 171: Cross Tabulation

Complete snapshots of simulation environments are kept as a deep copy of the simulation environment at the defined simulation step. Those snapshots can be

looked at after selecting a ”Show” button below the corresponding simulation step. The now appearing environment analysis window gives you detailed information about the *activation values* of the *word layer* (left side) and the *output layer* (right side).

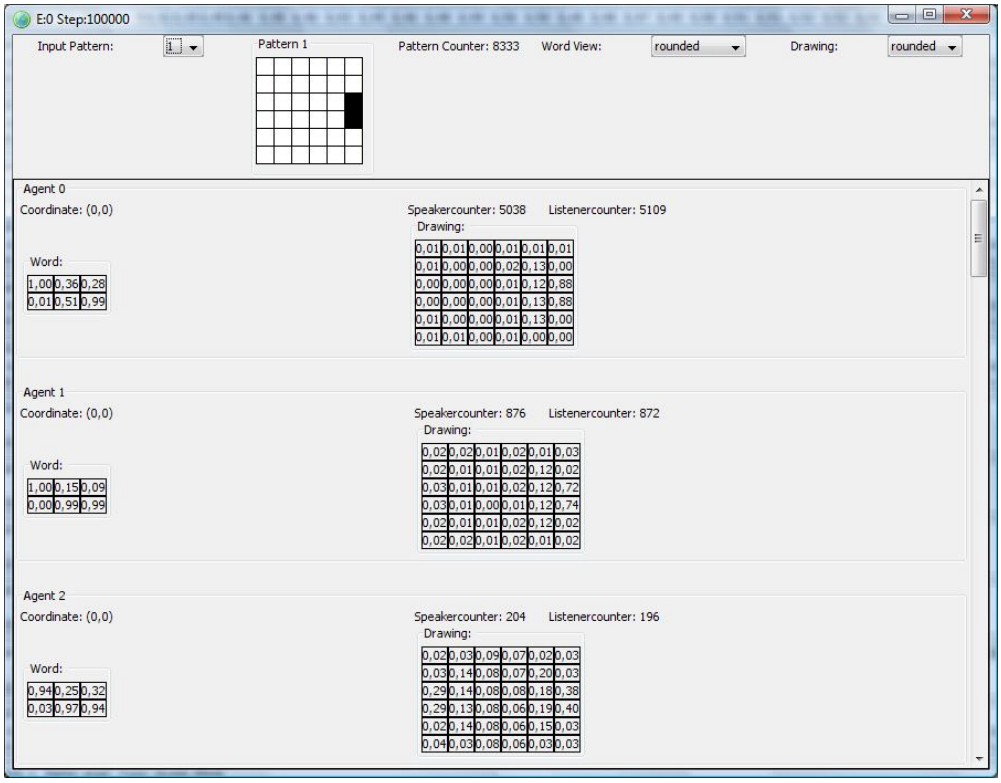


Figure 172: Environment Analysis Window (rounded)

Three combo boxes are placed in the upper part of the window. The left one lets you choose the input pattern which will be shown in the top middle of the window. Right of this combo box it is indicated how often this input pattern has been used in a communication. The second combo box allows you to define whether to see the word layer’s activation values, related to the chosen input pattern, in a *standard view*, showing the real values that are calculated inside the word layer, a *rounded view*, rounding those values to two decimal places, a *literal view*, mapping each activation value to a syllable in the way it is shown in figure 170, as a *graphical view*, mapping the agents’ verbal representation to a graphic pattern or as a *dct graphical view*, mapping the agents’ words to a graphic pattern after it was transformed by the discrete cosine transform. The third combo defines the way the output layer will be represented. Here *standard*, *rounded* and *graphical views* can be chosen to illustrate how each agent redraws the visual scene it was shown.

In the lower part you will find an agents list showing both layers’ activation values, represented in the chosen way. Figure 172 gives you a full-screen example

of the rounded representation while figure 173 shows both, a word layer and an output layer in the graphical mode. Figure 174 gives an example for both standard views.

