TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Mensch-Maschine-Kommunikation

Graphische Modelle im natürlichsprachlichen Mensch-Maschine-Dialog

Stefan Schwärzler

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. sc.techn. Andreas Herkersdorf
Prüfer der Dissertation:	1. apl. Prof. Dr.-Ing., Dr.-Ing. habil. Günther Ruske
	2. Univ.-Prof. Dr. rer. nat., Dr. rer. nat. habil. Anne Brüggemann-Klein

Die Dissertation wurde am 16.06.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 22.12.2009 angenommen.

Vorwort

Die vorliegende Arbeit ist das Ergebnis meiner Forschungstätigkeit als wissenschaftlicher Assistent am Lehrstuhl für Mensch-Maschine-Kommunikation der Technischen Universität München. Dem Leiter dieses Lehrstuhls, Herrn Prof. Dr.-Ing. habil. Gerhard Rigoll danke ich ganz besonders für das von ihm geförderte interdisziplinäre Arbeitsumfeld. Sehr gerne werde ich mich an die interessanten Projekte im Bereich des Exzellenzclusters für kognitiv-technische Systeme erinnern.

Mein außerordentlicher Dank gilt meinem Doktorvater Herrn Prof. Dr.-Ing. habil. Günther Ruske für die Inspiration und Möglichkeit zur Durchführung dieser Arbeit. Ein sehr angenehmes Arbeitsklima, wertvolle Anregungen und Diskussionen und genügend wissenschaftlicher Freiraum zur Verwirklichung meiner eigenen Ideen haben den Erfolg dieser Arbeit ermöglicht.

Ebenso gilt mein herzlicher Dank Frau Prof. Dr. rer. nat. habil. Anne Brüggemann-Klein von der Informatik XI für die Erstellung des Zweitgutachtens. Einige wichtige Grundlagen dieser Arbeit erlernte ich in den Vorlesungen ihres Fachgebietes.

Weiterhin möchte ich mich bei all meinen Kollegen bedanken. Ein spezieller Dank gilt meinem langjährigen Kollegen Herrn Dr. Joachim Schenk für die nahezu täglichen Diskussionen und gemeinsamen Forschungsabende. Bei Herrn Dr. Frank Wallhoff bedanke ich mich für die Förderung im Bereich der multi-modalen Dialogmanagementsysteme. Ebenso sind Alexander Bannat, Jürgen Blume, Jürgen Geiger, Benedikt Hörnler und Moritz Kaiser aufgrund der vielen Diskussionen nicht zu vergessen. An Frau Gertrud Günther, Herrn Peter Brand und Herrn Heiner Hundhammer ein großes Lob für die technische Unterstützung.

Abschließend und für mich am wichtigsten möchte ich den Dank meiner Familie ausdrücken: Allen voran möchte ich mich bei meiner Freundin Frau Dr. Konstanze Jähne für ihre Liebe, Geduld und permanenten Beistand bedanken. Für Motivation und manch hervorragender Verköstigung danke ich meiner Tante Frau Beate und meinem Onkel Herrn Michael Vogl herzlich. Ganz besonders danke ich meinen Eltern Frau Ulrike und Herrn Franz Schwärzler sowie meiner Großmutter Frau Elfriede Giselbrecht, die mich immer unterstützt und mir stets Rückhalt gegeben haben.

München, im Januar 2010 Stefan Schwärzler

Kurzfassung

In dieser Arbeit werden die Verarbeitungsstufen des natürlichsprachlichen Mensch-Maschine-Dialogs am Beispiel eines automatischen Flugauskunftssystems einheitlich mit Graphischen Modellen (GM) modelliert. Die Erkennung von bedeutungstragenden Wortphrasen aus dem Spracherkenner wird mit einem zweistufigen Modell realisiert, welches stochastische und regelbasierte Verfahren kombiniert. Um aus diesen Wortphrasen Strategien für einen natürlichen Dialogablauf zu entwickeln, wird ein diskretes Hidden-Markov-Modell verwendet. Mit der Berechnung der Dialogstrategie zur Laufzeit kann sich das System schnell auf veränderte Situationen einstellen und bleibt flexibel einsetzbar.

Die entwickelten GM werden theoretisch analysiert, Trainingsalgorithmen werden abgeleitet und Experimente für die jeweilige Verarbeitungsstufe durchgeführt. Abschließend wird eine Realisierungsmöglichkeit mithilfe von Agenten und deren Erweiterbarkeit aufgezeigt.

Inhaltsverzeichnis

Abbildungsverzeichnis

1

Einleitung

Die gesprochene Sprache ist die natürlichste und direkteste Form der zwischenmenschlichen Kommunikation. Der Mensch erwirbt seine sprachlichen Fähigkeiten in einem längeren Prozess von Kindheit an Die gesprochene Sprache markiert ein wesentliches Attribut der menschlichen Intelligenz und der Prozess des Sprachverstehens trägt substanziell zur Reifung des menschlichen Intellektes bei. Daher ist es wenig verwunderlich, dass die Vorstellung von „Künstlicher Intelligenz“ in „Science-Fiction-Genre“ (Literatur und Filmen) häufig mit der natürlichen Kommunikation zwischen Mensch und Maschine dargestellt wird. Jedem sind wahrscheinlich Szenen aus den Filmen „Per Anhalter durch die Galaxis“ [Ada79] bzw. „A Space Odyssey“ [KC68] bekannt, bei denen intelligente Maschinen, Roboter oder sprechende Autos mit dem Menschen kommunizieren und diesem bei der Lösung von komplexen Problemen unterstützen.

Die Vision des sprechenden, denkenden und handelnden Roboters kann durch die ständige Weiterentwicklung der Computertechnologie des sogenannten „Informationszeitalters“ in Erfüllung gehen. Denkbare Anwendungen sind die natürliche Kommunikation mit einem „elektronischen Organizer“, mit vielfältigen Funktionen im Fahrzeug oder mit einem kognitiven multi-modalen Serviceroboter.

Im Vergleich zu anderen Eingabemodalitäten (z. B. die Gestenerkennung im Serviceroboter) hat die gesprochene Sprache den Vorteil, dass intuitive Aussagen mit universellen und allgemein bekannten Wörtern ausgedrückt werden können. Ein weiterer Vorteil ergibt sich dadurch, dass die natürliche Sprache keine komplexe Bedienung von Eingabegeräten erfordert, weshalb diesem Kommunikationskanal im Fahrzeug eine besondere Bedeutung zukommt.

Der Nachteil der maschinellen Spracherkenner ist, dass nur ein Teil im Audiosignal des später geschriebenen Textes enthalten ist. Ebenso ist eine weitere semantische Verarbeitung nur aufgrund des bisherigen Wissens möglich. Der Mensch kann durch seine vielfältige Wahrnehmung auf Basis des Satzzusammenhangs und mithilfe von Plausibilitätsannahmen und Hintergrundwissen wesentlich leichter unklar gesprochene oder homophone Wörter erkennen als die Maschine.

In dieser Arbeit wird eine integrierte Verarbeitung beginnend vom Sprachsignal über das Sprachverstehen bis zum Dialogmanagement über mehrere Konzeptstufen vorgestellt. Nach linguistischen Grundsätzen kann das Sprachverstehen in verschiedene Konzeptstufen unterteilt werden [All95]:

Die **akustische Stufe** beschreibt die Verbindung zwischen dem akustischen Sprachsignal und dessen symbolische Repräsentation in der Lautsprache (bzw. Wörter, Silben, Teilsilben).

Die **morphologische Stufe** definiert die Anordnung der Wörter aus der orthographischen Stufe (z. B. durch Beugung und Verbundwörter).

Die **syntaktische Stufe** konstruiert, basierend auf grammatikalischen Regeln, gültige Sätze, welche bestimmten grammatikalischen Regeln folgen.

Die **semantische Stufe** versucht, die Bedeutung der Wörter bzw. der Satzphrasen zu erkennen, unabhängig vom Kontext vorangehender Teilsätze/Sätze.

Die **pragmatische Stufe** entscheidet mit dem Wissen, welches nicht explizit in einem Teilsatz/Satz ausgesprochen wurde, aber für eine erfolgreiche Interpretation des Satzes notwendig ist (Kontext-Wissen, Welt-Wissen).

Die **Diskurs Stufe** entscheidet, wie der vorausgehende Satz den nachfolgenden Satz beeinflussen soll.

Diese linguistische Unterteilung liefert einen ersten Anhaltspunkt zur Identifizierung der Teilaufgaben dieser Arbeit. Es zeigt sich jedoch im Detail, dass eine strikte Teilung dieser Aufgaben unmöglich erscheint. Häufig werden in diesem Forschungsbereich die orthographische Transkription und die semantische Decodierung (*semantic slots*) in einem Dialogmanagementsystem als Schnittstellen definiert. Aus dieser Konsequenz erstreckt sich diese Arbeit über drei Forschungsbereiche, nämlich erstens die **Spracherkennung**, zweitens das **Sprachverstehen** und drittens das **Dialogmanagement**.

Das Ziel der Spracherkennung ist es, aus dem Sprachsignal mithilfe der digitalen Signalverarbeitung und der Mustererkennung eine korrekte Wortfolge zu erkennen. Das Eingangssignal ist dabei unvollständig, mit Rauschem behaftet oder aufgrund der unterschiedlichen Aussprachevarianten der Wörter durch verschiedene Sprecher sehr divergent.

Im Gegensatz dazu befasst sich das Sprachverstehen mit einem wohl-definierten und eindeutigen Eingangssignal, nämlich der orthographischen Verschriftung und einem breitgefächerten semantischen Inhalt, welcher sich wiederum nicht eindeutig darstellen lässt.

Das Dialogmanagement zieht aus dieser unsicheren semantischen Darstellung und der Dialoggeschichte einen logischen Schluss und leitet daraus eine eindeutige Fragestellung oder Datenbankabfrage ab. Für all diese Teilbereiche werden

zur vereinfachten Problemdarstellung Modelle verwendet. Im Folgenden soll ein entsprechender Modellbegriff eingeführt werden.

1.1 Modellbildung

Der in der Wissenschaft häufig verwendete Begriff **Modell** wird oft universell gebraucht für abstrakte, konzeptuelle, graphische oder mathematische Modelle. Ein Modell vereinfacht durch Abstraktion die Sicht auf eine reale Welt. Nach Stachowiak [Sta73] ist der Begriff **Modell** durch drei Merkmale gekennzeichnet:

Abbildung: Ein Modell ist immer ein Abbild von etwas, eine Repräsentation natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.

Verkürzung: Ein Modell erfasst nicht alle Attribute des Originals, sondern nur diejenigen, die dem Modellierer bzw. Modellanwender relevant erscheinen.

Pragmatismus: Pragmatismus bedeutet so viel wie Orientierung am Nützlichen. Ein Modell ist einem Original nicht von sich aus zugeordnet. Die Zuordnung wird durch die Fragen „Für wen?“, „Warum?“ und „Wozu?“ relativiert. Ein Modell wird vom Modellierer bzw. Modellanwender innerhalb einer bestimmten Zeitspanne und zu einem bestimmten Zweck für ein Original eingesetzt. Das Modell wird somit interpretiert.

In einem Modell werden bewusst bestimmte, für den Modellierer unbedeutende Merkmale vernachlässigt. Ein Modell zeichnet sich also durch Abstraktion aus, durch die bewusste Vernachlässigung bestimmter Merkmale, um die für den Modellierer oder den Modellierungszweck wesentlichen Modelleigenschaften hervorzuheben. Modelle sind somit stets Abbilder von etwas und Interpretationsgrundlage für jemanden.

Aus diesem Grundsatz werden für jeden der in den folgenden Abschnitten eingeführten Teilbereiche verschiedene Modelle angefertigt.

1.2 Spracherkennung

Die heutigen Spracherkenner erkennen aus dem Sprachsignal meist mithilfe von stochastischen Modellen die gesprochenen Wörter. Die wichtigsten Parameter des Sprachsignals sind:

- Die Energie des Sprachsignals, welches die Lautstärke bestimmt.
- Spektrale Parameter, die die Struktur des Spektrums beschreiben, wodurch vor allem die Eigenschaften des Vokaltrakts erfasst werden.

- Die Grundfrequenz des Sprachsignals gibt eine Auskunft über die Tonhöhe von stimmhaften Sprachlauten. Es wird eine Unterscheidung zwischen stimmhaften und stimmlosen Lauten getroffen.

Jede Einheit wird dabei in einem phonetischen Modell umgesetzt [Rab89]. Dabei ist die lautsprachliche Information wichtig, welche als **Lautschrift** bezeichnet wird. Die Lautschrift aus der akustischen Modellierung wird mit einem Lexikon in die **orthographische** Schrift überführt. Die orthographische Verschriftung des akustischen Modells wird mit einem sogenannten Sprachmodell kombiniert. Dieses reduziert die Möglichkeiten an gesprochenen Wörtern aufgrund einer Wahrscheinlichkeitsverteilung, welche aus der Gesamtheit der vorher gesprochenen Wörtern ermitteln wurde. Ein Spracherkennungssystem führt mit effizienten Algorithmen eine integrierte Suche über alle vorliegenden Wissensquellen durch und decodiert das am besten passende Wort aus dem Sprachsignal. Die strikte Verarbeitungskette (bestehend aus Wortsequenz, einzelnem Wort, phonetischer Einheit und akustischem Muster) erlaubt eine erfolgreiche Anwendung datengetriebener Trainingsalgorithmen. Damit können wesentliche Modellparameter für die Spracherkennung extrahiert werden. Multiple Trainingsdaten aus den aufgezeichneten Sprachsignalen und deren Transkriptionen ermöglichen so eine sprecherunabhängige akustische Modellierung.

1.3 Sprachverstehen

Im Gegensatz zur Spracherkennung kann das Problem des Sprachverstehens, im Folgenden auch als semantische Decodierung bezeichnet, nicht auf eine singuläre Verarbeitungskette reduziert werden. Unter linguistischen Gesichtspunkten beinhaltet natürliches Sprachverstehen den überwiegenden Teil der bereits eingeführten Konzeptstufen (siehe Seite 2: Morphologische, Syntaktische, Semantische, Pragmatische und Diskurs-Stufe). Aus Sicht eines Dialogsystems beinhaltet die semantische Decodierung lediglich die Interpretation applikationsspezifischer Informationen. Üblicherweise erfolgt dies mithilfe von regelbasierten Ansätzen, wie z. B. **TINA** vom Massachusetts Institute of Technology (MIT) oder das **Gemini System** vom Stanford Research Institute (SRI) [Sen92, DG93]. Demgegenüber stehen Anwendungen, welche ähnlich wie im Bereich der Spracherkennung auf datengetriebenen Lernverfahren basieren. Dies sind z. B. das **Chronus System** der American Telephone & Telegraph Corporation (ATT) oder das **HUM System** von Bolt Beranek and Newman Technologies (BBN) [MB95, PT92]. Im Kontext der semantischen Decodierung sind jedoch hierarchisch annotierte und applikationsspezifische Trainingsdaten sehr rar. Maschinelle Lernverfahren benötigen aber einen beachtlichen Teil an hierarchisch annotierten Trainingsdaten, welche im Bereich des Sprachverstehens von Experten mühsam aus dem Sprachsignal transkribiert werden müssen. Ausgehend von dieser Problematik wird in aktuellen Fragen im Bereich der semantischen Decodierung eine Kombination aus regelbasierten Grammatiken und stochastischen Ansätzen

diskutiert [AW04, HY05]. Etwaige Lösungen sind Rahmenmodelle (*Frameworks*), bestehend aus einfachen handgeschriebenen Grammatikregeln und applikationsspezifischen Trainingsdaten in Form grammatikalisch schwer zu spezifizierender Sätze [WA05]. Die Informationen aus der semantischen Decodierung werden in Form eines semantischen Rahmens (*semantic slots*) an das Dialogmanagement übergeben.

1.4 Dialogmanagement

Das Dialogmanagement bestimmt aus der semantischen Decodierung über die semantischen Slots (*semantic slots*) die Dialogstrategie und legt damit fest, wie das System auf die (Sprach-)Eingaben des Benutzers reagiert. Dabei werden externe Quellen, wie z. B. das Wissen über den Benutzer oder den Dialogverlauf zur Entscheidungsfindung herangezogen. Zu einem Dialogmanagementsystem zählen auch Sprachsynthese (*text-to-speech*) (TTS)-Systeme. Die Antwortgenerierung ist praktisch das Gegenstück zur Sprachanalyse [Rus94].

In der bisher üblichen Anwendungspraxis ist das natürliche Sprachverstehen häufig in ein multi-modales Dialogsystem eingebettet. Dieses erlaubt neben der natürlichen Sprache auch Zeigegesten oder Emotionen in Form von Gesichtsausdrücken als Eingaben [SBG+08]. In vielen Fällen führt dies allerdings dazu, dass keine natürlichsprachige Kommunikation stattfindet, sondern dass sich die natürliche Sprache auf das Wesentliche beschränkt. Es wird von einem „Command&Control“ System gesprochen. Das Typische daran ist die Reduktion des Spracherkenners auf ein kleines Vokabular.

Ein wichtiger Aspekt dieser Arbeit ist, dass der Dialog möglichst „natürlich“ verlaufen soll, sodass sowohl der Benutzer als auch das System in den Dialog eingreifen und diesen steuern können (*Mixed-initiative dialog system*).

1.5 Integrierte Modellierung

In dieser Arbeit werden alle Verarbeitungsstufen, von der Spracherkennung bis zum Dialogmanagement, als Graphische Modelle (GM) beschrieben. Durch diese integrierte Darstellung kann die Wortakkuratheit aus der akustischen Erkennung bis zur höchsten semantischen Stufe fließen und entsprechend zur Entscheidung der Dialogebene beitragen. Ebenso ist aber auch eine Rückführung vom Dialogmanagement in den Spracherkenner möglich. Diese ermöglicht eine Verbesserung der Erkennungsleistung. Diese Modellierung hat gegenüber der seriellen Verwendung der einzelnen Komponenten den Vorteil, dass alle Parameter in die nächste Verarbeitungsstufe weitergeleitet und zum Teil auch wieder zurückgeführt werden. Zudem werden durch die integrierte Modellierung serielle Verarbeitungsschritte vereinfacht und die Antwortzeiten verkürzt.