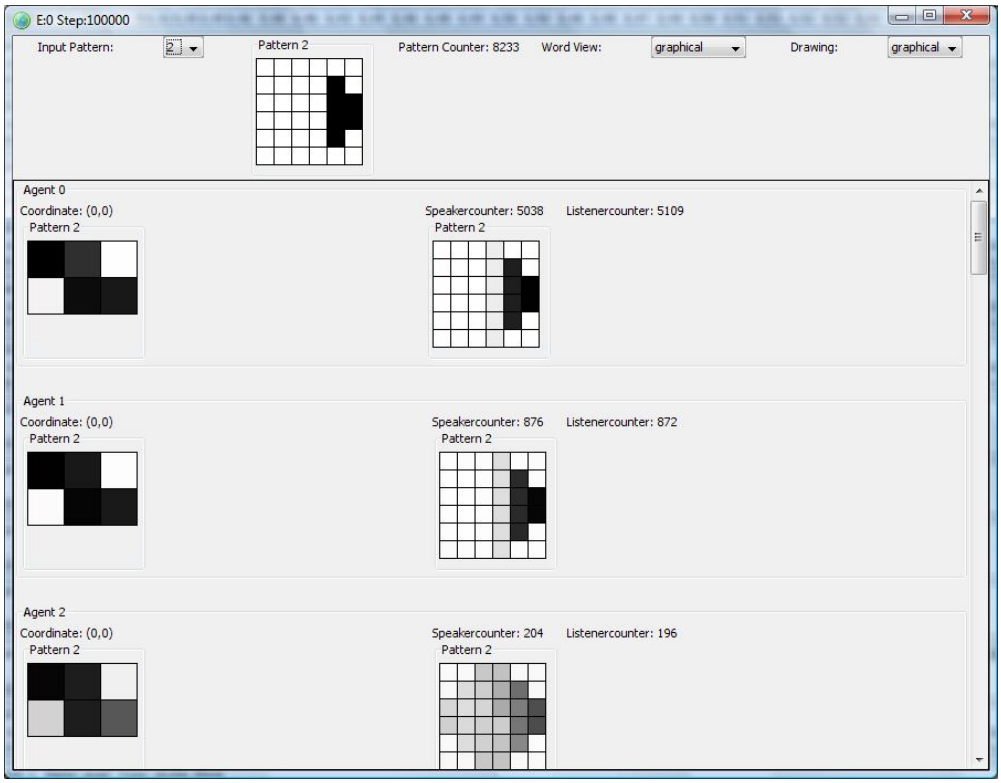


Figure 173: Environment Analysis Window (graphical)

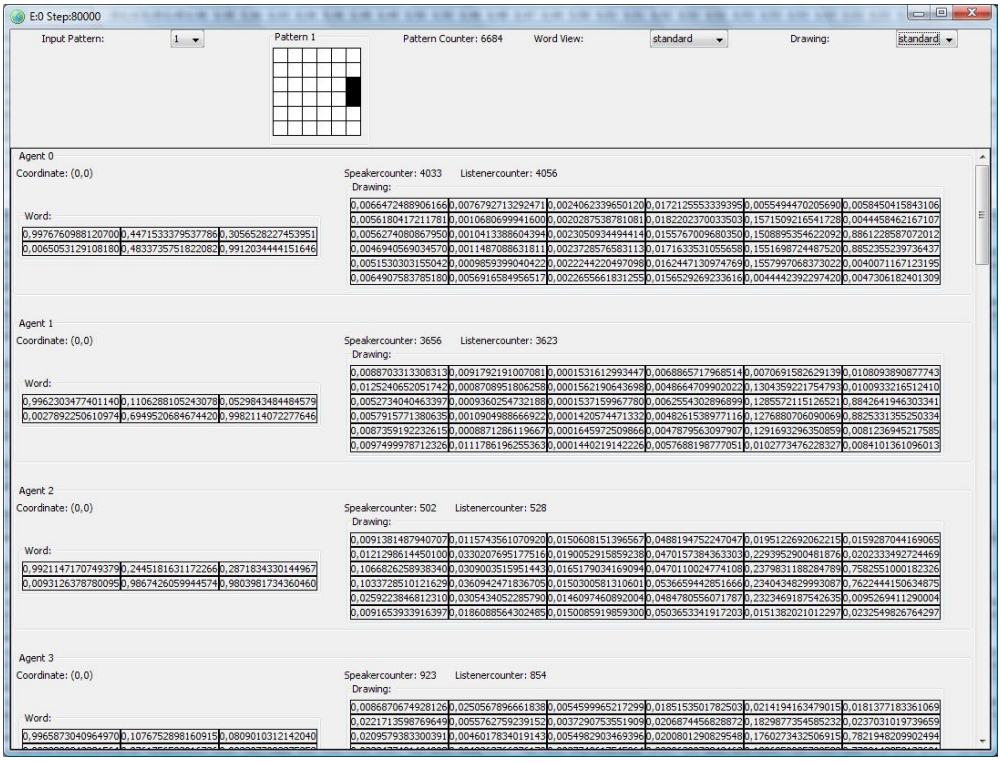


Figure 174: Environment Analysis Window (standard)

Figure 175 shows an example for a word pattern that was transformed via the *discrete cosine transform.*

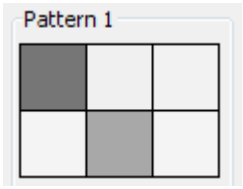


Figure 175: Word DCT Graphical View (Detail)

Figure 176 shows the same word in the literal view, using the *syllable matching mechanism* shown in the following figure.

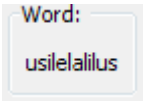


Figure 176: Word Literal View (Detail)

```
public static String getSyllable (double in)
{
String out = new String("");

        if (in < 0.1)  out += "al"; else
        if (in < 0.2)  out += "as"; else
        if (in < 0.3)  out += "es"; else
        if (in < 0.4)  out += "el"; else
        if (in < 0.5)  out += "il"; else
        if (in < 0.6)  out += "is"; else
        if (in < 0.7)  out += "os"; else
        if (in < 0.8)  out += "ol"; else
        if (in < 0.9)  out += "ul"; else
        if (in < 1.0)  out += "us";

        return out;
}
```

Figure 177: Syllable Matching Mechanism

Data collected using the *average words* function in an event calculates a simulation environment's lexicon. The words are formed by the average activation values of all agents. Three different ways to analyse the collected data are given. Figure 178 shows an environment's average words lexicon at one simulation step. This window is accessible via the "Show" button beneath its corresponding simulation step.

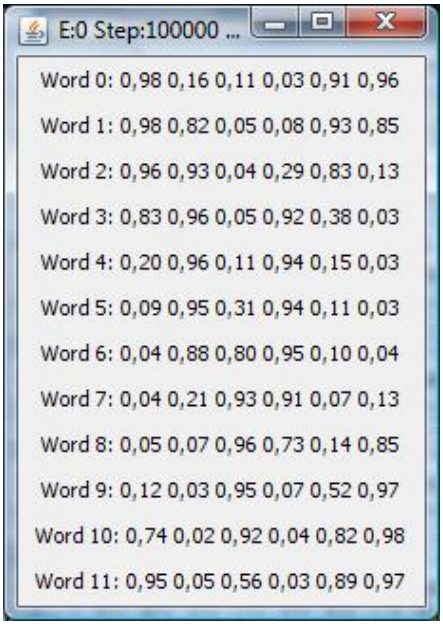


Figure 178: Average Words

Figure 179 illustrates each word's differences between two calculated lexicons. It is accessible via the "Differences" button. As it is relatively complicated to find permanent changes in a community this way, a third view was introduced.