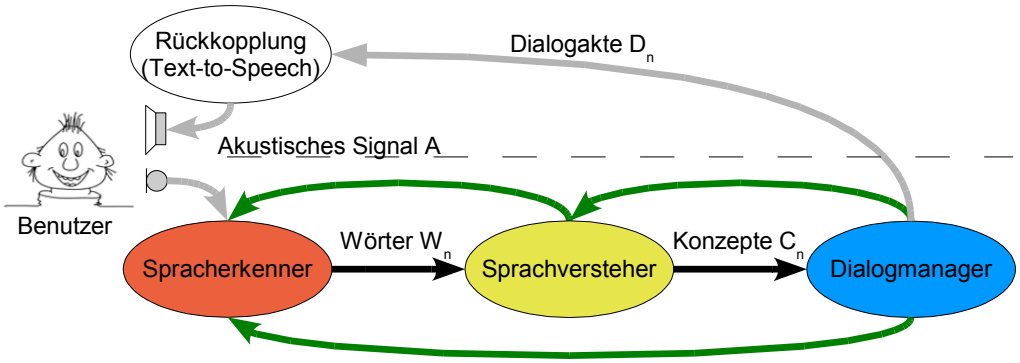


Abbildung 1.1: Komponenten und Schnittstellen der Modellierung: Die Spracherkennung liefert aufgrund der akustischen Eingangssignale eine orthographische Verschriftung der „semantischen Decodierung“ (Sprachversteher). Über die semantische Decodierung wird aus der orthographischen Verschriftung der semantische Rahmen an das Dialogmanagement geliefert. Das Dialogmanagement liefert sowohl der „semantischen Decodierung“, als auch dem Spracherkenner die kontextbezogenen Informationen zurück. Vom Dialogmanagement wird ein Sprachausgabesystem angesteuert.

1.6 Gliederung und Beitrag dieser Arbeit

In dieser Arbeit werden alle Verarbeitungsstufen einheitlich mit GM modelliert. Für jede Verarbeitungsstufe werden eigene Modellentwicklungen vorgestellt und mit gängigen Verfahren anhand von Experimenten evaluiert und begründet.

Bevor in Kapitel 3 das Anwendungsszenario mit einer Korpusbeschreibung für die Experimente vorgestellt wird, werden in Kapitel 2 die theoretischen Grundlagen ausgeführt. Nach einer kurzen Einführung der GM wird deren Flexibilität und Intuitivität am Beispiel eines Spracherkenners gezeigt. Anschließend werden insbesondere für das „Verstehen“ regelhaft gesprochener Äußerungen kontextfreie Grammatiken eingeführt. Des Weiteren werden in diesem Kapitel prinzipielle Arbeitsweisen von Dialogmanagementsystemen ausgeführt und mit der Einführung von Agenten eine entsprechende Realisierungsmöglichkeit dergleichen vorgestellt. Das Kapitel schließt mit den statistischen Evaluierungsmethoden, welche für anstehende Experimente benötigt werden.

Das Kapitel 4 befasst sich mit der semantischen Zuordnung von orthographisch verschrifteten Äußerungen. Dazu werden neben einem Referenzsystem verschiedene Modellierungen mithilfe von „erweiterten kontextfreien Grammatiken“ und GM vorgestellt. Das Ziel dieses Kapitels ist es, aus einer natürlich-gesprochenen Äußerung eine Menge an „semantischen Slots“ zu erhalten.

Die „semantischen Slots“ liefern die Daten für das Dialogmanagement, welches in Kapitel 5 behandelt wird. Darin werden zwei Verfahren vorgestellt, die sich in

der Berechnung der Dialogstrategie unterscheiden. Im „teilweise beobachtbaren markoffschen Entscheidungsprozess“ (POMDP) [You06] wird die Strategie in aufwendigen Rechenverfahren „offline“ berechnet, wohingegen ein mithilfe von GM modelliertes System die Strategie zur Laufzeit berechnen kann. Dadurch kann die Flexibilität erhöht werden und der Dialog möglichst „natürlich“ verlaufen.

In Kapitel 6 werden die Verarbeitungsstufen als Agenten in einem verteilten System prototypisch realisiert, bevor in Kapitel 7 die Arbeit mit einer Bilanz zusammengefasst wird und ein Ausblick auf mögliche weitere Arbeiten gegeben wird.

2

Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen dieser Arbeit gelegt. Der erste Teil befasst sich mit graphentheoretischen Grundbegriffen, welche für stochastische Beschreibungen eines Spracherkenners auf Wortebene in Abschnitt 2.2 benötigt werden. Anschließend in Abschnitt 2.3 wird die Zuordnung von Bedeutungen eines gesprochenen Wortes erläutert. In Abschnitt 2.4 werden die theoretischen Grundlagen für Dialogmanagementsysteme spezifiziert. Für eine Realisierung mit Agenten werden in Abschnitt 2.5 grundlegende Begriffe aus dem Bereich der „Künstlichen Intelligenz“ eingeführt. Die Gütekriterien zur Evaluierung, der im Rahmen dieser Arbeit durchgeführten Experimente, werden in Abschnitt 2.6 dargelegt.

2.1 Graphische Modelle (GM)

Graphische Modelle (GM) sind eine Mischung aus Wahrscheinlichkeits- und Graphentheorie [Jor01]. Die Graphen dienen zur intuitiven Darstellung von bedingten Unabhängigkeitsaussagen, dadurch enthält der Graph sogenannte Markov-Eigenschaften. Ein Knoten in einem GM repräsentiert eine Wahrscheinlichkeitsverteilung. Es können verschiedene Arten von GM verwendet werden, um unterschiedliche Unabhängigkeitsaussagen ausdrücken zu können. Durch die formalen Markov-Eigenschaften der Graphen ist stets sichergestellt, dass die zugrunde liegenden Wahrscheinlichkeitsaussagen eingehalten werden. Eine Berechnung in einem GM führt damit zum gleichen Ergebnis wie die direkte Berechnung über die Wahrscheinlichkeitstheorie, benötigt jedoch häufig eine signifikant geringere in der benötigten Zeitkomplexität und ist zusätzlich fast immer wesentlich intuitiver [Bilb].

In Abbildung 2.1 sind verschiedene Typen von GM dargestellt. Daraus ist ersichtlich, dass GM zum einen sehr flexibel eingesetzt werden können und zum anderen eine streng formale, abstrakte, mathematische, aber durch die Graphentheorie auch intuitive **Sprache** zur Beschreibung von kausalen und probabilistischen Problemen bieten. Dadurch können verschiedene Verfahren der Mustererkennung und

Wahrscheinlichkeitstheorie durch die einheitliche Sprache der GM zusammengefasst werden [Lau96].

Über die **Struktur** der GM können kausale und verknüpfte Elemente verbunden und deren Zusammenhänge intuitiv untersucht werden [Pea01]. Die Untersuchung kann mit standardisierten Verfahren (Algorithmen) als Entscheidung unter Unsicherheit erfolgen [Pea88, JLO90]. Durch die Struktur, die standardisierten Algorithmen und die Sprache der GM können Aussagen über entstehende Approximationsfehler gemacht werden. Diese entstehen, wenn die standardisierten Verfahren zur Berechnung zu komplex werden. Nach [Bila] entsprechen GM einer Art probabilistischer Datenstruktur (oder Datenbank). Die Strukturen und Parameter entsprechen dem Speichern von Informationen in der Datenbank. Die Algorithmen zum probabilistischen Schließen entsprechen dann den Anfragen an die Datenbank.

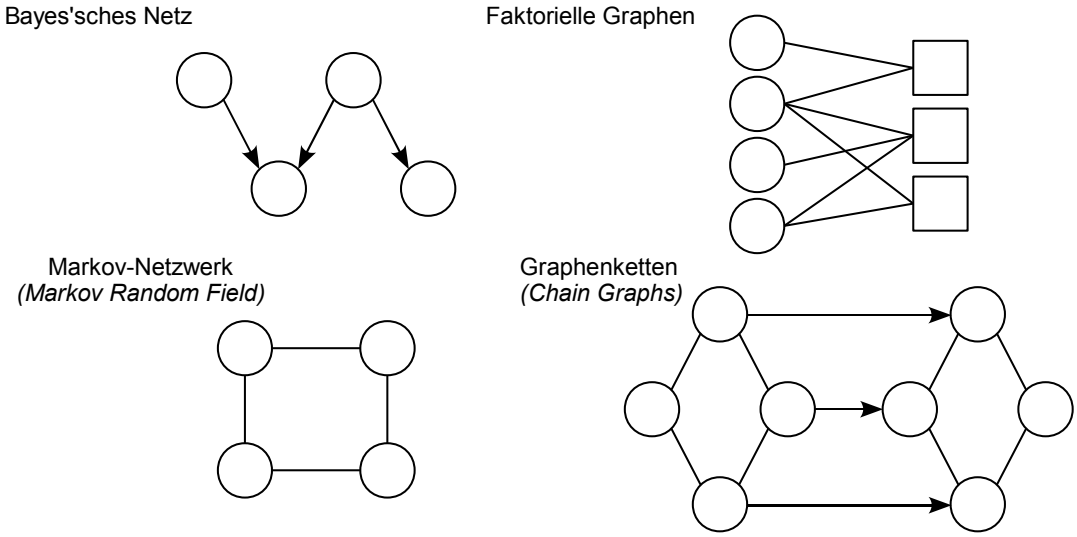


Abbildung 2.1: Beispiele für Graphische Modelle nach [Bilb].

2.1.1 Graphische Notationsformen

Ein Graph $\mathcal{G} = (V, E)$besteht aus einer endlichen Knotenmenge Vund einer Kantenmenge $E \subseteq V \times V$ von Paaren (u, v) mit $u \neq v$. Eine Kante $(v, v) \in E$ heißt **Schlinge**. Ein Graph heißt **ungerichteter** Graph, wenn für alle Kanten $(u, v) \in E$ auch $(v, u) \in E$ gilt. Man schreibt dann auch $\{u, v\} \in E$. Eine Kante heißt **gerichtete** Kante $(u, v) \in E$ und $(v, u) \notin \forall u, v \in V$. Gerichtete Graphen ohne Schlingen werden als **einfache Graphen** bezeichnet.

Die Menge der Knoten $V_v \subseteq V$ wird als **Vorgänger** oder **Elternknoten** (*parent*) von v bezeichnet, wenn $V_v \rightarrow v$ gilt. Der Knoten der Menge v wird dann auch als Nachfolger oder Kindknoten (*child*) von V_v bezeichnet.

Ein Graph heißt **zusammenhängend**, wenn es für alle $u, v \in V$ einen einfachen Weg (*path*) gibt, der u und v verbindet. Ein Graph $\mathcal{G}$ ist **azyklisch**, wenn er keine Kreise enthält. Dieser wird dann als **Wald** (*forest*) bezeichnet. Ist der azyklische Graph zusammenhängend, so wird dieser als **Baum** (*tree*) bezeichnet.

Ein Graph $\mathcal{G}_n = (V_n, E_n)$ heißt **vollständig** (*complete*), wenn alle Knoten V_n untereinander mit einer gerichteten oder ungerichteten Kante verbunden sind, d. h. $(\forall u, v \in \mathcal{G}_n)[\{u, v\} \in E_n]$. Ein Graph $\mathcal{G}' = (V', E')$ ist ein **Teilgraph** (*subgraph*) von $\mathcal{G} = (V, E)$, falls $V' \subseteq V$ und $E' \subseteq E$ ist. Ein Graph $\mathcal{G}$ heißt **bipartit**, wenn es eine Partition $V = V_1 \uplus V_2$, d. h. $V_1 \cup V_2 = V$ und $V_1 \cap V_2 = \emptyset$, $V_1 \neq V_2 \neq \emptyset$ und $E \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$ gibt.

Weitere Begriffe in der Graphentheorie sind die Cliquen ψ; diese sind **vollständige** Teilgraphen von $\mathcal{G}$, bei denen alle Knoten untereinander verbunden sind. Ein vollständiger Untergraph $\mathcal{G}'$ wird als **maximale Clique** bezeichnet.

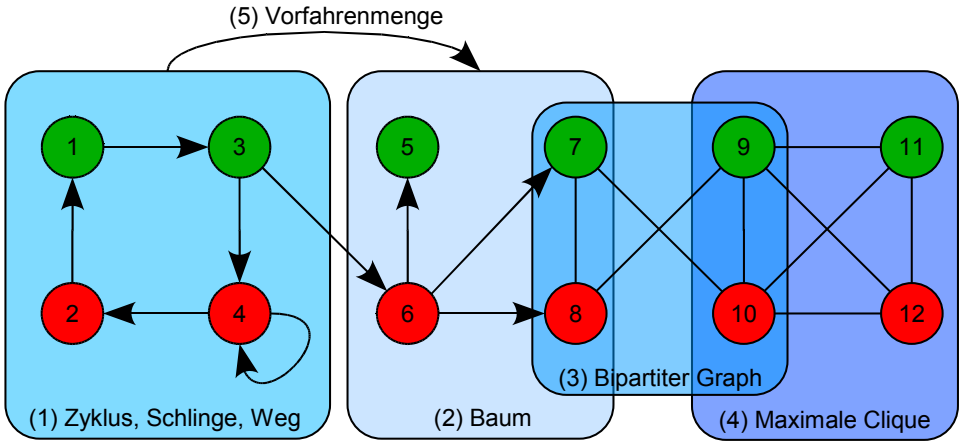


Abbildung 2.2: In (1) werden gerichtete Graphen mit einem Zyklus, einer Schlinge am Knoten 4 und einem gerichteten Weg zwischen $(3 \rightarrow 6)$ dargestellt. In (2) wird hingegen ein Graph ohne Zyklus dargestellt (Wald), in dem sogar jeder Knoten erreicht wird (Baum). In (3) ist ein ungerichteter vollständiger bipartiter Graph dargestellt, bei dem nach jedem ungeraden Knoten (grün) alle geraden Knoten (rot) erreicht werden. In (4) ist eine maximale Clique der Knoten $9-12$ dargestellt.

2.1.2 Arten von Knoten und Kanten

Im weiteren Verlauf dieser Arbeit wird die auf [Mur01, BB05] basierende Notation für die Darstellung der Wahrscheinlichkeitsdichtefunktion (WDF)[1] und die deterministischen Kantenbeziehungen verwendet (siehe Abbildung 2.3).

[1]Per Definition gilt für eine kontinuierliche WDF $p(x) = 0, \forall x$ (siehe [BW00]). In der Praxis wird jedoch der Wert der WDF als Wahrscheinlichkeit interpretiert, siehe z. B. [Bil98] und derart

Ein **Knoten** kann entweder **diskret** oder **kontinuierlich** sein. Des Weiteren kann ein Knoten entweder **beobachtet** oder **verborgen** sein [Mur01]. Bei einem beobachteten Knoten (grau) ist der Zustand der Zufallsvariablen bekannt. Diese werden z. B. verwendet, um Observierungen in das GM zu integrieren. Verborgene (*hidden*) Knoten (weiß) werden verwendet, um beispielsweise Markov-Ketten abzubilden. Eine diskrete Zufallsvariable, deren Wert genau N Zustände repräsentieren kann, wird mit einem Quadrat dargestellt. Ein kontinuierlicher Knoten wird mit einem Kreis dargestellt, weil dessen WDF durch z. B. eine Normalverteilung dargestellt wird.

Eine **gerichtete Kante** (siehe Abbildung 2.2) repräsentiert die Beziehung zwischen zwei Knoten, sie können nach [BB05] folgendes beschreiben: Ein **kontinuierlicher** Pfeil E_{ij} repräsentiert die Wahrscheinlichkeitsbeziehung zwischen den Knoten V_i nach V_j. Der Kindknoten V_j hat dann eine **bedingte** Wahrscheinlichkeit, welche vom Elternknoten V_i abhängig ist. Ein **gestrichelter Pfeil** E_{ij} stellt eine deterministische Beziehung zwischen zwei Knoten V_i und V_j dar. Dabei nimmt der Kindknoten V_j nur **einen** definierten Zustand, abhängig vom Elternknoten V_i, ein. Ein **geschwungener** Pfeil symbolisiert nach Bilmes [BB05] eine **bestimmende** Transition, welche z. B. eine Wort- oder Phonemgrenze beschreibt. Durch eine Bigramm-Wahrscheinlichkeit kann dann beispielsweise ein Wortübergang stattfinden. Diese Transition kann auch durch eine deterministische Beziehung beschrieben werden. In [BB05] werden noch **wechselnde** Beziehungen (dargestellt als geschwungener und kontinuierlicher Pfeil) zwischen V_i und V_j beschrieben, welche abhängig von V_i, die Transition nach V_j entweder deterministisch oder probabilistisch beschreiben.

Die Modellierung von Mustererkennungsproblemen mit GM wird in [AH08] ausführlich dargestellt. Diese Arbeit stützt sich wiederum auf [BB05, JGJS01, Jor01].

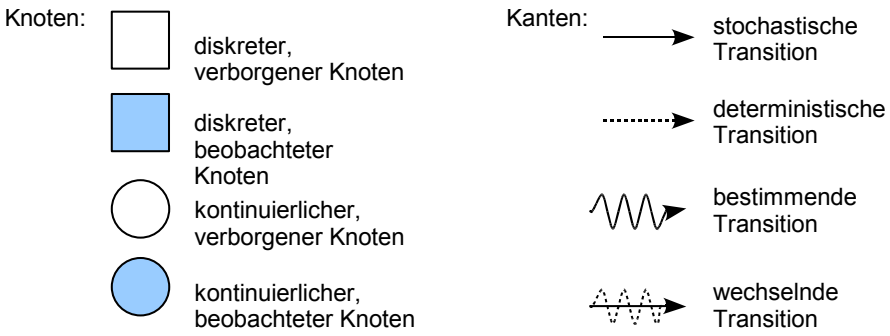


Abbildung 2.3: In dieser Arbeit verwendete Notation für Knoten- und Kantentypen [Mur01] mit den in der GMTK-Notation [Bil04] erweiterten Kantentypen für deterministische, bestimmende und wechselnde Transitionen.

verwendet [BZ02, YEH+02].

2.2 Spracherkennung mit stochastischen Modellen

Bei fließend gesprochener Sprache verbinden die Sprecher aufeinanderfolgende Wörter miteinander, sodass zum einen die Wortgrenzen nicht eindeutig aus dem Audiosignal hervorgehen und zum anderen der Anwender Wortbestandteile in der Nähe der Wortgrenzen möglicherweise gar nicht ausspricht oder zumindest nicht klar artikuliert. Heutige Spracherkenner müssen „verschluckte“ Teilsilben bzw. Wortkomponenten aus dem Sprachsignal eigenständig ergänzen. Dazu werden aus dem Audiosignal zunächst verschiedene Merkmale, wie Formanten oder LPC-Parameter[2], extrahiert. Des Weiteren ergeben sich je nach Vorverarbeitung sogenannte Merkmalsvektoren mit unterschiedlichen Komponenten, möglichst derart, dass sich unterschiedliche Laute in unterschiedlichen Merkmalsvektoren ausprägen und sich trennen lassen. Diese Ausprägung sollte sprecherunabhängig erfolgen.

Um ein bestimmtes Sprachsignal in die am besten geeignete Wortfolge zu übersetzen, benutzt ein Spracherkenner (siehe Abbildung 2.4) eigene hierarchische Wissensbasen (akustische Modellierung, Lexikon und Sprachmodell). In den folgenden Abschnitten wird auf die Modellierung, das Training und die Decodierung derartiger Wissensbasen eingegangen.

2.2.1 Akustische Merkmale

Das **akustische Modell** $P(\mathbf{x}|\mathbf{w})$ beschreibt die Zusammenhänge der Merkmalsvektoren $\mathbf{x}$ und der Wörter $\mathbf{w}$ des Vokabulars. Man spricht auch vom Klang der Wörter, gleich klingende Wörter werden auch als **Homophone** bezeichnet. Das **Sprachmodell** $P(\mathbf{w})$ spezifiziert die Wahrscheinlichkeit jeder Artikulation (Wörter). Sprachmodelle sind in der Regel als einfache **Bigramm-Modelle** realisiert.

In Abbildung 2.5 ist die typische Transformation von Schall in Kurzzeitspektren (*Frames*) dargestellt. Üblicherweise wird in einem Spracherkenner die Länge des Intervalls eines Frames auf 10 *ms* begrenzt, dies entspricht 80 Abtastungen bei 8 *kHz*. Aus jedem überlappenden Zeitfenster (*Frame*) wird ein Merkmalsvektor $\mathbf{x}$ (siehe Abbildung 2.5, unterste Zeile) mit dem analogen akustischen Signal und dem daraus quantisierten digitalen Signal gebildet[3].

Zur Berechnung der Merkmalsvektoren $\mathbf{x}$ sind nachfolgende Arbeitsschritte notwendig:

- Aufzeichnung und Digitalisierung der gesprochenen Sprache (Zeitsignal)

[2] Linear Predictive Coding (LPC) ist ein in der Audio-Signalverarbeitung und Sprachverarbeitung für die Audiodatenkompression verwendetes Verfahren; dabei wird die im Eingangssignal in Form von Korrelation (Abhängigkeit) aufeinanderfolgender Abtastwerte enthaltene Redundanz ausgenutzt und mithilfe eines Prädiktionsverfahrens, also einer Berechnung zukünftiger Signalwerte, eine Datenreduktion erreicht [MG76].

[3] In der Regel arbeiten heutige Spracherkenner mit $N = 39$ Merkmalen: 12 Mel-Frequenz-Cepstrum-Koeffizienten, 12Δ-Cepstrum-Koeffizienten, $12\Delta\Delta$-Cepstrum-Koeffizienten und je ein Wert für die Energie, Δ-Energie und $\Delta\Delta$-Energie [Rus94].

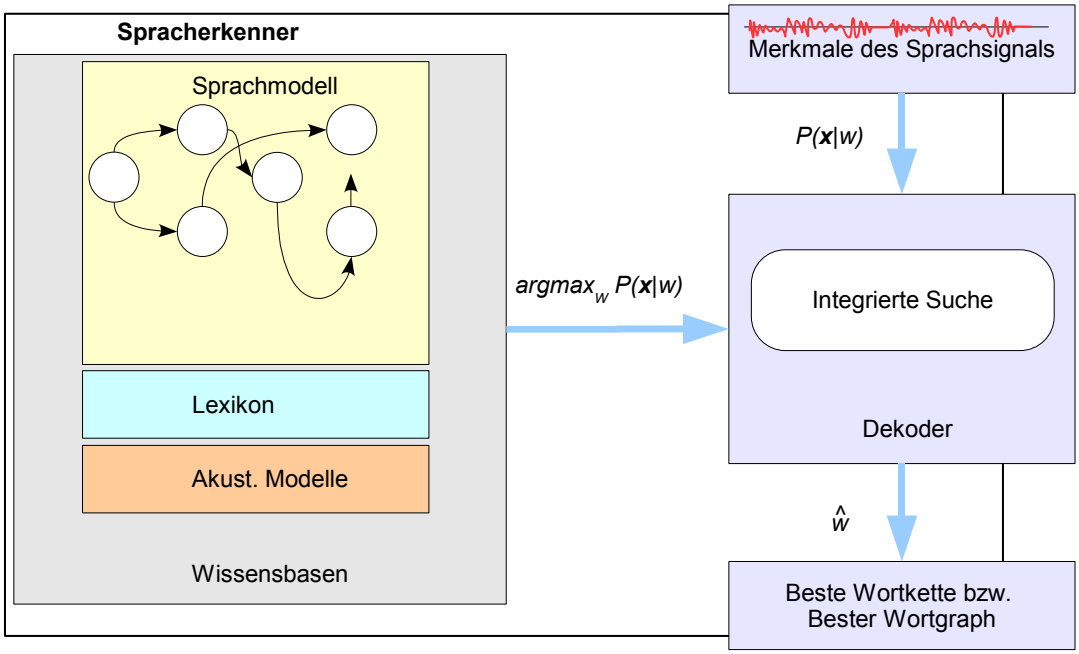


Abbildung 2.4: Zeitgleiche Suche nach den besten Worthypothesen in der aufgespannten Wissensbasis eines Spracherkenners. Die Parameter der Wissensbasen werden in einer Trainingsphase bestimmt.

- Fensterung des Zeitsignals in quasi-stationäre Abschnitte
- Berechnung des Kurzzeitleistungsspektrums eines Fensters
- Filterung des Spektrums mit einer Mel-Filterbank (ähnlich dem menschlichen Gehör, mel-bewertetes Spektrum)
- Logarithmieren des melskalierten Leistungsspektrums
- inverse diskrete Kosinus-Transformation (IDFT) des logarithmierten Spektrums (Cepstrum)

Phonem-/Triphonmodell

Das Wahrscheinlichkeitsmodell $P(\mathrm{x}|\mathrm{w})$ ist für alle Zeitfenster gleich die meisten Laute haben hingegen eine umfassende interne Struktur, z. B. hat Laut *t* zwei Stoppkonsonanten [Lin86]. Die interne Struktur von Lauten wird durch ein **Dreizustands-Lautmodell** dargestellt. Jeder Laut hat einen Anfangs-, Mitte- und Endzustand, und jeder Zustand hat seine eigene Verteilung über die Merkmale.

In [Rus94] wird ein weiteres Phänomen der Spracherkennung beschrieben: der Klang eines bestimmtes Lautes kann von einem anderen Laut in der Umgebung

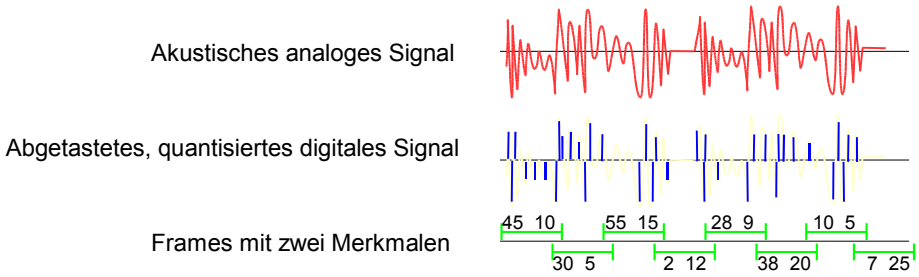


Abbildung 2.5: Zusammenhang zwischen dem akustischen Signal und dessen abgetasteten, quantisierten digitalen Signal; jedes Zeitfenster (*Frame*) wird hier durch zwei akustische Merkmale beschrieben.

beeinflusst werden. Bei dem Wort „süß“ wird das *s* in Erwartung des folgenden *ü* gerundet. Eine Lösungsvariante für dieses Phänomen bietet das sogenannte **Triphon-Modell**, womit diese Effekte im Spracherkenner erfasst werden können. Die Verwendung eines Triphon-Modells erhöht die Anzahl möglicher Zustände des temporären Prozesses aus n Lauten im ursprünglichen Lautalphabet auf $3\,n^3$, wodurch sich die Komplexität in der zeitgleichen Suche des besten Pfades erhöht.

2.2.2 Hidden-Markov-Modelle

In der Sprachverarbeitung haben sich die Hidden-Markov-Modelle (HMM) λ bewährt, da sie eine zeitgleiche **Segmentierung** und **Klassifizierung** der akustischen Parameter durchführen [RJ93]. Das HMM $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ ist ein zweistufiger gekoppelter stochastischer Prozess.

Der erste stochastische Prozess beschreibt die Übergangswahrscheinlichkeit eines Zustandes $q_t = s_j$ zum Zeitpunkt t in Abhängigkeit der vorausgegangenen Zustände $q_{t-1}, \ldots, q_1$. Dies geschieht durch eine Modellierung mit einer Markovkette erster Ordnung mit N Zuständen s_j, $1 \leq j \leq N$, deren Übergangswahrscheinlichkeiten a_{ij} in einer $N \times N$ Parametermatrix

$$\mathbf{A} = [a_{ij}]_{N \times N} \text{ mit } a_{ij} = P(q_t = s_j | q_{t-1} = s_i) \tag{2.1}$$

zusammengefasst werden. In einem Einsprungsvektor $\pi = (\pi_1, \ldots, \pi_n)^T$, mit

$$\pi_i = P(q_i = s_i) \text{ mit } \sum_{i=1}^{N} q_i = 1, \tag{2.2}$$

werden die Einsprungswahrscheinlichkeiten bestimmt.

Der zweite stochastische Prozess beschreibt in einer Emissionsmatrix

$$\mathbf{B} = [b_{ij}]_{N \times K} \text{ mit } b_{ij} = b_i(\mathbf{x}_j) = P(\mathbf{x_j} | q_t = s_i), \tag{2.3}$$

eine Ausgabeverteilung. Bei $b_i(\cdot)$ handelt es sich im Allgemeinen um eine Funktion, welche in den folgenden Unterabschnitten mit konkreten Verteilungsfunktionen beschrieben wird.

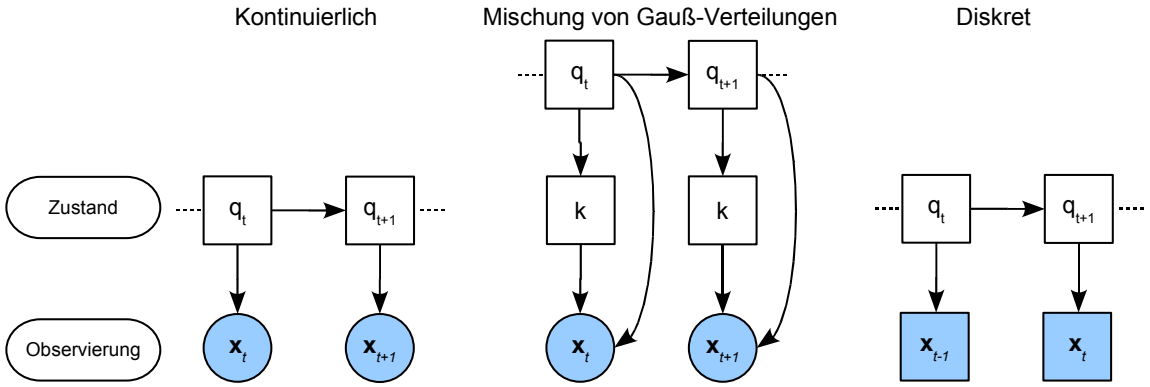


Abbildung 2.6: Beispiel eines kontinuierlichen HMM für die Modellierung akustischer Merkmale dargestellt als Graphisches Modell.

Kontinuierliche Hidden-Markov-Modelle

Im einfachsten Fall (siehe Abbildung 2.6 links) werden die Emissionen (bzw. Observierungen) durch „Gauß-Knoten“ ersetzt, wodurch die Emissionsmatrix $\mathbf{B}$ durch eine Normalverteilung mit $b_i(\mathbf{x}) = \mathcal{N}(\mathbf{x}, \mu_i, \Sigma_i)$ ersetzt wird.

Damit kann aus Abbildung 2.6 (links) folgende Verbundwahrscheinlichkeit

$$P(\mathbf{x}, \mathbf{q}) = P(q_1)\,\mathcal{N}(\mathbf{x}_1, \mu_{q_1}, \Sigma_{q_1}) \sum_{t=2}^{T} P(q_t|q_{t-1})\,\mathcal{N}(\mathbf{x}_t, \mu_{q_t}, \Sigma_{q_t}) \tag{2.4}$$

berechnet werden. $\mathcal{N}(\mathbf{x_t}, \mu_{q_t}, \Sigma_{q_t})$ beschreibt die Gauß'sche WDF mit dem Mittelwert μ_{q_t} und der Kovarianzmatrix Σ_{q_t}.

Kontinuierliche Hidden-Markov-Modelle mit Gauß Mixtur Modellierung

In der Sprachverarbeitung werden für **kontinuierliche** HMM (siehe Abbildung 2.6 Mitte) die WDF $b_j(\cdot)$ einer Beobachtung $\mathbf{x}_t$ durch Mixtur- oder Mischmodelle beschrieben [Fin03]. Für jeden Zustand s_i wird eine beliebige WDF, bestehend aus K-Mixturen mit dem Faktor c_{jk} gewichtete Normalverteilungen $b_{q_t}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t, \mu^k_{q_t}, \Sigma^k_{q_t})$, durch

$$b_j(\mathbf{x}) = \sum_{k=1}^{K} c_j^k \cdot \mathcal{N}(\mathbf{x}_k, \mu_j^k, \Sigma_j^k) \text{ und } \sum_{k=1}^{K} c_j^k = 1, \tag{2.5}$$

beschrieben.

Das GM in Abbildung 2.6 links faktorisiert die Beobachtungswahrscheinlichkeit $P(\mathbf{x}|\lambda)$ zu jedem Zeitpunkt t in einem dreistufigen Zufallsprozess, mit dem Zustand $q_t \in \mathcal{Q}$, der Mischverteilung $k \in K_{q_t}$ und der Ausgabeverteilung $\mathbf{x}_t$ zu:

$$\begin{aligned} P(\mathbf{x}, \lambda) &= \sum_{q \in \mathcal{Q}} \sum_{k \in K_{q_t}} P(\mathbf{x}, q, k|\lambda_t) \\ &= \sum_{\mathbf{q} \in \mathcal{Q}} P(q_1) \sum_{k}^{K_{q_1}} P(\mathbf{x}_1|q_1, k) \cdot \prod_{t=2}^{T} P(q_t|q_{t-1}) \sum_{k=1}^{K_{q_t}} P(\mathbf{x}_t|q_t^k)\ P(k|q_t) \\ &= \sum_{q \in \mathcal{Q}} \pi_{q_1} \sum_{k=1}^{K_{q_1}} c_{q_1}^k \mathcal{N}(\mathbf{x}_1, \mu_{q_1}^k, \Sigma_{q_1}^k) \prod_{t=2}^{T} a_{q_{t-1}, q_t} \sum_{k=1}^{K_{q_t}} c_{q_t}^k \mathcal{N}(\mathbf{x}_t^k, \mu_{q_t}^k, \Sigma_{q_t}^k) \end{aligned} \tag{2.6}$$

In der Spracherkennung wird oft eine unkorrelierte Beobachtung $\mathbf{x}_t$ vorausgesetzt, wodurch eine diagonale Kovarianzmatrix $\Sigma_{q_t}^k$, mit reduzierten zu schätzenden Parametern verwendet werden kann.

Diskrete Hidden-Markov-Modelle

Die Parameteroptimierung kontinuierlicher HMM ist durch die schrittweise Anpassung mit einem Gauß'schen Mixtur Modell sehr rechenaufwendig [Nor95].

Bei diskreten HMM (siehe Abbildung 2.6 rechts) werden die akustischen Merkmale $\mathbf{x}$ durch einen Vektor-Quantisierer (VQ) auf eine relative Häufigkeit $y_{q_t}^k$ abgebildet, wenn $\mathbf{x}$ im Unterraum M_k liegt: Die Verbundwahrscheinlichkeit kann dann folgendermaßen faktorisiert werden:

$$P(\mathbf{x}|\lambda) = \sum_{q \in \mathcal{Q}} \pi_{q_1} y_{q_t}^k \prod_{t=2}^{T} a_{q_{t-1}, q_t} y_{q_t}^k \tag{2.7}$$

Das Codebuch des VQ wird in den meisten Fällen mit dem k-means-Algorithmus generiert [NR94].

Im Rahmen dieser Arbeit werden die diskreten HMM nicht für die akustische Wortmodellierung, jedoch für die Modellierung der Semantik in gesprochenen Äußerungen verwendet (siehe Kapitel 4). Für Untersuchungen zu diskreten HMM, insbesonders mit multiplen Observierungen, wird hier auf die Arbeiten von Schenk verwiesen [SSRR08, SSR08a, SSR08b].

2.2.3 Lernverfahren und Parameterschätzung

Für stochastische Modelle werden Lernverfahren benötigt, welche die Auftrittswahrscheinlichkeit einer bestimmten Beobachtung maximieren. Dazu sind I Trainingsbeispiele mit den Beobachtungen $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_I\}$ gegeben. Damit lässt

sich die gesamte Auftrittswahrscheinlichkeit mit

$$P(\mathcal{X}|\lambda) = \log \prod_{i=1}^{I} P(\mathbf{x_i}|\lambda) = \sum_{i=1}^{I} \log P(\mathbf{x_i}|\lambda) \tag{2.8}$$

berechnen [Gha98].

Dabei enthält λ die Parameter, d. h. im Wesentlichen die WDF eines GM, jedoch nicht dessen Netzstruktur $\mathcal{G}^D$ beschreibt. Für ein kontinuierliches HMM (siehe Abbildung 2.6 links) werden die Parameter nach Gleichung 2.4 so bestimmt, dass die Produktionswahrscheinlichkeit $P(\mathbf{x}, \mathbf{q})$ einer Beobachtung $\mathbf{x}$ wie folgt maximiert wird, d. h.

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, P(\mathbf{x}, \lambda) = \underset{\lambda}{\operatorname{argmax}} \sum_{\mathbf{q}\in\mathcal{Q}} P(\mathbf{x}, \mathbf{q})|\lambda) = \sum_{\mathbf{q}\in\mathcal{Q}} \pi_{q_1} b_{q_1} \cdot \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{x}_t), \tag{2.9}$$

Allgemein werden beim Lernen nach der Maximum-Likelihood Methode (ML) die Parameter $\hat{\lambda}$ so bestimmt, dass diese die Auftrittswahrscheinlichkeit $\mathcal{L}$ nach

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, \mathcal{L}(\lambda|\mathcal{X}), \tag{2.10}$$

iterativ maximieren.

Durch die Summierung der Produkte lässt sich Gleichung 2.9 nicht analytisch optimieren. In der Literatur wird die Anpassung der Parameter der HMM mithilfe des Baum-Welch-Algorithmus ermittelt, der eine Spezialform des Expectation-Maximization (EM)-Algorithmus darstellt [DLR77]. Der EM-Algorithmus im Allgemeinen und der Baum-Welch-Algorithmus als dessen spezifischer Realisierung werden nachfolgend erläutert.