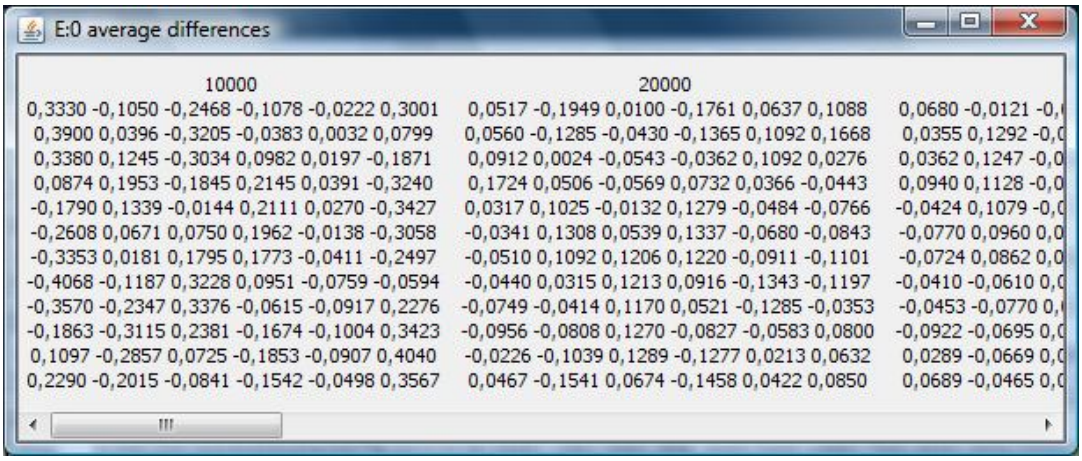
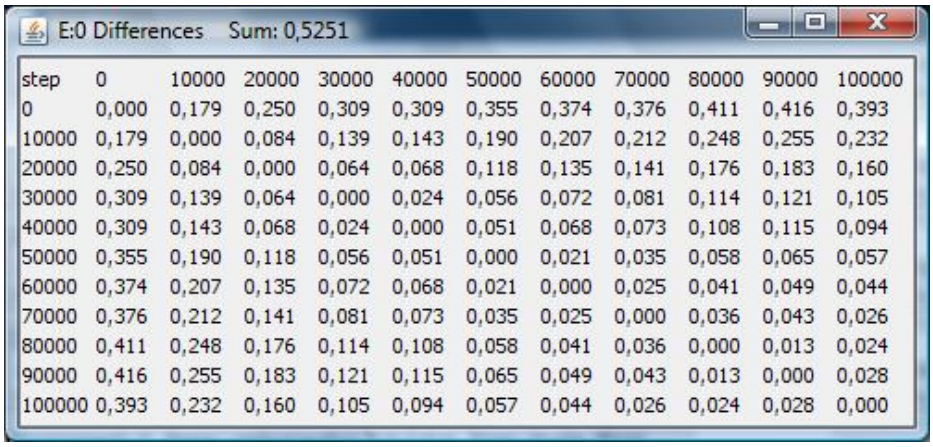


Figure 179: Word Differences

This view in form of a cross tabulation shows the summed up absolute values of all differences for all activation values of all words and agents between all the collected lexicons. An example is given in figure 180. The summed up differences from each step to another are shown in the title of the window. It can be calculated by summing up all the differences either above or beneath the diagonal in the cross tabulation. This view is available both for the differences of all words together or for each single word either activating the "Differences Table" or the "DiffTableWordX" button, whereas "X" has to be replaced by the specific word.



step	0	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
0	0,000	0,179	0,250	0,309	0,309	0,355	0,374	0,376	0,411	0,416	0,393
10000	0,179	0,000	0,084	0,139	0,143	0,190	0,207	0,212	0,248	0,255	0,232
20000	0,250	0,084	0,000	0,064	0,068	0,118	0,135	0,141	0,176	0,183	0,160
30000	0,309	0,139	0,064	0,000	0,024	0,056	0,072	0,081	0,114	0,121	0,105
40000	0,309	0,143	0,068	0,024	0,000	0,051	0,068	0,073	0,108	0,115	0,094
50000	0,355	0,190	0,118	0,056	0,051	0,000	0,021	0,035	0,058	0,065	0,057
60000	0,374	0,207	0,135	0,072	0,068	0,021	0,000	0,025	0,041	0,049	0,044
70000	0,376	0,212	0,141	0,081	0,073	0,035	0,025	0,000	0,036	0,043	0,026
80000	0,411	0,248	0,176	0,114	0,108	0,058	0,041	0,036	0,000	0,013	0,024
90000	0,416	0,255	0,183	0,121	0,115	0,065	0,049	0,043	0,013	0,000	0,028
100000	0,393	0,232	0,160	0,105	0,094	0,057	0,044	0,026	0,024	0,028	0,000

Figure 180: Word Differences Cross Tabulation

C.4 Saving and Loading a Simulation

A simulation can be saved after clicking the “Save As” item in the file menu. In that case all the created environments with all contained attributes, events and all the belonging data in the database will be stored in the state they are at the moment of saving.

Two steps are necessary, choosing the directory where to save the file and entering a valid name for the *.sim* file. Simulations are saved as binary files, so editing saved simulations is nearly impossible beyond the use of DiaLex.