Expectation-Maximization-Algorithmus

In der Spracherkennung kann es vorkommen, dass $P(\mathbf{x}_t|\lambda)$ unvollständig ist, z. B. wenn mangels Trainingsdaten einige Zustände des HMM während des Trainings nicht beobachtet (*hidden*) werden können [DLR77]. Wird daraus gefolgert, dass neben der Menge der beobachtbaren Zustände $\mathcal{X}$ eine Menge an weiteren Zuständen $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ existiert und die Funktion $P(\mathbf{x}, \mathbf{y}|\lambda)$ damit vollständig ist, kann die Gesamtdatenwahrscheinlichkeit $\mathcal{L}$ für Bayes'sche Netze (BN) mit

$$\log \mathcal{L}(\lambda|\mathcal{X}, \mathcal{Y}) = \log P(\mathcal{Y}, \mathcal{X}|\lambda) = \log \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}, \mathcal{X}|\lambda), \tag{2.11}$$

berechnet werden.

Mit einer Verteilungsfunktion Q über die verborgenen Zustände $\mathcal{Y}$ kann eine untere Schranke für $\mathcal{L}$ nach [Gha98] hergeleitet werden:

$$\log \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}, \mathcal{X}|\lambda) = \log \sum_{\mathbf{y}\in\mathcal{Y}} Q(\mathbf{y}) \frac{P(\mathbf{y}, \mathcal{X}|\lambda)}{Q(\mathbf{y})} \tag{2.12}$$

$$\geq \sum_{\mathbf{y}\in\mathcal{Y}} Q(\mathbf{y}) \log \frac{P(\mathcal{X}, \mathbf{y}|\lambda)}{Q(\mathbf{y})} \tag{2.13}$$

$$\geq \sum_{\mathbf{y}\in\mathcal{Y}} Q(\mathbf{y}) \log P(\mathcal{X}, \mathbf{y}|\lambda) - \sum_{\mathbf{y}\in\mathbf{y}} Q(\mathbf{y}) \log Q(\mathbf{y}) \tag{2.14}$$

$$=: \mathcal{F}(Q|\lambda) \tag{2.15}$$

Der EM-Algorithmus wechselt zwischen der Maximierung der untere Schranke $\mathcal{F}(Q, \lambda) \leq \mathcal{L}(\lambda)$ [BPSW70, DLR77]:

Abschätzungs- (*estimation*) Schritt:

$$Q_{k+1} \leftarrow \operatorname*{argmax}_{Q} \mathcal{F}(Q, \lambda_K) \tag{2.16}$$

Maximierung- (*maximization*) Schritt:

$$\lambda_{k+1} \leftarrow \operatorname*{argmax}_{Q} \mathcal{F}(Q_{k+1}, \lambda) \tag{2.17}$$

$$\lambda_{k+1} \leftarrow \operatorname*{argmax}_{Q} \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathcal{X}, \lambda_K) \log P(\mathbf{y}, \mathcal{X}|\lambda) \tag{2.18}$$

Nach [DLR77] wird Gleichung 2.18 als EM Algorithmus präsentiert, jedoch wird die Interpretation mittels Hilfsfunktion $\mathcal{F}$ (siehe Gleichung 2.15) oft nicht dargestellt. Die initialen Parameter des Modells λ werden bis $\mathcal{F} = \mathcal{L}$ in jedem EM Schritt verbessert. Dabei erfolgt in jedem Schritt ein Wechsel von Schätzen und Maximieren der Parameter, bis der Algorithmus konvergiert.

Baum-Welch-Algorithmus

Zum Trainieren der Parameter des EM-Algorithmus wird auf das HMM bezogen auch der Begriff **Baum-Welch-Algorithmus** verwendet. Dieser berechnet effizient die Parameter des HMM mit dem Vorwärts- und Rückwärtsalgorithmus.

Vorwärtsalgorithmus

Der Vorwärtsalgorithmus berechnet einen Zustand s_i zum Zeitpunkt t mit der Bedingung, dass alle bisherigen Beobachtungen $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t)$ ein Teil der Realisierung sind.

Basisfall: Die Gesamtwahrscheinlichkeit, Zustand s_i im Zeitpunkt 1 zu erreichen, ist die Anfangswahrscheinlichkeit für s_i multipliziert mit der Wahrscheinlichkeit, dass $\mathbf{x}_1$ ausgegeben wird:

$$\alpha_1(i) = \pi_i \, b_i(\mathbf{x}_1), 1 \leq i \leq N. \tag{2.19}$$

Rekursion:

$$\alpha_{t+1}(i) = \left\{ \sum_{j=1}^{N} \alpha_t(i) \, a_{ij} \right\} \, b_i(\mathbf{x}_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N. \tag{2.20}$$

Die Gesamtwahrscheinlichkeit, Zustand s_j im Zeitpunkt $t+1$ zu erreichen, ist die Summe der Wahrscheinlichkeiten von einem beliebigen Zustand s_i in Zustand s_j zu kommen, unter Ausgabe des nächsten Merkmalssatzes $\mathbf{x}_{t+1}$.

Gesamt:

$$P(\mathcal{X}|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{2.21}$$

Die Gesamtwahrscheinlichkeit der Realisierung ist die Summe der Wahrscheinlichkeiten, zum Zeitpunkt T in einen beliebigen Zustand zu kommen.

Rückwärtsalgorithmus

Der Rückwärtsalgorithmus funktioniert analog; hier wird betrachtet, dass sich das HMM zum Zeitpunkt t im Zustand s_i befindet und noch folgende Beobachtungssequenz $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ gesehen wird:

$$\beta_t(i) = P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T | q_t = s_i, \lambda) \tag{2.22}$$

Vorwärts-Rückwärts-Algorithmus

Durch die Kombination von Gleichung 2.21 mit Gleichung 2.22 werden die Parameter des Modells mit

$$P(q_t = s_i|\mathcal{X}, \lambda) = \frac{P(q_t = s_i, \mathcal{X}|\lambda)}{P(\mathcal{X}|\lambda)}, \tag{2.23}$$

rekursiv über die Zeit t berechnet.

Mit der Verwendung von $\alpha_t(s_i)$ und $\beta_t(s_i)$ kann die WDF zum Zeitpunkt t, die auch a-posteriori Wahrscheinlichkeit $\gamma_t(s_i)$ genannt wird, nach

$$\gamma_t(s_i) = P(q_t = s_i|\mathcal{X}, \lambda) = \frac{\alpha_t(s_i)\beta_t(s_i)}{P(\mathcal{X}|\lambda)} \tag{2.24}$$

berechnet werden.

In diesem Abschnitt wurde das Training für die in der Sprachverarbeitung wichtigen kontinuierlichen HMM mittels Baum-Welch-Algorithmus gezeigt. GM erlauben jedoch weitere Lernverfahren und Variationen des EM-Algorithmus. Der Schätzschritt (*Expectation*) wird dabei durch den Verbund Baum (VB)-Algorithmus berechnet. Der Maximierungsschritt (*Maximization*) ist abhängig von der zu modellierenden WDF zu bestimmen; es existieren jedoch geschlossene algorithmische Lösungen für GM. An dieser Stelle wird auf Heckerman „Learning in GM“ verwiesen [Hec98].

2.2.4 Decodierung

Aus dem akustischen Signal (siehe Abbildung 2.5) werden Merkmalsvektoren (bzw. Beobachtungen) $\mathbf{x}$ extrahiert, womit der optimale Pfad (Zustandsfolge) $\hat{q}$ eines Modells λ mit der maximalen a-posteriori Wahrscheinlichkeit

$$\hat{q} = \underset{q}{\operatorname{argmax}}\, P(q|\mathbf{x}, \lambda) \text{ mit} \qquad P(q|\mathbf{x}, \lambda) = \frac{P(\mathbf{x}, q|\lambda)}{P(\mathbf{x}|\lambda)}, \tag{2.25}$$

berechnet wird. Da die Produktionswahrscheinlichkeit $P(\mathbf{x}|\lambda)$ bei einem festen Modell λ und Observierung $\mathbf{x}$ konstant bleibt kann die Gleichung 2.25 zu

$$\hat{q} = \underset{q}{\operatorname{argmax}}\, P(q|\mathbf{x}, \lambda) = \underset{q}{\operatorname{argmax}}\, P(\mathbf{x}, q|\lambda), \tag{2.26}$$

vereinfacht werden.

Viterbi-Algorithmus

Der EM-Algorithmus auf das hier beschriebene HMM-Schätzproblem angewandt, beschreibt eine entscheidungsüberwachte Variante des Baum-Welch-Algorithmus, welcher unter dem Namen des Viterbi-Algorithmus bekannt ist [ST95].

Der Viterbi-Algorithmus berechnet iterativ, ähnlich zum Vorwärtsalgorithmus (siehe Abschnitt 2.2.3), den partiell optimalen Pfad $\delta_t(i)$ bei einer gegebenen Observierung $\mathbf{x}_t$ mit

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, q_1, q_2, \ldots, q_{t-1}, q_t = s_i|\lambda). \tag{2.27}$$

Die Berechnung von $\delta_t(i)$ in Gleichung 2.27 unterscheidet sich zum Vorwärtsalgorithmus einzig darin, dass anstelle einer Summierung über die in den Vorgängerzuständen vorliegende Wahrscheinlichkeitsverteilung eine **Maximierung** erfolgt. Die Entscheidung $\delta_t(i)$ ist nur lokal optimal, erst mit der Auswertung $\delta_T(i)$ kann die gesamte Folge $\hat{q}$ decodiert werden.

Backtracking: Die Decodierung des besten Pfades erfolgt durch Rückverfolgung (siehe Abbildung 2.7) über den „Rückwärtszeiger" $\psi_t(j)$,der für jedes $\delta_t(j)$ den optimalen Vorgängerzustand mit

$$\psi_t(j) = \underset{i}{\operatorname{argmax}}\, \delta_{t-1}(i)\, a_{ij}, \tag{2.28}$$

berechnet. Der optimale Pfad kann beginnend mit

$$\hat{q}_T = \underset{t}{\operatorname{argmax}}\, \delta_T(i), \tag{2.29}$$

rekursiv nach

$$\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1}) \tag{2.30}$$

in umgekehrter zeitlicher Reihenfolge decodiert werden.

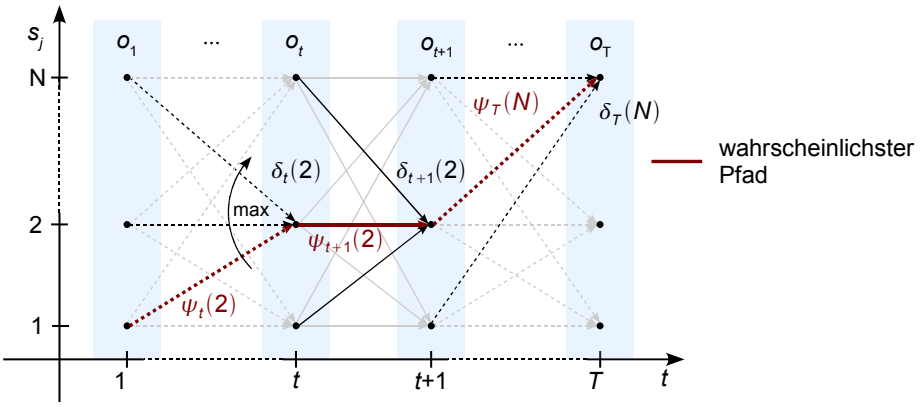


Abbildung 2.7: Trellisdiagramm und wahrscheinlichster Pfad, gefunden mit dem Viterbi-Algorithmus [SR07].

Trellisdiagramm: Die Decodierung mithilfe des Viterbi-Algorithmus [Vit67] kann graphisch in Form eines Trellisdiagramm dargestellt werden (siehe Abbildung 2.8). Durch den wahrscheinlichsten Pfad werden hier die einzelnen Phonemmodelle verbunden. Die Phonemfolge, die durch den Verlauf des wahrscheinlichsten Pfades angezeigt wird, entspricht im Idealfall dem gesprochenen Satz.

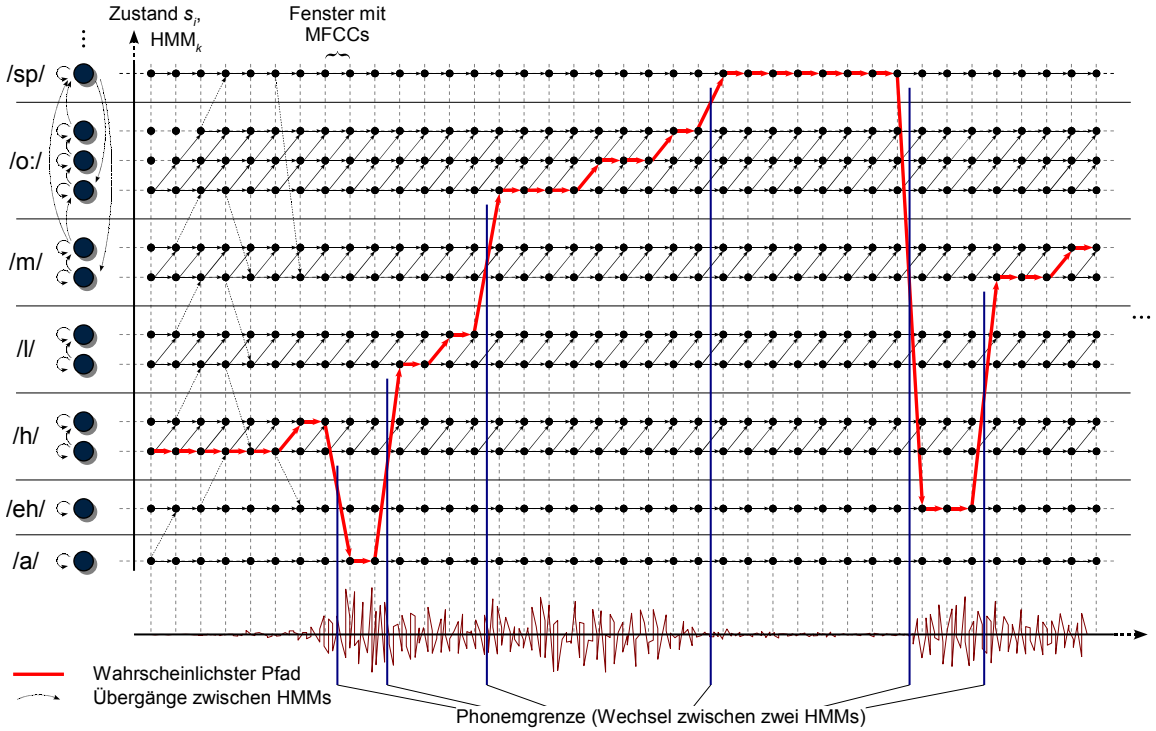


Abbildung 2.8: Trellisdiagramm zur Erkennung eines gesprochenen Satzes [SR07].

Um die Anzahl der Kombinationsmöglichkeiten einzuschränken, benutzt man entweder ein Wörterbuch, das alle zu erkennenden Wörter in Phonemschreibweise enthält, Grammatiken, die die Aussprache als Regeln beschreiben, darüberhinaus n-Gramme, die die Wahrscheinlichkeiten für bestimmte Phonemkombinationen liefern, sowie Sprachmodelle, die Wahrscheinlichkeiten für bestimmte Wortkombinationen angeben.

Durch Zuordnung der erkannten Phonemfolge zu einem Wort (etwa durch ein Aussprachelexikon) kann letztlich aus dem Sprachsignal das gesprochene Wort oder der ganze Satz erkannt werden.

2.2.5 Sprachmodell

Die Erkennungsleistung von Wörtern mit HMM kann durch ein Sprachmodell $P(w)$ verbessert werden. In der Sprachverarbeitung werden dazu entweder Grammatikmodelle (Formale Sprachen) oder N-Gramm-Modelle verwendet. Dieses wird mit

der Wahrscheinlichkeit

$$P_{\text{N-Gramm}}(\mathbf{w}) = P(w_N|w_1,\ldots,w_{N-1}) = \frac{P(w_1,\ldots,w_N)}{P(w_1,\ldots,w_{N-1})}, \tag{2.31}$$

$$\approx P(w_1)\prod_{n=2}^{N}\underbrace{P(w_n|w_{n-(N-1)},\ldots,w_{n-1})}_{N-\text{Symbole}}, \tag{2.32}$$

aus einer Sequenz von N-Wörtern beschrieben. Die Verwendung von N-Gramm-Sprachmodellen erfordert große Textkorpora der Anwendungsdomäne. Das Problem der spärlich vorhandenen Trainingsdaten wird hier in Kapitel 3 mithilfe „Absolute Discounting“ gelöst.

In Abbildung 2.9 ist das Sprachmodell $P(\mathbf{w})$ mit dem kontinuierlichen HMM in ein GM integriert. Zusätzlich kann die HMM Decodierung mit einem Lexikon (Wortmodell) verbessert werden. Die weiteren Verarbeitungsstufen werden in den folgenden Kapiteln behandelt.

2.2.6 Anwendung von Graphischen Modellen

Die HMM für die Spracherkennung werden mit den bereits eingeführten GM (siehe Abschnitt 2.1) realisiert. Das Ziel ist die integrierte Modellierung über mehrere Verarbeitungsstufen; eine Verbesserung der Erkennungsleistung der HMM-Modellierung ist nicht das Ziel. Dazu werden im Folgenden erweiterbare GM in der Notation der verwendeten GM-Software (*Graphical Model Toolkit*) (GMTK) erläutert [Bilb]. Eine Erweiterung der klassischen HMM-Modellstruktur für die Decodierung wird in Abbildung 2.10 gezeigt.

Das Modell (siehe Abbildung 2.10) zeigt auf der untersten Ebene die observierten kontinuierlichen Merkmalsvektoren $\mathbf{x}_1,\ldots,\mathbf{x}_T$. Dies ist das klassische HMM mit kontinuierlichen Beobachtungen $\mathbf{x}$ und diskreten, verborgenen Zuständen $q_1,\ldots,q_T$. Die Verbundwahrscheinlichkeit der untersten Ebene wird mit

$$\pi_{q_i}\prod_{i=1}^{T-1}a_{i\,i+1}\prod_{i=1}^{T}b_{i\,j} \tag{2.33}$$

berechnet. Dabei beschreibt $b_{i\,j}$ die Beobachtungsfunktion mit der Wahrscheinlichkeit einer Beobachtung $\mathbf{x}_j$ zum Zeitpunkt i. Die Kardinalität der diskreten Zustände q_i ist äquivalent der Anzahl der Phoneme N_{phon}. Nach jedem Zeitschritt t wird mithilfe der Phonemübergangsfunktion $\mathbf{p}^{tr}$ geprüft, ob die observierten Merkmale $\mathbf{x}_t$ das Ende eines Phonems p_t beschreiben. Zu welchem Zeitpunkt ein Phomenübergang stattfindet, hängt vom Phonem p_t ab und erfolgt über eine bestimmende Transition (Zickzack-Linie).