For loading a *.sim* file just click the “Load” item in the file menu. Choose a directory and a file and confirm your choice.

C.5 Further Information

For further information or questions on DiaLex feel free to contact me:

Christian Klein anaconda@uni-koblenz.de

Literaturverzeichnis

[AS10] Abbasi, R.; Staab, S.: *RichVSM: enRiched Vector Space Models for Folksonomies*. Information Systems and Semantic Web Research Group, Universität Koblenz, 2010.

[Bal01] Balzert, Helmut: *Lehrbuch der Software-Technik*. 2. Auflage, Spektrum Akademischer Verlag, 2001.

[Bic90] Bickerton, D.: *Language and Species*, University of Chicago Press, Chicago, 1990.

[Bop16] Bopp, F.: *Über das Conjugationssystem der Sanskritsprache in Vergleichung mit jenem der griechischen, lateinischen, persischen und germanischen Sprache*. Andreaischen, 1816.
URL: http://books.google.de/books?id=N3UHAAAAQAAJ [12.08.2009]

[Bre21] Bredsdorff, J.H.: *Über die Ursachen der Sprachveränderung*, 1821. Deutsche Ausgabe: Narr Dr. Gunter Verlag, 2. Auflage, 1984.

[Brü71] Brückner, W.: *Frankfurter Wörterbuch*. Kramer, Frankfurt, 1971.

[BT42] Bloch, B.; Trager, G.L.: *Outline of Linguistic Analysis*. Linguistic Society of America, Baltimore, 1942.

[Cas94] Caspers, P.: *Op Kölsch jesaat*, Greven Verlag, Köln, 1994.

[Cho57] Chomsky, N.: *Syntactic Structures*. Mouton, Den Haag, 1957.

[Cho65] Chomsky, N.: *Aspects of the Theory of Syntax*. MIT Press, Cambridge, 1965.

[Cho72] Chomsky, N.: *Language and Mind*. Harcourt Brace Jovanovich, New York, 1972.

[CHVB04] Crockford, C.; Herbinger, I.; Vigilant, L.; Boesch, C.: *Wild Chimpanzes Produce Group-Specific Calls: a Case for Vocal Learning?*, In: *Ethology*, Blackwell, Berlin, 2004.

[Cur77] Curtiss, S.: *Genie: A Linguistic Study of a Modern-Day 'Wild Child'*, Academic Press, 1977.

[deB97] de Boer, B.: *Generating Vowel Systems in a Population of Agents*. In: Husband, P.; Harvey, I.: *Fourth European Conference on Artificial Life*, Cambridge, MIT Press, 1997.

[deB99] de Boer, B.; Vogt, P.: *Emergence of Speech Sounds in Changing Populations*. In: Floreano,D; Nicoud, J.-D.; *Proceedings of the Fifth European Conference on Artificial Life*, Springer, Berlin, 1999.

[deB02] de Boer, B.: *Evolving Sound Systems*. In: Cangelosi, A.; Parisi, D.: Simulating the Evolution of Language. Springer, London, 2002.

[DS06] Dürr, M.; Schoblinski, P.: *Deskriptive Linguistik: Grundlagen und Methoden*, Vandenhoeck & Ruprecht, Göttingen, 3. Auflage, 2006.

[EB96] Ebert, C., Dumke, R.: *Software- Metriken in der Praxis: Einführung und Anwendung von Software- Metriken in der industriellen Praxis*, Springer, 1996.

[FDS00] Ford, J.K.B.; Deecke, V.B.; Spong, P.: *Dialect change in resident killer whales: implications for vocal learning and cultural transmission*. In: *Animal Behaviour*, Elsevier, 2000.

[FK07] Fuchs, D.; Klein, C.: *LexLearn – Emergenz eines gemeinsam genutzten Lexikons*. Diplomarbeit, Fachbereich IV, Universität Koblenz-Landau, 2007.
URL: http://en.scientificcommons.org/23792361 Stand: 28.10.2008

[Gal04] Galantucci, B.: *An Experimental Study of the Emergence of Human Communication Systems*. In: *Cognitive Science 29*, S. 737-767, Austin, 2004.

[Gra06] Grassegger, H.: *Phonetik Phonologie*. Schulz-Kirchner Verlag, Idstein, 2006.

[Gre63] Greenberg, J. H.: *Universals of Language*. MIT Press, Cambridge, 1963.

[Gri54] Grimm, J; Grimm, W.: *Deutsches Wörterbuch*. v. Hirzel Verlag, Leipzig, 1854.

[Hau96] Hauser, M.: *The Evolution of Communications*, MIT Press, Cambridge 1996.

[Hal68] Hall, R.A.: *An Essay on Language*. Chilton Books, New York, 1968.

[HH95] Hutchins, Edwin; Hazlehurst, Brian: *How to Invent a Lexicon: The Development of Shared Symbols in Interaction.* In: Gilbert, Nigel; Conte, Rosaria: *Artificial Societies: The Computer simulation of social life*. London, UCL Press, 1995.

[HS98] Huhns, M., Singh, M. P.: *Readings in Agents*, Morgan Kaufmann, 1998.

[Hur89] Hurford, J.: *Biological Evolution of the Saussurean sign as a component of the language acquisition device*. In: *Lingua* 77(2): S. 187-222, Elsevier, Amsterdam, 1989.

[IPA05] International Phonetic Association: *The International Phonetic Alphabet*, Link: http://www.arts.gla.ac.uk/IPA/IPA_chart_(C)2005.pdf, 2005.

[Jac99] Jackendoff, R.: *Possible Stages in the Evolution of the Language Capacity*. In: *Trends in Cognitive Science*. Vol.3.7, Seiten 272-279, London, 1999.

[Jon86] Jones, W.: *The Sanscrit Language*. 1786 In: Teignmouth, L.: *The Works of Sir William Jones.* London, 1807.

[Kel90] Keller, R.: *Sprachwandel: Von der unsichtbaren Hand in der Sprache*, Francke Verlag, Tübingen, 1990.

[KH02] Kirby, S.; Hurford, J.:*The Emergence of Linguistic Structure: An Overview of the Iterated Learning Model*. In: Cangelosi, A.; Parisi, D.: *Simulating the Evolution of Language*. Springer, London, 2002.

[Kon65] Konishi, M: *The role of auditory feedback in the control of vocalization in the white-crowned sparrow*, In: *Zeitschrift für Tierpsychologie*, Blackwell, Berlin, 1965.

[KP97] Klein, W.; Perdue, C.: *The Basic Variety, or: Couldn't Language be much simpler?* Second Language Resulotion 13, Seiten 301-347, 1997.

[Kru00] Krüger, Guido: *GoTo Java 2, Handbuch der Java-Programmierung*. 2. Auflage, München, Addison-Wesley, 2000.

[Lew09] Lewis, M. P.: *Ethnologue: Languages of the World*. 16. Auflage, SIL International, Dallas, 2009.
URL: http://www.ethnologue.com Stand: 12.08.2009

[LF98] Livingstone, D.; Fyfe, C.: *A Computational Model of Language-Physiology Coevolution*. In: Computing and Information Systems, Vol. 5, No. 2, P. 55-62. University of Paisley, Paisley, 1998.

[Lip05][29] Lippe, Wolfram-Manfred: *Soft-Computing: Mit neuronalen Netzen, mit Fuzzy-logic und evolutionären Algorithmen*. Springer, 2005.

[Liv02] Livingstone, D.: *The Evolution of Dialect Diversity*. In: Cangelosi, A.; Parisi, D.: *Simulating the Evolution of Language*. Springer, London, 2002.

[29]Teile daraus sind unter http://cs.uni-muenster.de/Professoren/Lippe/lehre/skripte/wwwnnscript als interaktives Skript erhältlich.

[Lüd80] Lüdtke, H.: *Sprachwandel als universales Phänomen*. In: Lüdtke, H.: *Kommunikationstheoretische Grundlagen des Sprachwandels*. De Gruyter, Berlin, 1980.

[Lyo81] Lyons, J.: *Language and Linguistics*. Cambridge University Press, 1981. Deutsche Übersetzung: *Die Sprache*, C.H. Beck'sche Verlagsbuchhandlung, München 1983.

[Mar70] Marler, P.: *A comparative approach to vocal learning: song development in white-crowned sparrows*. In: *Journal of Comparative and Physiological Psychology*, Cambridge Scholars Publishing, 1970.

[McM99] McMahon, A.M.S.: *Understanding Language Chance*. Cambridge University Press, 1999.