Mithilfe eines Aussprache-Wörterbuches (siehe Abbildung 2.9) wird ausgehend von der gebildeten Phonemkette $\mathbf{p}$ die Wortposition w^p des Phonems p_t zu jedem

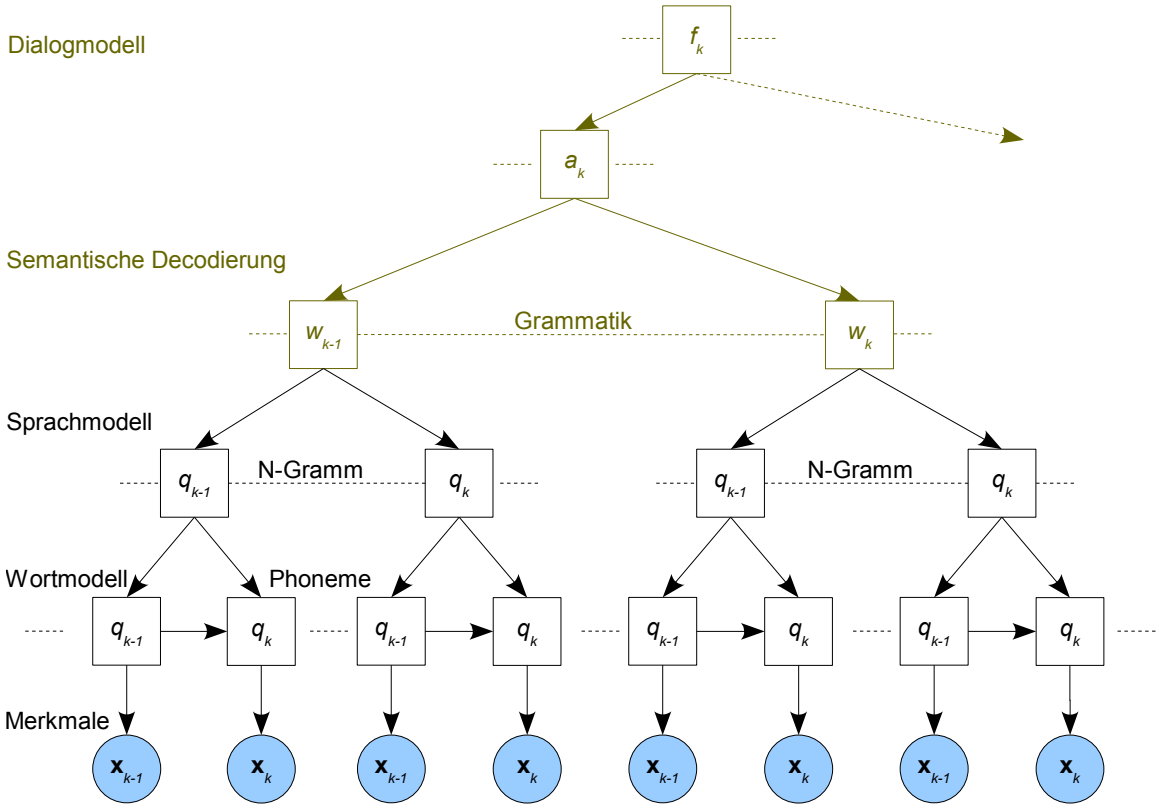


Abbildung 2.9: Vereinfachte Darstellung eines mit Graphischen Modellen realisierten kontinuierlichen HMM (unten), durch ein Wortmodell (Mitte) und ein Sprachmodell (oben) erweitert. Die Übergänge im Wort- bzw. Sprachmodell erfolgen mittels stochastischen N-Gramm Modellen. Die angedeutete Erweiterung (grün) bezieht sich auf die Decodierung semantischer Inhalte (Sprachverstehen) und das stochastische Dialogmanagement. Dies ist aber bereits ein Vorgriff auf das nachfolgende Kapitel.

Zeitpunkt t bestimmt. Mit der Wortposition des Wortes kann wiederum auf das Phonem zurückgeschlossen werden. Des Weiteren wird zu jedem Zeitpunkt t geprüft, ob ein Wortübergang w^{tr} stattfindet. Ein Wortübergang kann nur dann stattfinden, wenn auch ein Phonemübergang stattfindet. Der Wortübergang $\mathbf{w}^{tr}$ ist ebenfalls eine bestimmende Transition, d. h. wenn $w_{t-1}^{tr} = 0$, dann ist die Transition des Wortes $w_{t-1} \rightarrow w_t$ eine deterministische Funktion. Wenn aber ein Wortübergang stattfindet, d. h. $w_{t-1}^{tr} = 1$, dann ist w_t eine Zufallsvariable entsprechend dem Bigramm Sprachmodell (siehe Gleichung 2.34). Aus diesem Grund ist die Transition $w_{t-1} \rightarrow w_t$ sowohl stochastisch (durchgehende Linie), als auch bestimmend (Zickzack-Linie). Eine bedeutende Bedingung ist der beobachtete Zustand $= 1$. Er beschreibt das Ende der Beobachtung einer gesprochenen Äußerung. Damit wird gewährleistet, dass alle Wege im Trellis im Endzustand des GM zusammengeführt werden.

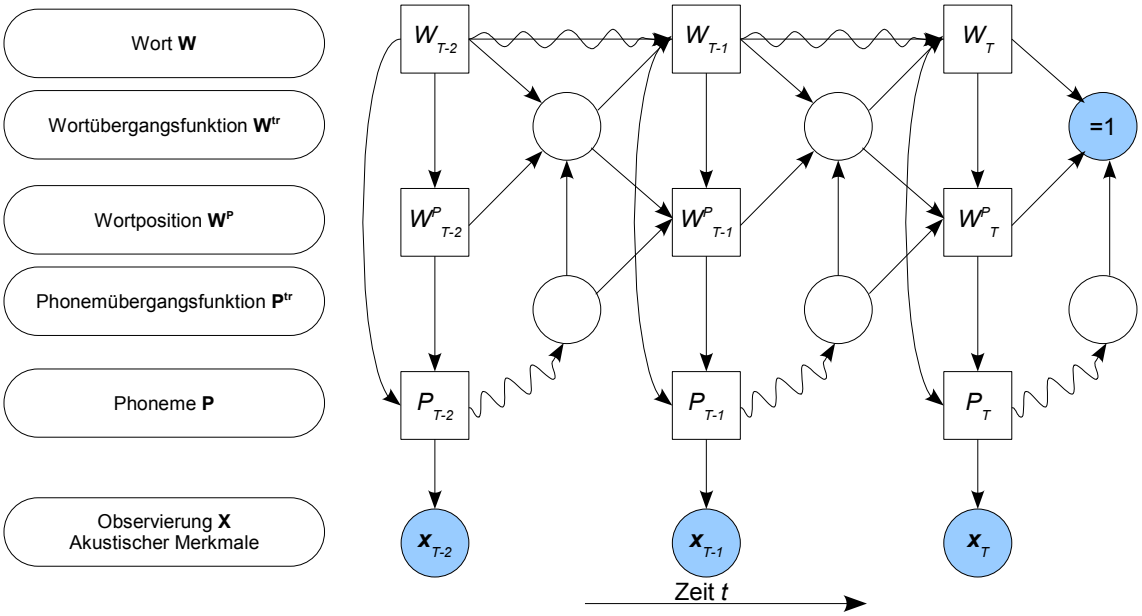


Abbildung 2.10: Bigramm Modell in GMTK-Notation [BZ02]. Über die beobachtbaren Merkmalsvektoren $\mathbf{x}$ kann eine Phonemkette $\mathbf{p}$ bestimmt werden. Die weitere Modellierung erfolgt sowohl stochastisch, als auch bestimmend (Zickzack-Linie). Mit der Beobachtung $= 1$ wird gewährleistet, dass alle Wege im Endzustand des Trellis zusammengeführt werden.

$$P(\mathbf{w}) = \prod_{i=1}^{T} P(w_t|w_{t-1}) \tag{2.34}$$

Dagegen wird bei einem Trigramm-Modell die Wahrscheinlichkeit über die Wortsequenz $\mathbf{w}$ faktorisiert nach

$$P(\mathbf{w}) = \prod_{i=1}^{T} P(w_t|w_{t-1}, w_{t-2}) \tag{2.35}$$

mit den beiden vorigen Wörtern w_{t-1} und w_t. Das GMTK-Modell für die Erkennung von gesprochener Äußerung mittels Trigramm wird in Abbildung 2.11 dargestellt. Mit der zusätzlichen Ebene „voriges Wort“, welches das Wort w_{t-2} repräsentiert, wird das bestehende Modell zu einem Trigramm-Modell erweitert. Für den Fall, dass zum Zeitpunkt t kein Wortübergang stattfindet, wird $w_t = w_{t-1}$ vom vorigen Zeitschritt $t - 1$ übernommen. Findet aber ein Wortübergang statt, dann wird w entsprechend der Faktorisierung (siehe Gleichung 2.35) berechnet. Dazu werden die Transitionen $w_{t-1}^p \rightarrow w_t$ und $w_{t-1} \rightarrow w_t$ in Abbildung 2.11 bestimmend (Zickzack-Linie) dargestellt.

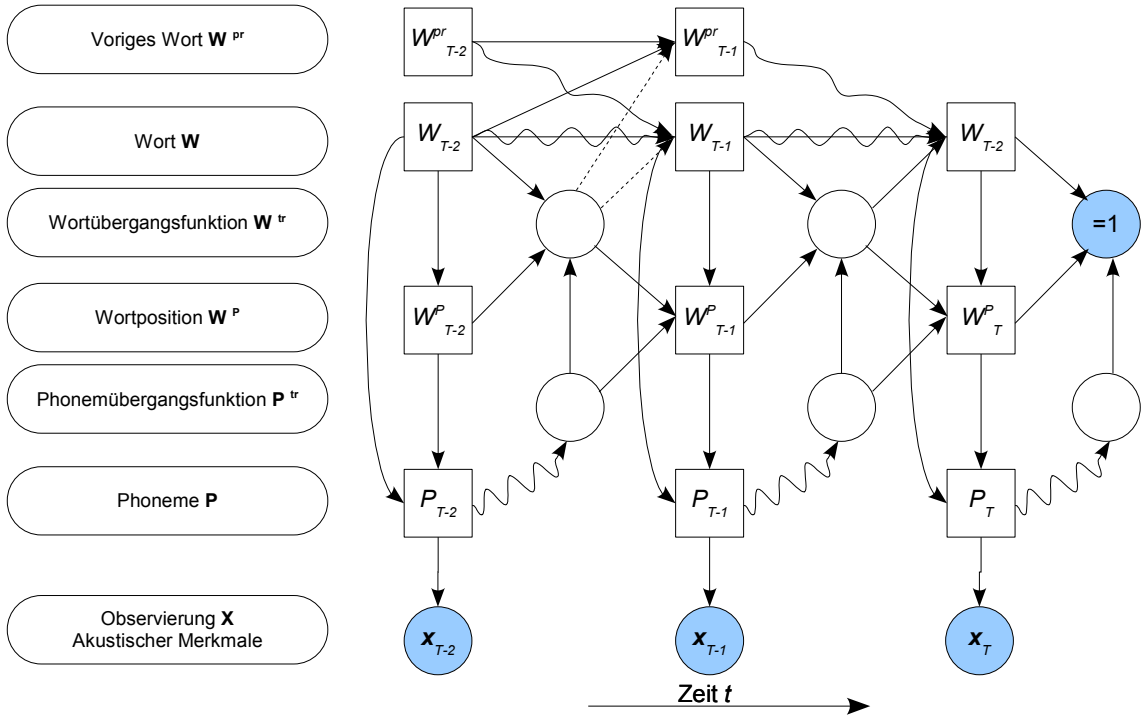


Abbildung 2.11: Trigramm Modell zur Erkennung von natürlich gesprochener Sprache in GMTK Notation [BZ02]. Als Erweiterung zum Bigramm Modell wird der Wortknoten w_{t-1}^{p} eingeführt. Dieser trägt über eine bestimmende Transition zur Entscheidung entsprechend Gleichung 2.35 über das Wort w_t bei.

2.3 Semantische Decodierung

Mit der semantischen Decodierung erfolgt eine Abbildung zwischen der realen Welt und einem semantischen Darstellungsschema. Das semantische Wissen gesprochener Sätze kann mit verschiedenen Möglichkeiten dargestellt werden. Wissen in der Künstlichen Intelligenz (KI) lässt sich als eine Menge von Fakten, Regeln, Prozeduren, Modellen, Daten und Heuristiken, die in KI-Systemen zur Problemlösung verwendet werden, definieren [RN03]. Die Wissensrepräsentation ist außerdem notwendig, um umfangreiches Spezialwissen eines Experten schematisch darzustellen. Mithilfe von Repräsentationsmechanismen kann das Wissen interpretierbar gemacht werden. Die populärsten Methoden der Wissensrepräsentation sind die Prädikatenlogik, regelbasierte Wissensverarbeitung, Semantische Netze und Rahmen (*frames*) [LC08].

2.3.1 Semantische Netze

Die semantischen Netze sind ein wichtiger Agent zur Darstellung von Wissen in sprachverarbeitenden Systemen. Mithilfe von GM wird Wissen über Beziehungen zwischen Objekten dargestellt. In diesem Kapitel werden natürlichsprachliche Aussagen mit verschiedenen Methoden in ein semantisches Netz überführt. Ein Beispiel für semantische Netze zeigt Abbildung 2.12.

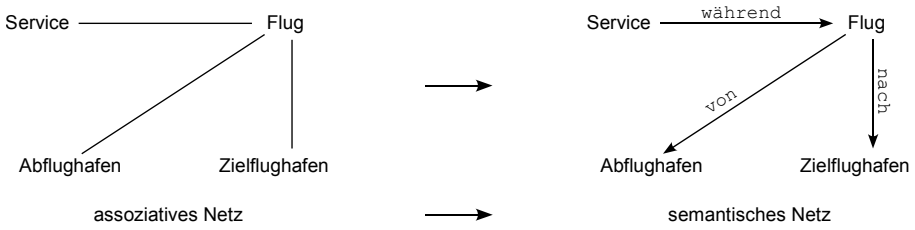


Abbildung 2.12: Zur Veranschaulichung wird ein semantisches Netz einer Flugauskunftsdomäne betrachtet, welches aus Konzepten (Objekten) und Beziehungen (Kanten) besteht. Ein Konzept c kann mehrere Unterkonzepte c_i beinhalten; ein Konzept, welches Wörter w vom Spracherkenner beinhaltet, wird im Folgenden als Wortklasse l bezeichnet.

Die Knoten des Netzes entsprechen eindeutigen Objekten, die Kanten entsprechen den Prädikaten. Üblicherweise werden in semantischen Netzen zweistellige Prädikate beschrieben. Alle im Netz repräsentierten Aussagen sind implizit konjunktiv (UND-) verknüpft. Die Richtung der Kanten in einem semantischen Netz ist maßgebend zur Beschreibung der Beziehung notwendig. Jedoch ist die Darstellung mehrstelliger Prädikate nur durch die Einführung neuer Hilfsknoten möglich.

Für den Inferenzmechanismus bei semantischen Netzen gibt es verschiedene Möglichkeiten. Entweder verwendet man sogenannte „Wenn-Dann-Regeln“, wobei der Wenn-Teil als Fakt im semantischen Netz ermittelt wird, oder spezielle Suchalgorithmen im semantischen Netz, z. B. um herauszufinden, in welcher Beziehung zwei entfernte Knoten im Netz zueinander stehen (durch Ermittlung der Schnittpunkte der von ihnen ausgehenden Graphen).

In Kapitel 4 werden verschiedene Verfahren der Transformation einer sprachlichen Äußerung in semantischen Netzen dargestellt. Dazu werden unter anderem kontextfreie Grammatiken verwendet.

2.3.2 Erweiterte Kontextfreie Grammatiken

Eine Grammatik ist als ein Beschreibungsmittel für formale Sprachen definiert, womit Parserwerkzeuge generiert werden können. Die Syntax von Programmiersprachen wird z. B. mit kontextfreien Grammatiken (*context-free grammars*) (CFG) beschrieben [BKW02]. Neben den CFG werden häufig Reguläre Ausdrücke (*regular*

expressions) (RE), welche in einen äquivalenten endlichen Automaten umgewandelt werden können, zur Beschreibung einer Untermenge einer gesprochenen Äußerung verwendet. Die erweiterten kontextfreien Grammatiken (*extended context-free grammars*) (ECFG) sind verbreitete **Mischformen** der beiden Grammatiktypen, bei denen die terminalen Symbole der CFG durch RE ersetzt werden.

Das Beispiel einer ECFG in Gleichung 2.36 nach [BK08] kann in einen äquivalenten Glushkov-Automaten [Glu60] überführt werden (siehe Abbildung 2.13).

$$\begin{aligned} S &\rightarrow a^* \, T \, b^* \\ T &\rightarrow c \, T \, d \mid \mathcal{E} \end{aligned} \tag{2.36}$$

Definition: Eine ECFG **G** ist definiert als 4-Tupel der Form (N, Σ, P, S), wobei N nicht-terminale, Σ terminale Zeichen, $S \in N$ das Startsymbol und P die Menge der Produktionen der Form $A \longrightarrow L_A$ beschreiben mit $A \in N$ und L_A eine reguläre Sprache über dem Alphabet $\Sigma \cup N$ [BKW02].

Die **rechtsentwickelnde Sprache** L_A bildet somit die gleiche Klasse wie eine kontextfreie Sprache. In den letzten Jahren haben sich ECFG vor allem im *World Wide Web* durch Extensible Markup Language (XML)-Anwendungen etabliert[BK08].

Nicht-deterministischer endlicher Automat

Die Sprache L_A kann durch einen nicht-deterministischen endlichen Automaten (NDA) der Form $M_A = (Q_A, \delta_A, s_A, F_A)$, mit einer Menge an Zuständen $Q_A = \Sigma \cup N$, einer Menge an Transitionen δ_A, einem Startzustand s_A und einer Menge an Endzuständen F_A, definiert werden.