[Mei07] Meibauer, J.: *Einführung in die germanistische Linguistik*. J. B. Metzler, Stuttgart, 2007.

[MHGMB92] Mitani, J.; Hasegawa, T.; Gros-Luis, J.; Marler, P.; Byrne, R.: *Dialects in Wild Chimpanzes?*, In: *American Journal of Primatology*, Wiley-Liss, 1992.

[MT62] Marler, P.; Tamura, M.: *Song „Dialects" in three Populations of White-Crowned Sparrows*, In: *The Condor*, 1962.

[MWA99] Marshall, A.; Wrangham, R.; Arcadi, A.: *Does learning affect the structure of vocalizations in chimpanzes?* In: *Animal Behaviour*, Elsevier, 1999.

[Neu03] Neubauer, N.: *Emergence in a Multiagent Simulation of Communicative Behaviour*. In: *PICS, Publication Series of the Institute of Cognitive Science*, Volume 11, University of Osnabrück, 2004.

[NHBF87] Nevo, E.; Heth, G.; Beiles, A.; Frankenberg, E.: *Geographic dialects in blind mole rats: Role of vocal communication in active speciation*. In: *Proceedings of the National Academy of Sciences*, 1987.

[Oli97] Oliphant, M.: *Formal Approaches to Innate and Learned Communication: Laying the Foundation of Language*. Dissertation, University of California, San Diego, 1997.

[PAM07] Pagel, M.; Atkinson,Q.; Meade, A.: *Frequency of word-use predicts rates of lexical evolution throughout Indo-European history*. In: Nature, 11. Okt. 2007, Nature Publishing Group, London, 2007.

[PS02] Perla, B.; Slobodchikoff, C.N.: *Habitat structure and alarm call dialects in Gunnison's prairie dog*. In: *International Society for Bevahioral Ecology*, 2002.

[Red88] Rédei, K.: *Uralisches Etymologisches Wörterbuch. Band I: Uralische und finnisch-ugrische Schicht. Band II: Finnisch-permische und finnisch-wolgaische Schicht. Band III: Register.* Harrassowitz, Wiesbaden, 1988 (Band I und II) und 1991 (Band III).

[Rob79] Robins, R.H.: General Linguistics: An Introductory Survey. Longman, London, 1979.

[Rum86] Rumelhart, David E., Hinton Geoffrey G., Williams R. J.: *Learning Internal Representation by Error Propagation*. In: Rumelhart, David E., McClelland, James L.: *Parallel Distributed Processing*, Volume 1: *Foundations*, Seiten 318-362. London, MIT Press, 1986.

[Rum94] Rumelhart, David E.; Widrow, Bernhard; Lehr Michael A.: *The Basis Ideas in Neural Networks*. In *Communication of the ACM*, Volume 37, Issue 3, Seiten 87-92. New York, ACM Press, 1994.

[Rus90] Russel, Beale; Jackson, Tom: *Neural Computing: An Introduction*. Bristol, CRC Press, 1990.

[SAE98] Slobodchikoff, C.N.; Ackers, S.H.; van Ert, M.: *Geographic Variation in Alarm Calls of Gunnison's Prairie Dogs*, In: *Journal of Mammology*, 1998.

[Sal13] von Parma, Salimbene: *Chroniken.* In: G. G. Coulton: *St. Francis to Dante*. London: David Nutt, 1906, S. 242f.
URL: http://www.fordham.edu/halsall/source/salimbene1.html, Stand 09.08.2007

[Sap21] Sapir, E.: *Language: An Introduction to the Study of Speech*. New York: Harcourt, Brace & World, 1949.

[Sil91] Silbernagl, S.: *Taschenatlas der Physiologie*. Thieme Verlag, 4. Auflage, Stuttgart, 1991.

[Smi02] Smith, K.: *The cultural evolution of communication in a population of neural networks.* In: *Connection Science Vol.14*, S.65-84, Taylor & Francis, London, 2002.

[Ste05] Steels, L.: *The Emergence and Evolution of Linguistic Structure: From Lexical to Grammatical Communication Systems.* In: *Connection Science*, Volume 17, Issue 3 & 4, Seiten 213 – 230, Taylor & Francis, London, 2005.