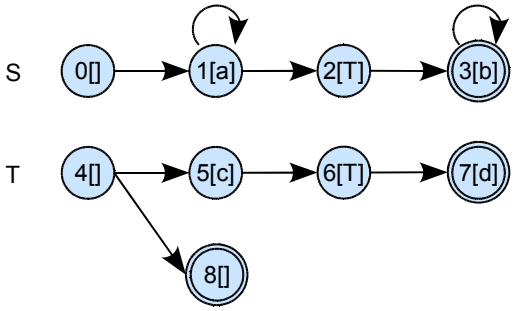


Abbildung 2.13: Beispiel einer erweiterten kontextfreien Grammatik, dargestellt als Glushkov-Automat [Glu60].

Ein nicht terminales Zeichen N kann eine weitere Produktionsregel beschreiben. Die Grammatik bleibt innerhalb der KLEENESCHEN HÜLLE mit den Operatoren $*, +$ und $?$ abgeschlossen. Mithilfe dieser Konstruktion erhält man aus einer Reihe von Produktionsregeln eine hierarchische Struktur, welche die **semantische** Bedeutung von gesprochenen Sätzen wiedergeben kann.

2.4 Dialogstrategie

Nach [McT04] wird eine Dialogstrategie nach der **Initiative** während des Dialoges, der Art der **Verifikation** und der **Modellform** zur technischen Beschreibung des Dialogablaufs unterschieden.

2.4.1 Initiative

Im einfachsten Fall liegt die Initiative beim System (*system directed*). Das bedeutet, es ist eine feste Reihenfolge gespeichert, in der das System Informationen einholt bzw. ausgibt. In diese Reihenfolge kann der Benutzer nicht eingreifen, außer das System stellt ihm explizit verschiedene Auswahlmöglichkeiten zur Verfügung (siehe Abbildung 2.14 links).

Die Initiative kann auch beim Benutzer liegen; dann wird dies als ein benutzergesteuerter (*user lead*) Dialog bezeichnet. In diesem Fall stellt das System Freiheiten zur Verfügung, die dem Benutzer erlauben, die Eingaben in beliebiger Reihenfolge oder auch mehrere Eingaben auf einmal zu machen. Vom System werden mit gezielten Fragen die offenen semantischen Slots gefüllt.

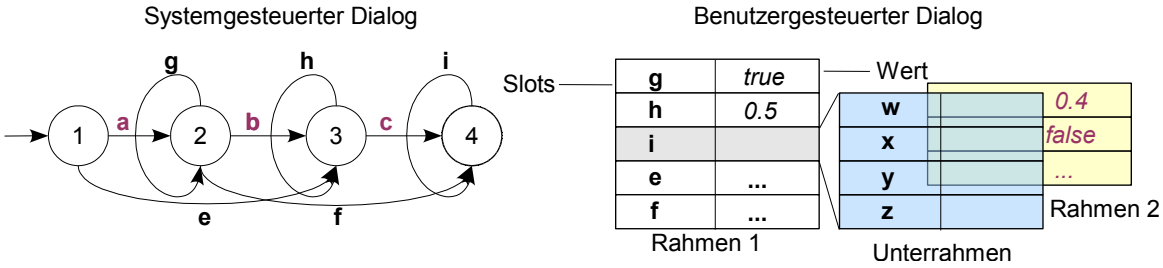


Abbildung 2.14: Modelle von Dialogmanagementsystemen: Systemgesteuerter Dialog, realisiert als endlicher Automat (links) und benutzergesteuerte offene Dialoge, realisiert mit semantischen Rahmen.

In Abbildung 2.14 werden zwei prinzipielle Verfahren gegenübergestellt. Bei der gemischten Initiative (*mixed-initiative*) können sowohl System als auch Benutzer die Initiative des Dialoges übernehmen. Im Rahmen dieser Arbeit wird ein derartiges Verfahren mithilfe von GM realisiert.

2.4.2 Verifikation

Ein weiteres Unterscheidungskriterium eines Dialogmanagementsystems ist die Verifikation der Benutzeraussagen auf ihre Richtigkeit. **Explizite Verifikation** bedeutet, dass die Benutzereingabe in einer eigenständigen Frage vom Dialogmanagementsystem wiederholt wird und das System nach einer **expliziten** Bestätigung in den

nächsten Dialogzustand übergeht. In dieser Arbeit wird hingegen mit der **impliziten Verifikation** die zu verifizierende Aussage im Rahmen der nächsten Frage wiederholt.

2.4.3 Systemstruktur

Einem Dialogmanagementsystem können unterschiedliche Modelle zur Dialogsteuerung zugrunde liegen, entweder ein endlicher Zustandsautomat (*finite-state machine*) (siehe Abbildung 2.14 links), eine formularfüllende Struktur (*form filling structure*) (siehe Abbildung 2.14 rechts) oder ein agentenbasiertes System (*agent based sytem*) (siehe Abschnitt 2.5).

Der **endliche Zustandsautomat** hat eine fest vorgeschriebene Reihenfolge der Zustände, die (in Abhängigkeit der Benutzereingaben) ineinander übergehen. Er ist besonders geeignet für klar strukturierte Dialoge mit einer geringen Anzahl an Zuständen. Es wird nur ein kleines Vokabular benötigt und die technischen Anforderungen an die Satzanalyse und die Sprachsynthese sind in der Regel gering, sodass das System weniger fehleranfällig ist. Oft können vorgefertigte Sätze mit einigen eingefügten Wörtern als Antworten genutzt werden. Damit wird eine gegenüber der vollständigen Synthese natürlichere Ausgabe erzeugt, was wiederum stark zur Zufriedenheit des Benutzers beiträgt. Der endliche Zustandsautomat stößt allerdings sehr schnell an seine Grenzen, wenn viele Zustände komplex miteinander verbunden werden oder eine implizite Verifikation implementiert werden soll. Dadurch werden die Dialoge verlängert. Überinformation kann weder verarbeitet noch ignoriert werden, sondern führt zu einer Fehleingabe [McT04].

Die **formularfüllende Struktur** stellt eine Vorlage (*template*) mit Leerstellen (*slots*) zur Verfügung. Der Benutzer kann die Reihenfolge seiner Angaben selbst wählen und mehrere Angaben auf einmal machen, auch Angaben, nach denen nicht gefragt wurde. Diese Überinformation kann verarbeitet werden, indem die betreffenden „semantischen Slots“ (siehe Kapitel 3) mithilfe der semantischen Decodierung (siehe Kapitel 4) befüllt werden.

Bei den **agentenbasierten Systemen** übernimmt ein Agent die Dialogsteuerung. Diese Softwareagenten bezeichnen eine Form der künstlichen Intelligenz, die zu eigenständigem Verhalten fähig ist, indem sie unabhängig von Benutzereingriffen aufgrund eigener Initiative Aktionen auslöst. Dabei reagiert sie auf Änderungen der Umgebung, lernt aufgrund zuvor getätigter Entscheidungen bzw. Beobachtungen und kommuniziert mit anderen Agenten bzw. dem Benutzer [Grü06]. Im folgenden Abschnitt werden die theoretischen Grundlagen für die in dieser Arbeit verwendeten Softwareagenten ausführlich dargestellt.

2.5 Agenten

Agenten[4] beobachten ihre Umgebung über Sensoren, bemerken Veränderung und setzen **autonom** entsprechende Handlungen ab [LC08]. Die zu beobachtende Umgebung ist normalerweise nicht-deterministisch, d. h. die Agenten müssen auf unerwartete Ereignisse reagieren können.

Ein Agent besteht aus einer Menge von Zuständen $S = \{s_1, s_2, \ldots, s_n\}$ und Aktionen $A = \{a_1, a_2, \ldots, a_n\}$ und erfüllt folgende Funktionen:

$$\textit{Aktion} : S^* \rightarrow A \tag{2.37}$$

$$\textit{Umgebung} : S \times A \rightarrow P(S) \tag{2.38}$$

Eine Interaktion mit einem Agenten kann in einem Verlauf h wie folgt wiedergegeben werden:

$$h : s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \tag{2.39}$$

Je nach Agentenarchitektur verfügt ein Agent über weitere Funktionen, wie z. B. *see* : $S \rightarrow P$ oder *next* : $I \times P \rightarrow I$ mit den internen Zuständen I und der Perzeption P. Weitere Funktionen für Agenten werden in [Wei99] ausführlich dargestellt.

2.5.1 Intelligente Agenten

Nach [Wei99] werden Agenten als **intelligent** bezeichnet, wenn sie nachfolgende Kriterien erfüllen:

Reaktionsfähig, d. h. schnell Veränderungen in ihrer Umgebung bemerken.

Proaktiv die Initiative übernehmen, um entschlossen ihr Ziel zu erreichen.

Sozial, d. h. mit anderen Agenten agieren können, um gemeinsame Ziele zu verfolgen.

Architektur als Unterscheidungskriterium für intelligente Agenten:

- logische Agenten (*logic based agents*)
- reaktionsfähige Agenten (*reactive agents*)
- Vorstellung-Wunsch-Absichts Agenten (*belief-desire-intention agents*) (BDI)
- Geschichtete Agenten

Sehr häufig werden BDI-Agenten verwendet. Sie halten mit dem Faktor γ, welche die Veränderung der Umgebung beschreibt, das Gleichgewicht zwischen der Reaktionsfähigkeit (*event driven*) und der Proaktivität (*goal directed*).

[4] hier: Softwareagent, welcher mithilfe von Sensoren die Umgebung in einem Modell wahrnimmt und gewisse Aufgaben eingeständig erfüllen kann [RN03].

Zielrichtung und Verhalten: Je nach Aufgabengebiet kann die Umgebung mit verschiedenen Agenten betrachtet werden z. B. Roboter-Agenten koordinieren Fahrwege, Internet-Agenten stellen Informationszusammenhänge her oder ein menschlicher Agent der mit seinen Augen, Ohren und Organen die Umgebung wahrnimmt und entsprechend reagiert. Nach [RN03] existieren verschiedene Ausprägungen von intelligenten Agenten (siehe Abbildung 2.15). Sie werden nach deren Zielrichtung und Verhalten unterschieden:

Kognitive Agenten haben eine Vorstellung über ihre Umwelt und können dementsprechend planen.

Reaktive Agenten haben keine Vorstellung über ihre Umwelt und reagieren ausschließlich auf äußere Einflüsse.

Reflexives Verhalten, wenn die Agenten eine Aufgabe zu erledigen bekommen.

Bewusstes Verhalten, wenn die Agenten bestimmte Ziele verfolgen.

	Kognitive Agenten	Reaktive Agenten
Bewusstes Verhalten	Intentionale Agenten	Absichtsgetriebene Agenten
Reflexives Verhalten	Modulbasierte Agenten	Tropistische Agenten

Abbildung 2.15: Unterscheidung von Agententypen nach deren Verhalten und kognitiven Fähigkeiten: Bewusstes Verhalten; ein intentionsbasierter Agent agiert kognitiv und zielbasiert. Hingegen reagiert ein absichtsgetriebener Agent aufgrund bestimmter ihm zugeteilter Aufgaben. Ein modulbasierter Agent hat eine Vorstellung von seiner Umwelt (kognitiv), reagiert aber reflexiv. Im Gegensatz dazu arbeiten die tropistischen Agenten zielgerichtet.

Im Rahmen dieser Arbeit werden für das Dialogmanagement intentions-basierte, intelligente Agenten (*intensional agents*) verwendet. Für die Wissensverarbeitung in der Wissensbasis (Knowledge Base) (KB) werden hier reflexartige Agenten (*reflex-based agents*) verwendet.

2.5.2 Multiagenten

In einem Multiagentensystem werden mehrere Agenten auf verteilten Systemen eingesetzt. Ein Agent agiert dabei als Koordinator (*Master*) und regelt die Kommunikation. Ein Multiagentensystem bietet eine Plattform für geographisch verteilte Systeme mit verschiedenen Komponenten und einem breiten Aufgabenspektrum.

Das hier entwickelte Agentensystem kommt in einer kognitiven Küche, einem Teilprojekt von CoTeSys[5] zum Einsatz. Darin sind die verteilten Küchengeräte (Herd, Backofen, Kaffeemaschine, Toaster) via verteilte Agenten in einem Multiagentensystem ansteuerbar. Ebenso agieren darin die hier entwickelten Methoden des natürlichsprachigen Mensch-Maschine-Dialogs als eigenständige Agenten.

2.5.3 Graphische Modelle und Agenten

Die GM (siehe Abschnitt 2.1) werden hier zur Modellierung, Entscheidungsfindung und Aktionsplanung des gesamten Mensch-Maschine-Dialogs innerhalb eines Agentensystems eingesetzt. Sie können sowohl regelbasiert als auch für die stochastische Modellierung verwendet werden. Mit GMTK existiert dazu ein Toolkit zum Experimentieren. Damit sind GM derzeit, im Gegensatz zu den Agenten, nicht parallelisierbar [BZ02].

2.6 Experimente

In jedem Kapitel werden für die entwickelten Systeme Experimente mit Hypothesen unter bestimmten Annahmen (siehe Kapitel 3) durchgeführt. Eine Überprüfung dieser Hypothesen erfolgt mit Experimenten, in denen die ermittelten Resultate mit „Referenzsystemen“ verglichen werden [Kro01]. In diesem Abschnitt wird die Versuchsdurchführung und die Bewertung im Bereich der Informationsabfrage (*information retrieval*) anhand der üblichen Gütekriterien erläutert [Rij79]. Sie bilden die Grundlage für alle in dieser Arbeit durchgeführten Experimente.

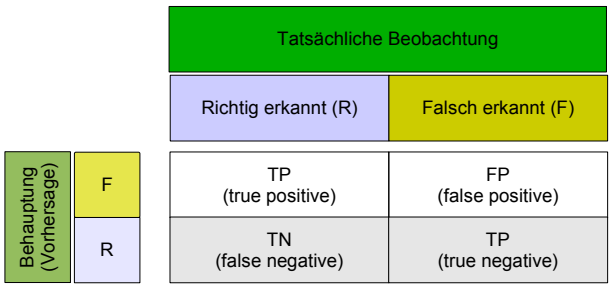


Abbildung 2.16: Binäre Klassifikation als Entscheidungskriterium für die Evaluierung der Experimente.

[5] CoTeSys ist ein Projekt des „Cluster of Excellence“ der TU-München und steht für „Cognition for Technical Systems“ [BBW07].

2.6.1 Kreuzvalidierung

Die Ergebnisse für die semantische Decodierung werden aufgrund geringer Annotierungsmengen mittels *k*-facher Kreuzvalidierung (*cross validation oder leave-one-out*) erzielt. Der Nachteil dieses Verfahrens ist, dass durch die Kreuzvalidierung eine weitere Stratifizierung nicht möglich ist und es dadurch in Einzelfällen zu Messfehlern kommen kann.

2.6.2 Versuchsdurchführung

Im Rahmen dieser Arbeit erfolgt eine Trennung der Versuche für die semantische Decodierung und das Dialogmanagement.

Semantische Decodierung

Mit der semantischen Decodierung wird hier die Zuordnung gesprochener Äußerungen zu einem semantischen Inhalt (z. B. Slots) verstanden (siehe Kapitel 4). Als Messgröße dient das binäre Entscheidungskriterium (siehe Abbildung 2.16) der F Score (siehe Gleichung 2.42 und 2.43) basierend auf der semantischen „Slot-Ebene“, d. h. die Effizienz der Systeme wird mit einer sogenannten „Goal Accuracy Rate“ bestimmt. Dieses Messverfahren berücksichtigt

- die Präzision (siehe Gleichung 2.40), welche das Verhältnis der richtig semantisch decodierten Wörter zu allen gefundenen semantischen Konzepten wiedergibt. Dieser Wert wird auch **positiver Vorhersagewert** genannt und entspricht der stochastischen Relevanz.

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2.40}$$

- den **Negativen Vorhersagewert** (*Recall*) (siehe Gleichung 2.41), welcher das Verhältnis der richtig semantisch decodierten Wörter im Verhältnis zu allen relevanten Konzepten beschreibt. Er entspricht der **statistischen Sensibilität**.

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.41}$$

Mit allen möglichen Decodierungen wird P „Precision“ zu 100% und bei richtiger Decodierung einer relevanten Antwort wird R „Recall“ 100%. Um aus den semantisch decodierten Wörtern eine bessere Aussagekraft zu erhalten, wird mit der F-Score das harmonische Mittel aus P und R gebildet:

$$F(R, P) = \frac{2\,R\,P}{R + P} \tag{2.42}$$

$$RR(R,P) = \frac{P}{P+R} \tag{2.43}$$

Diese Verfahren werden sowohl auf Ebene der semantischen Slots, als auch auf der Satzebene durchgeführt. Ein Satz wird als **falsch erkannt**, wenn sich ein einzelnes semantisch decodiertes Wort vom Referenzsystem unterscheidet. Die Aussagekraft dieses Verfahren beschränkt sich auf den direkten Vergleich mit dem Referenzsystem.

Dialogmanagement

Um Folgebewertungsfehler zu vermeiden, erfolgt die Bewertung des Dialogmanagements getrennt von der semantischen Decodierung. Die Kriterien für einen falschen Dialog sind, dass falsch erkannte semantische Slots durch das Dialogsystem nicht korrigiert werden. Dagegen sind die Kriterien für einen erfolgreich durchgeführten Dialog, dass mit Fehler behaftete semantische Slots durch das Dialogsystem korrigiert werden und der Dialog erfolgreich zu Ende geführt wird.

Segreganz oder negativer Vorhersagewert (*negative predictive value*) (NPV) zeigt die Trennfähigkeit des Ergebnisses. Sie gibt den Anteil der richtig als negativ (richtig negativ) erkannten Ergebnisse an der Gesamtheit der als negativ erkannten Ergebnisse an.

$$P(\mathrm{TN}|\mathrm{N}) = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} \tag{2.44}$$

Relevanz entspricht der stochastischen bedingten Wahrscheinlichkeit:

$$P(\mathrm{TP}|\mathrm{P}) = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{2.45}$$

Korrekturklassifikationsrate (*Accuracy*) wird mit

$$P(\text{richtig klassifiziert}) = \frac{\mathrm{TN}+\mathrm{TP}}{\mathrm{FN}+\mathrm{TN}+\mathrm{TP}+\mathrm{TN}} \tag{2.46}$$

bestimmt und ist ein Gütekriterium der richtigen Klassifikation der semantischen Informationen.

Falschklassifikationsrate von semantischen Informationen ist ein wichtiges Maß für die Benutzerzufriedenheit. Fehldiagnosen führen dazu, dass der Benutzer subjektiv unzufrieden über das System urteilt. Die Falschklassifikation wird mit

$$P(\text{falsch klassifiziert}) = \frac{\text{TN}}{\text{FN} + \text{TN}} \tag{2.47}$$

berechnet.

Receiver Operator Characteristic stellt die Abhängigkeit der Effizienz zur Fehlerrate dar [Faw06].

2.7 Zusammenfassung des Kapitels

Dieses Kapitel befasste sich mit den theoretischen Grundlagen der Graphischen Modellierung, auf welchen die nachfolgenden Kapitel aufbauen. In Abschnitt 2.1 wurden die graphentheoretischen Grundbegriffe erläutert. In Abschnitt 2.2 wurden die GM auf die stochastische Modellierung der Spracherkennung angewendet. Dabei wurden die Lernverfahren und die Decodierung mittels Viterbi-Algorithmus ausführlich dargelegt. Alternative Verfahren zur stochastischen Modellierung wurden in Abschnitt 2.3 mit den ECFG vorgestellt. In Abschnitt 2.4 wurden Grundprinzipien des Dialogmanagements erläutert, welche mithilfe von Agenten in Abschnitt 2.5 realisierbar sind. Grundprinzipien zur Evaluierung der Experimente wurden abschließend in Abschnitt 2.6 ausgeführt.

3

Anwendungsszenario

Stochastische Modelle, wie sie in dieser Arbeit verwendet werden, benötigen Trainingsdaten, um ihre Modellparameter der Domäne anpassen zu können. Im Rahmen dieser Arbeit werden Untersuchungen mit zwei Trainingskorpora aus dem Bereich eines natürlichsprachigen Flugauskunftssystems verwendet: der Korpus der „Natürlichsprachigen Dialogführung im Auto“ (NADIA) für Untersuchungen in der semantischen Decodierung und der „Air Travel Information System“ (ATIS)-Korpus für die semantische Decodierung und das Dialogmanagement. Diese werden in Abschnitt 3.1 eingeführt. Die Systeme der semantischen Decodierung werden sowohl mit dem deutschsprachigen NADIA, als auch mit dem englischsprachigen ATIS-Korpus evaluiert. Somit können linguistische Modellierungsfehler in der jeweiligen Sprache vermieden werden.

In Abschnitt 3.2 werden die semantischen Annotierungen für die verwendeten Trainingskorpora eingeführt. Damit können die Parameter zur Decodierung bedeutungstragender Wortphrasen mithilfe stochastischer Modelle (siehe Kapitel 4) angepasst werden.

Durch ein annotiertes Benutzerziel (*user goal*) (siehe Abschnitt 3.2.4) und der semantischen Annotierungen (*semantic slots*) kann ein Dialogverlauf modelliert werden (siehe Kapitel 5). Dazu ist ein idealer Dialogkorpurs mit annotierten Benutzerzielen erforderlich (siehe Abschnitt 3.1, idealer Korpus). Um dies zu erreichen, sind einige Änderungen am ATIS-Sprachkorpus notwendig, welche in Abschnitt 3.3 beschrieben werden.

Die nachfolgend präsentierten Methoden wurden bereits in der eigenen Veröffentlichung [SRWR09] eingesetzt und etabliert.

3.1 Korpora

Für den Forschungsbereich des Sprachverstehens und Dialogmanagements existieren meist nur wenig aussagekräftige Korpora mit einer brauchbaren Menge an Annotationen. Deshalb müssen meist entsprechende Adaptionen an existierenden

Sprachkorpora durchgeführt werden. In der Regel werden die Daten für den Korpus mit Wizard-of-Oz (WOZ)[1]-Experimenten gesammelt. Mit dieser Methode ist die Dialoglänge $N(d_s)$ variabler, als bei mithilfe von Regeln generierten statistischen Korpora. Diese Aufzeichnungen werden in domänenspezifischen Korpora gesammelt, wie z. B. in [HGD90, Lie03]. Dazu existieren im Falle der Spracherkennung eine Sammlung von Beispielaufzeichnungen, welche natürlichsprachliche Dialoge wiedergeben. Diese Daten sind mit einer Transkriptionssoftware [BGWL01] manuell annotiert, indem die gesprochenen Wörter mit verfügbaren „Tags“ in einem maschinenlesbaren Format gespeichert werden. Die Korpora werden von Experten annotiert, wodurch in der Regel Fehlannotationen gering gehalten werden.

Beispielhaft dazu ist eine ideale Annotation für ein Dialogmanagementsystem in Tabelle 3.1 abgebildet. Neben der Intention (*user goal*) (siehe Abschnitt 3.2.4) ist die Veränderung der semantischen Information (*semantic slot*) im zeitlichen Dialogverlauf Δt von Interesse, woraus der Einfluss von Systemantworten auf den Benutzer gemessen wird. Daraus kann mithilfe von einer Belohnungsfunktion (*reward function*) im Dialogmanager (*user goal estimator*) die am besten passende Intention (*user goal*) angestrebt werden.

3.1.1 Natürlichsprachige Dialogführung im Auto (NADIA)

Die Daten zur Evaluierung der deutschsprachigen Experimente zur semantischen Decodierung in Kapitel 4 stammen auch einer Kollektion und Annotation eines WOZ Experiments. Die Aufzeichnungen des Referenzsystems beinhalten natürlich geführte Dialoge in deutscher Sprache einer Flugauskunftsdomäne im Automobil (NADIA). In diesem Szenario soll der Benutzer auf dem Weg zum Flughafen die nötigen Informationen über den Abflug und die Ankunft (Fluglinie, Flugnummer, Terminal, Gateway, Uhrzeit, etc.) und mögliche Parkinformationen erhalten. Insgesamt werden $N(d) = 495$ Dialoge mit $N(u) = 2\,688$ Benutzeräußerungen und $N(w) = 14\,671$ Wörtern geführt. Somit enthält jeder Dialog im Durchschnitt $\varnothing(u) = 5,4$ Äußerungen, diese bestehen im Durchschnitt aus $\varnothing(w) = 5,5$ Wörtern (siehe Tabelle 3.2).

Die Annotation für die Sprachinterpretation in den gesprochenen Dialogen erfolgt hierarchisch nach [Tho06]. Somit werden insgesamt $N(\mathcal{A}) = 53$ semantische Kategorien extrahiert, davon $N(\mathcal{L}) = 12$ auf der „Wortklassenebene“. Die restlichen $N(\mathcal{C}) = 41$ Kategorien repräsentieren semantische Konzepte, angeordnet in zwei hierarchischen Annotationsebenen. Die prinzipielle Unterscheidung zwischen der Annotation auf Wortklassenebene (*low-level-annotation*) l und der semantischer Konzepte c ist, dass eine Wortklasse eine Menge an einzelnen semantisch gleichbedeutenden Wörtern beschreibt, während die semantischen Annotationsebenen

[1] In der Mensch-Maschine-Kommunikation werden Experimente zur Kommunikation eines Menschen (Proband) mit einem zu entwickelnden System, welches zu Versuchszwecken von einem Mensch im Verborgenen bedient wird, Wizard-Of-Oz Experimente genannt.

Speaker	Dialog and Actions
HUMAN	**I would like to book a cheap flight to New York.**
INTENTION	Dest.City, Price information
DECODED INFORMATION	Dest.City, Price information
GOAL	Dest.City, Orig.City, Price important, Trip type
SYSTEM ANSWER	**Where do you want to start?**
ANWER TYPE	Orig.City
HUMAN	**In Los Angeles.**
INTENTION	Orig.City
DECODED	Orig.City
GOAL	Dest.City, Orig.City, Price important, Trip type
SYSTEM ANSWER	**Do you wish a one way or a round trip?**
ANWER TYPE	Trip type
HUMAN	**One way, please.**
INTENTION	Trip type
DECODED	Trip type
GOAL	Dest.City, Orig.City, Price important, Trip type
SYSTEM ANSWER	**The cheapest one way flights from LA to NY are,**
SYSTEM	*List flights from selected SQL-Query*

Tabelle 3.1: Beispiel eines ideal annotierten Dialogablaufs.

a aus verschiedenen Untereinheiten bestehen. Diese können entweder terminale Einheiten (Wortklassenannotation) oder nicht-terminale Einheiten (semantische Annotation oder „Konzepte“) beinhalten. Annotationen auf Wortebene sind Flughäfen (Orte), Präpositionen des Ortes, Fluggesellschaft (Abkürzung) und Ziffern (entweder für Uhrzeiten oder Flugnummern). Dagegen beschreiben Konzepte, dass z. B. der Flugcode aus der Fluggesellschaft und vier Ziffern bestehen muss.

Mithilfe der Annotierungen der $N(u_{\text{lab}}) = 1\,365$ Benutzeräußerungen steht einerseits ein hierarchisch annotiertes Referenzsystem (siehe Tabelle 3.3) zur Verfügung, andererseits werden alle l zur Bestimmung der Parameter im Training in Kapitel 4 verwendet.

3.1.2 Air Traveling Information System (ATIS)

Das ATIS [HGD90] beschreibt in WOZ-Versuchen eine Flugauskunftsdomäne in englischer Sprache mit $N(u) = 21\,650$ Benutzeräußerungen in $N(d) = 1\,585$ natürlich gesprochenen Dialogen. Dieser Sprachkorpus ist weltweit sehr verbreitet und eignet

	Laborversuche	Versuche im Automobil	Gesamt
#Veruche v	15	17	32
#Dialoge d	240	255	495
#Benutzeräußerungen u	1 365	1 323	2 688
#Wörter w	7 047	7 624	14 671

Tabelle 3.2: Referenzsystem „Natürlichsprachige Dialogführung im Automobil“, welches durch WOZ-Experimente erstellt wurde.

	Training	Eval	Xval	Gesamt
#Wortklassen l	3 689	1 036	1 019	5 744
#Konzepte c	3 904	1 018	1 098	6 020

Tabelle 3.3: Anzahl der Wortklassen und semantische Konzepte, aufgeteilt in vollständig disjunkte Versuchsreihen.

sich deshalb gut für Versuche und Evaluierungen in der semantischen Decodierung sowie für das Dialogmanagement. Dazu sind allerdings einige Anpassungen notwendig, welche für die semantische Decodierung in Abschnitt 3.2 und für das Dialogmanagement in Abschnitt 3.3 beschrieben werden. Die ATIS `*.log`-Dateien enthalten im Wesentlichen nur den Dialogablauf, welcher prototypisch in Tabelle 3.4 dargestellt ist.

3.2 Semantische Slots und Dialogziele

In diesem Abschnitt werden die für eine vollständige Beschreibung eines Dialoges notwendigen Dialogziele (*user goals*) und semantischen Rahmen (*semantic slots*) eingeführt (vgl. Tabelle 3.4 mit Tabelle 3.1). Diese Einführung legt die Grundlagen für die in Kapitel 4 verfolgten Ziele, nämlich aus dem Spracherkenner (siehe Kapitel 2) sogenannte „semantische Slots“ zu extrahieren.

Ein Annotator weist den Benutzer- und Sytemantworten bestimmte semantische Rahmen (*semantic slots*) zu (siehe Abbildung 3.1). Für die Weiterverarbeitung in einem Dialogsystem beinhalten diese Rahmen auch Konfidenzmaße, wie sie in Kapitel 5 zu sehen sind.

Sprecher	Dialoge bzw. Aktionen
HUMAN	Hello.
SYSTEM	What can I do for you?
HUMAN	Show me all the nonstop flights from Atlanta to Philadelphia.
SYSTEM	*List flights from cities* *whose city name is Atlanta and to cities* *whose city name is Philadelphia and* *whose stops is 0*
HUMAN	Yes, I would like some information on the flights on April 22nd, evening
SYSTEM	*List flights from cities* *whose city name is Atlanta and to cities* *whose city name is Philadelphia and whose departure* *time is between 1645 and 1715 and* *flying on April 22nd*

Tabelle 3.4: Beschreibung eines Dialogablaufs zwischen Mensch und Maschine aus dem „Air Traveling Information System“.

3.2.1 Semantische Annotierung eines Flugauskunftssystems

Der im Rahmen dieser Arbeit verwendete ATIS-Korpus [HGD90] verfügt über keine semantische Annotation. Die `*.log`–Dateien beinhalten lediglich die Transkribierung der gesprochenen Äußerungen, wie z. B. „Could you list the cheapest flights from Phoenix to Washington on next Monday?“

Verwendete Annotierung (Labeling)

Um eine zweistufige hierarchische Bedeutungszuweisung vornehmen zu können, werden jedem Wort w zwei verschiedene Labels zugeteilt: Jedes Wort erhält genau ein Konzept (*concept*) c und genau eine Wortklasse (*label*) l. Die Wortklasse kann als Untergruppe des Konzepts betrachtet werden. Innerhalb jedes Konzepts sind jeweils nur einige wenige bestimmte Wortgruppen möglich. In Tabelle 3.5 sind alle verwendeten Konzepte aufgelistet. Zusätzlich finden sich in der Tabelle die verwendete Abkürzung, eine kurze Beschreibung jedes Konzepts und die Anzahl der Vorkommnisse in den verwendeten Daten, also die Häufigkeit, die angibt, wie viele von den 9 185 Wörtern diesem Konzept zuzuschreiben sind.

Es gibt elf Konzepte, die relevante Information darstellen sowie das „Dummy“-Konzept (siehe Tabelle 3.5). Wörter, die für die Datenbankabfrage nicht benötigt werden und somit irrelevant sind, werden dem Konzept „Dummy“ zugeteilt.

In Tabelle 3.6 sind die verwendeten Wortklassen, mit deren Abkürzung und eine

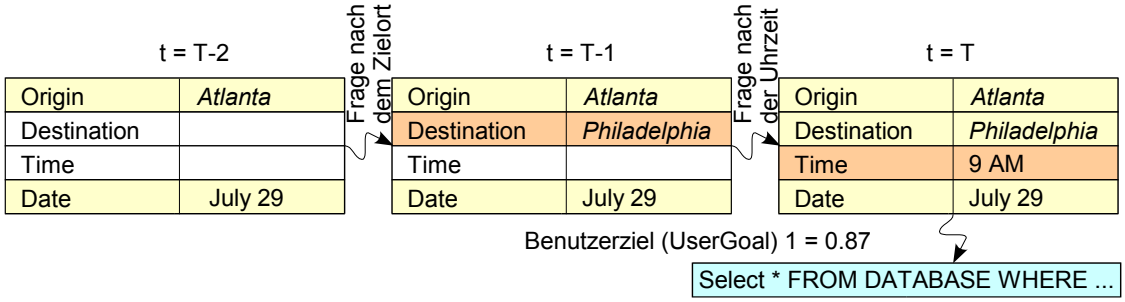


Abbildung 3.1: Zu jedem Zeitschritt t werden die semantischen Rahmen mit Bezeichnungs-Wertpaaren (*slot value pairs*) befüllt. Die Reihenfolge der Befüllung wird mithilfe eines stochastischen Modells ermittelt, dessen Parameter beispielsweise mit dem ATIS-Korpus [HGD90] trainiert werden. Das Ziel des Dialogmanagements ist es, dem Benutzer seinen Intentionen (*user goal*) entsprechende Fragen zu stellen, um durch möglichst wenige Dialogschritte eine SQL-Datenbankabfrage durchführen zu können. Im konkreten Fall wird der Benutzer entsprechend des trainierten Modells zuerst nach dem Ziel und anschließend nach der Uhrzeit befragt.

Konzept	Abkürzung	Beschreibung	Häufigkeit
Origin	OR	Startort eines Fluges	769
Destination	DE	Zielort eines Fluges	851
Request	RE	Anfrage auf die Datenbank	359
Book	BO	Buchungswunsch	150
Time/Date	TD	Zeit oder Datum	366
Additional	AD	Zusätzliche Information	684
Flight Number	FN	Flugnummer	683
Price	PR	Preis	323
Class/Fare Code	CC	Class Code/Fare Code	376
Flight Code	CF	Flight Code	49
Airline	AL	Name einer Fluggesellschaft	291
Dummy	DU	Nicht relevante Information	4 284

Tabelle 3.5: Auflistung aller verwendeten Konzepte c in der Flugauskunftsdomäne ATIS.

kurze Beschreibung (Beispiel) sowie die Häufigkeit gegeben.

Jedes bedeutungstragende Wort wird einer der $N(l) = 25$ definierten Wortklassen

annotiert, irrelevante Wörter werden der Wortklasse „Dummy“ zugeordnet. Die Einteilung in Konzept und Wortklasse (siehe Abbildung 3.2) erfolgt hierarchisch.

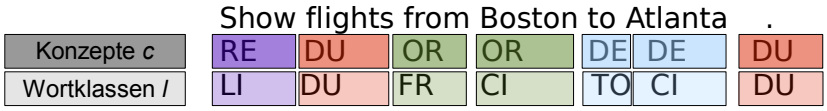


Abbildung 3.2: Beispielsatz mit einer Zuordnung von Konzept c und Wortklasse l zu den Wörtern w.

Im Rahmen dieser Arbeit erfolgt die Zuordnung der Konzepte c und Wortklassen l mithilfe von „Keyword-spotting“ Methoden, unter Zuhilfenahme von Parserwerkzeugen [LS, Joh].

3.2.2 Extrahierung semantischer Slots eines Flugauskunftssystems

Mithilfe des eingeführten „Labelings“ werden die semantischen Slots für ein Dialogmanagementsystem konkretisiert, d. h. ein semantischer Slot ist ein Rahmen, welcher dann mit einem Konzept c befüllt wird.

Das Rahmenmodell (*semantic slots*) wurde hier für $N(c) = 11$ Konzepte entwickelt. Daraus ergeben sich insgesamt $N(T) = 2^{10} = 1\,024$ mögliche Transitionen des Dialogmanagementsystems. Diese Anzahl konnte vom Referenzsystem „teilweise beobachtbaren markoffschen Entscheidungsprozess“ (POMDP) aufgrund der (Zeit-)Komplexität in der Berechnung der Dialogvorschrift nicht verarbeitet werden und musste deshalb auf acht semantische Slots mit insgesamt $N(T) = 2^8 = 256$ Transitionen reduziert werden (siehe Kapitel 5). Daraus ergeben sich insgesamt $N(T) = 2^8 = 256$ mögliche Transitionen des Dialogmanagementsystems (siehe Abbildung 3.3).

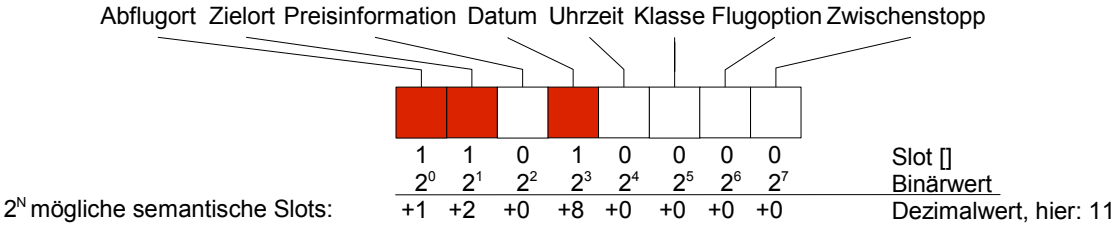


Abbildung 3.3: Extrahierung von semantischen Slots aus dem ATIS Korpus. Hier liegen semantische Werte des Abflughafens, Zielflughafens und des Datums vor. Die Darstellung erfolgt in binärer Schreibweise mit dem niedrigsten Bit an erster Stelle (LSB); die semantische Konfiguration wird in s gespeichert, hier ist $s = 8$.