[TG05] Gilbert, Nigel; Troitzsch, Klaus G.: *Simulation for the Social Scientist.* Buckingham, Philadelphia, Open University Press, 2005.

[Tho01] Thompson, Richard F.: *The Brain*. Deutsche Übersetzung: *Das Gehirn: Von der Nervenzelle zur Verhaltenssteuerung*. Spektrum, Akad. Verl., Heidelberg, 2001.

[TS94] Tubaro, P.; Segura, T.: *Dialect Differences in the Song of Zonotrichia Capensis in the Southern Pampas: A Test of the Acoustic Adaptation Hypothesis*. In: *Cooper Ornithological Society*, 1994.

[Wen75] Wenker, G.: *Deutscher Sprachatlas.* 1875. Aktuelle Ausgabe: *Digitaler Wenker-Atlas*. Forschungszentrum Deutscher Sprachatlas, Marburg, 2009.
URL: http://www.diwa.info/ [12.08.2009]

[WK96] West, M.; King, A.: *Social learning, synergy and songbirds*. In: Heyes, C.; Galef, B.: *Social Learning in Animals*, Academic Press, San Diego, 1996.

[WW97] Weilgart, L.; Whitehead, H.: *Group-specific dialects and geographical variation in coda repertoire in South Pacific sperm whales*. In: *Behavioral Ecology and Sociobiology*, Vol. 40, Springer, 1997.

Curriculum Vitae

Persönliche Daten

Name:	Christian Klein
Anschrift:	Am Hochkreuz 24 56729 Monreal
Telefon	+49-2651-73912
Mobil:	+49-170-9025010
E-Mail:	christian@christianklein.de
Matrikelnummer:	201210280
Geburtsdatum:	27.05.1981
Geburtsort:	Mayen
Familienstand:	ledig
Staatsangehörigkeit:	deutsch

Ausbildung

10/2007 - heute **Universität Koblenz-Landau**
Promotionsstudiengang Informatik
angestrebter Abschluss: Dr.rer.nat.

10/2001 - 09/2007 **Universität Koblenz-Landau**
Diplomstudiengang Informatik
Abschluss: Diplom-Informatiker, Note: 1,4

07/2000 - 05/2001 **Zivildienst**
Caritas Sozialstation, Mayen

07/1991 - 06/2000 **Megina-Gymnasium, Mayen**
Abschluss: Allgemeine Hochschulreife, Note: 2,6

Wissenschaftliche Vorarbeiten

04/2007 - 09/2007 Universität Koblenz-Landau
Diplomarbeit „LexLearn - Emergenz eines gemeinsam genutzten Lexikons“
Implementierung des Modells von Hutchins und Hazlehurst zur Entstehung eines gemeinsamen Lexikons in einem in Java geschriebenen Simulationstool; beliebig viele parallel arbeitende Simulationsumgebungen und frei konfigurierbare neuronale Netze der Simulationsagenten ermöglichen nicht nur eine Verifizierbarkeit der bereits bekannten Ergebnisse, sondern auch eine Fülle neuer Simulationen; Multi-Agenten-Simulation.

09/2005 - 06/2006 Universität Koblenz-Landau
Studienarbeit „CoMicS“
Erweiterung und Verbesserung der Mikrosimulationstools MicSim, insbesondere durch die Einbettung einer Datenbank, die Einführung eines Schichtensystems und verbesserte Auswertungsmöglichkeiten.

07/2004 - 06/2005 Universität Koblenz-Landau
Projektpraktikum „MicSim“
Konzeptionierung und Implementierung eines Tools zur Erstellung von Mikrosimulationen.

Universitäre Vorträge

09/2009 ADAPT-Oberseminar, Koblenz
Präsentation der Ergebnisse der Dissertation
„DiaLex – Die Entstehung von Dialekten“

11/2008 Doktorandenworkshop Universität Koblenz-Landau, Koblenz
Präsentation der Zwischenergebnisse der Dissertation
„DiaLex – Die Entstehung von Dialekten“

07/2007 EMIL-Meeting, Koblenz
Präsentation der Ergebnisse der Diplomarbeit
„LexLearn – Emergenz eines gemeinsam genutzten Lexikons“

04/2006 Advanced Simulation Workshop, Koblenz
Präsentation der Ergebnisse der Studienarbeit
„CoMicS”