Jede Kombination semantischen Wissens, repräsentiert durch eine Anordnung von semantischen Slots in einem Vektor $\mathbf{d}_s$, wird als **Dialogzustand** s_i betrachtet. Die dezimale Zahl i berechnet sich aus der Summe

$$i = v(\text{slot 1}) \cdot 2^0 + v(\text{slot 2}) \cdot 2^2 + \cdots + v(\text{slot 8}) \cdot 2^8, \tag{3.1}$$

mit $v(\text{slot } j) \in \{0; 1\}$ aus der binären Zahl des semantischen Slots j. Daher ergeben sich aus s semantischen Slots $N(T) = 2^s$ mögliche Dialogzustände. In Abbildung 3.3 werden die semantischen Informationen binär in $\mathbf{d}_s$ angeordnet.

3.2.3 Beobachtung $\mathcal{O}$

Im Rahmen dieser Arbeit werden mögliche Beobachtungen o semantischer Slots, resultierend aus dem Spracherkennungsprozess und der semantischen Decodierung (siehe Kapitel 4), in gleicher Weise betrachtet, wie die Dialogzustände s. Beobachtungen o_t werden kumulativ in jedem Zeitschritt t zu einem Beobachtungszustand $o_T = \sum_t^T o(\text{slot x})\forall x \in \mathcal{O}$ addiert. Die Menge möglicher Beobachtungen $\mathcal{O}$ ist also deckungsgleich mit jener der Dialogzustände $\mathcal{S}$.

3.2.4 Definition möglicher Benutzerziele $\mathcal{G}$

Ein Benutzerziel (*user goal*) $g_t \in \mathcal{G} \subseteq \mathcal{S}$ ist ein Zustand s , der dem System eine Structured Query Language (SQL)-Datenbankabfrage ermöglicht. Benutzerziele werden auch als Teilziele bezeichnet; diese können entweder durch Annotation im Trainingskorpus oder durch logische Regeln, wie z. B. $(\text{Abflughafen} \wedge \text{Zielflughafen} \wedge \neg\text{Preis})$, definiert sein.

Im Rahmen dieser Arbeit werden die Benutzerziele durch logische Regeln beschrieben. In Abbildung 3.4 ist die Codierung mehrerer Benutzerziele g dargestellt. Mithilfe von Belohnungsfunktionen (*reward functions*) wird der Dialog vom Dialogmanager (*user goal estimator*) zu einem dieser Zwischenziele g_t gelenkt.

3.3 Korpusbeschreibung für Dialogsysteme

Da ein idealer Dialogkorpus (siehe Tabelle 3.1) im Rahmen dieser Arbeit nicht zur Verfügung stand, mussten anhand des Dialogablaufes im ATIS (siehe Abschnitt 3.1.2) Transitionen abgeleitet werden. Aus dem ATIS-Sprachkorpus kann die Intention (*user goal*) des Benutzers nur über die `*.log`–Dateien entschlüsselt werden (siehe Tabelle 3.4). Dazu werden im Rahmen dieser Arbeit die $N(u) = 21\,650$ natürlich gesprochenen Benutzeräußerungen untersucht und die Antworten mithilfe von Schlüsselwortsuche (*keyword searching*) und annotierter SQL-Datenbankabfragen extrahiert und entsprechende Benutzerziele (*user goals*) festgelegt (siehe Abbildung 3.1). Die Transitonen $t_{i\,j}$ der Transitionsmatrix $\mathbf{T}$ des Dialogmodells werden aus $1\,585$ Dialogen gelernt.

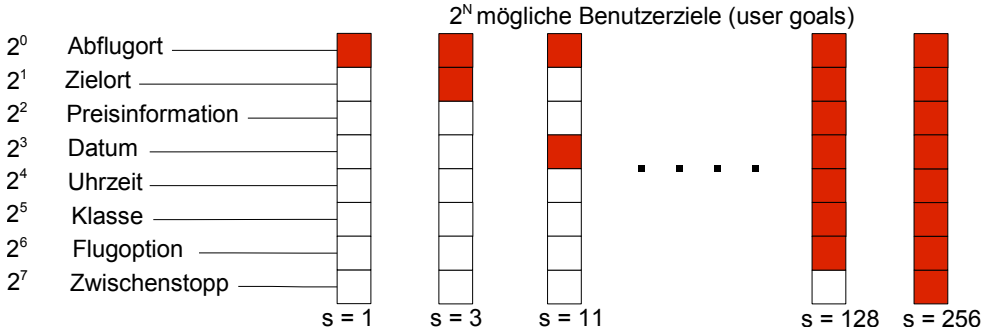


Abbildung 3.4: Schematische Darstellung der semantischen Benutzerziele (user goals): Sowohl der Dialogzustand $s_t \in \mathcal{S}$, als auch die Beobachtung $\mathcal{O}$ der semantischen Information werden in binärer Form (LSB) codiert. Jede Codierung ergibt einen Dialogzustand, insgesamt sind 2^s Dialogschritte d_s möglich.

Im Rahmen dieser Arbeit wurde aus dem ATIS-Sprachkorpus durch Vorverarbeitungsstufen und Annahmen ein Dialogkorpus für die nachfolgende Dialogmanagementsysteme geschaffen (siehe Abschnitt 3.2). Daraus resultieren drei Matrizen:

- Transitionsmatrix $\mathbf{T} = [t_{ij}]_{c \times c}$ mit $t_{ij} = P(c_j|c_i)$, aus dem ATIS-Korpus extrahiert,
- Beobachtungsmatrix $\mathbf{O} = [o_{ij}]_{c \times c}$ mit $o_{ij} = P(c_j|c_i)$, mittels Eingaben simuliert und
- Belohnungsmatrix $\mathbf{R}$, wird durch ein Regelwerk bestimmt.

Die semantischen Paare (siehe Abschnitt 3.2) beschreiben die Transitionsmatrix $\mathbf{T}$, wobei der Eintrag t_{ij} den Übergang von c_i nach c_j darstellt. Die Wahrscheinlichkeiten $P(c_1)$ und $P(c_i|c_{i-1})$, werden aus dem ATIS-Korpus geschätzt. Aufgrund der bereits erwähnten Einschränkungen sind bestimmte Kombinationen c_{i-1}, c_i nicht im Trainingskorpus vorhanden, d. h. $t_{i-1\,i} = 0$. Um dennoch eine Bigrammstatistik für diese semantischen Paare zu erhalten, wird das stochastische Modell in dieser Arbeit mit dem sogenannten Absolute Discounting (ADC) nach [Kat87] verbessert.

3.3.1 Absolute Discounting

Dazu wird die absolute Anzahl $t_{i-1\,i}$ der extrahierten semantischen Informationspaare um einen festen Wert β verringert, wodurch die Randwahrscheinlichkeit verringert wird, d. h. $\sum_{c_i \in \mathbf{T}} P(c_i|c_{i-1}) < 1$. Die resultierende Differenz $1 - \sum_{c_i \in \mathbf{T}} P(c_i|c_{i-1})$ kann gemäß $P(c_i)$ auf die **nicht** im Trainingskorpus vorkommenden, verbleibenden Informationspaare aufgeteilt werden. Somit erhält man eine Statistik für nicht beobachtbare semantische Informationsübergänge.

In Abbildung 3.5 ist der Effekt des „Absolute Discounting“ zusammen mit dem „Backing-Off“ graphisch dargestellt.

3.3.2 Backing-Off

Zusätzlich wird ein „backing-off“ Faktor γ verwendet, der die minimale Anzahl $t(c_i, c_{i-i})$ für die Berechnung der semantischen Paare (c_i, c_{i-i}) nach [Kat87] festlegt. Man erhält so

$$P(c_i|c_{i-1}) = \begin{cases} \frac{t(c_i,c_{i-1})-\beta}{t(c_{i-1})} & \text{wenn } t(c_i, c_{i-1}) > \gamma \\ b(c_i, c_{i-1}) \cdot P(c_i) & \text{sonst,} \end{cases} \tag{3.2}$$

mit $b(c_i, c_{i-1}) = \frac{1-\sum_{c_i \in \mathcal{B}(c_{i-1})} P(c_i|c_{i-1})}{1-\sum_{c_i \in \mathcal{B}(c_{i-1})} P(c_i)}$ und $\mathcal{B}(c_{i-1}) = \{c_i|c_i, c_{i-1} \in \mathcal{C} \wedge t_{c_i,c_{i-1}} > \gamma\}$. Die um β reduzierte Wahrscheinlichkeit ist in Abbildung 3.5 graphisch dargestellt.

3.3.3 Regelwerk

Für den Fall, dass ein Zustand c_i nicht im Trainingskorpus existiert, hat sich aus Experimenten die Erstellung einer Prioritätenliste bewährt (siehe Tabelle 3.7). Hier werden die semantischen Informationen „Abflughafen“, „Zielflughafen“, „Uhrzeit“ auf $P(c_i) = \frac{\text{prio}(c_i)}{\sum_{c_i} \text{prio}(c_i)} = 0,21$, „Datum“, „Flugoption“ auf $P(c_i) = 0,11$ und alle anderen auf $P(c_i) = 0,053$ gesetzt.

Aus empirischen Untersuchungen hat sich eine Fusion der regelbasiert generierten Daten mit den annotierten Trainingsdaten im Verhältnis 1 zu 9 etabliert. Daraus wird der Zustandsübergangsvektor $\mathbf{t}(c_{i-1}, c_i)$ bestimmt (siehe Algorithmus 1). Mithilfe dieser Kombination hat zum einem das Regelwerk eine sehr geringe Bedeutung, zum anderen ist die gesamte Übergangswahrscheinlichkeit $\mathbf{t} > 0$.

Algorithmus 1 Mischung der Modellparameter aus der Stochastik und einem Regelwerk.

for $k \in \{1; \ldots; 2^s\}$ **do**
$\mathbf{t}_k = 0.9 \cdot \mathbf{t}_k^{\text{stat}} + 0.1 \cdot \mathbf{t}_k^{\text{prio}}$ ▷ $\mathbf{t}_k$ wird zu $^9/_{10}$ aus dem statistischen Traininigskorpus, aber auch zu $^1/_{10}$ aus dem definierten Regelwerk bestimmt.
end for

3.4 Zusammenfassung des Kapitels

Dieses Kapitel befasste sich mit den verfügbaren Korpora für stochastische Modelle im Bereich der semantischen Decodierung als auch für das Dialogmanagement.

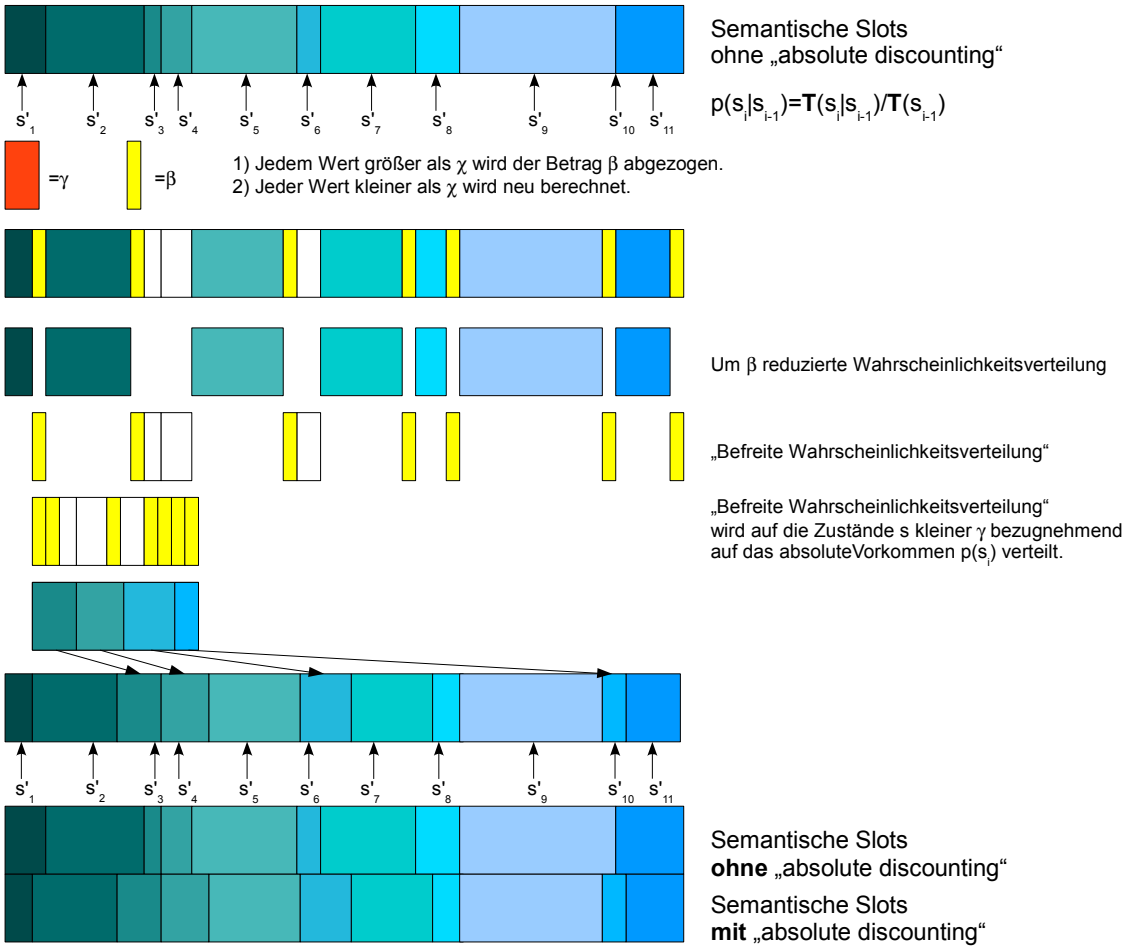


Abbildung 3.5: Graphische Darstellung der Auswirkung von „Absolute discounting“. Hier erhält der Zustand s_{10} eine Wahrscheinlichkeitsverteilung $P(s_{10}) > 0$, durch eine Reduzierung der Wahrscheinlichkeitsverteilungen $\forall s_i$: wenn $P(s_i) > \gamma P(s_i) = P(s_i) - \beta$.

In Abschnitt 3.1 wurden die in dieser Arbeit verwendeten Korpora NADIA und ATIS vorgestellt. Dabei war eine Definition von semantischen Slots und Dialogzielen (siehe Abschnitt 3.2) für eine vollständige Modellierung notwendig. In Abschnitt 3.3 erfolgte eine Beschreibung des verwendeten Korpus für ein Dialogmanagementsystem, mit verwendetem „Absolute Discounting“, „Backing-Off“ und Regelwerk.

Wortklasse	Abkürzung	Beschreibung/Beispiel	Häufigkeit
FromLoc	FR	„from“, „leaving“	360
ToLoc	TO	„to“, „arriving in“	406
City	CI	Name einer Stadt	778
Airline	AL	Name einer Fluggesellschaft	283
Indicator Flight Number	IN	„flight number“	224
Indicator Price	IP	Zeigt einen Preis an	164
Indicator Class Code	IC	Zeigt einen Class Code an	350
Indicator Flight Code	IF	Zeigt einen Flight Code an	12
Indicator Time	IT	„p.m.“, „o'clock“, „date“	70
Number Word 1	NU	oh, zero, one – nine	453
Number Word 2	NW2	ten - ninety_nine	185
Number 10	N10	$0-9$	20
Number 60	N60	$00-59$	33
Number Rest	NR	„hundred“, „600.000“	23
Year	YE	Jahreszahl, „1990“	10
Month	MO	Monatsname	39
Date	DA	„4^{th}“, „23^{rd}“	37
Weekday	WD	Wochentag	13
Daytime	DT	Tageszeit, „morning“, „noon“	40
List	LI	Wunsch nach Daten	261
Define/Describe	DE	Wunsch nach Erklärung	92
Sort	SO	Wunsch nach Sortierung	4
Book	BO	Buchungswunsch	112
Adjective/Additional	AD	Adjektiv, zusätzliche Information	820
Flight Information	NS	„nonstop“, „round-trip“	114
Dummy	DU	Nicht relevante Information	4 284

Tabelle 3.6: Auflistung aller verwendeten Wortklassen l in der Flugauskunftsdomäne ATIS.

prio[0]	4	Zielflughafen
prio[1]	4	Abflughafen
prio[2]	1	Preis
prio[3]	2	Datum
prio[4]	4	Uhrzeit
prio[5]	2	Flugoption
prio[6]	1	Menüauswahl
prio[7]	2	Zwischenstopp

Tabelle 3.7: Prioriätenreihung der semantischen Slots in der Flugauskunftsdomäne.

4

Semantische Decodierung

Dieses Kapitel behandelt die semantische Decodierung von natürlich gesprochener Sprache. Die Bearbeitung der semantischen Ebene wird als **Sprachinterpretation** bezeichnet, die feststellt, was der Sprecher gemeint hat. Sie ist das Schlüsselelement in der Verarbeitungskette eines natürlich sprachlich geführten Dialogsystems. Um ein sprachverstehendes System realisieren zu können, muss die zu bearbeitende Domäne entsprechend eingegrenzt werden. Bei der gewählten Domäne handelt es sich in dieser Arbeit um ein Flugauskunftsystem, welches sowohl in englischer, als auch in deutscher Sprache untersucht wurde. In Kapitel 2 wurde die natürlich gesprochene Sprache und deren Verarbeitung eingeführt. Natürlich gesprochene Sätze werden im Gegensatz zu den **Formalen Sprachen** der theoretischen Informatik als „schwach kausal[1]“ bezeichnet. Der Satzteil „den Kopf abschneiden“ hat z. B. eine ganz andere Bedeutung als „den Zopf abschneiden“, obwohl sich im Satzteil nur eine Silbe verändert hat. Ebenso kann bei **ambigen** Wörtern, wie z. B. „die Bank“, deren Bedeutung (die Bank im Park oder ein Kreditinstitut) nur durch eine Interpretation des Gesprochenen erkannt werden (siehe Abbildung 4.1). Diese vermeintliche Schwäche ist durch die kompakte Codierung der natürlich gesprochenen Sprache zu erklären [Rec94].

Mit der **semantischen Decodierung**[2] ist demzufolge das Verstehen eines gesprochenen Wortes innerhalb eines Satzes gemeint. Die semantische Decodierung nimmt die Ergebnisse eines Spracherkenners auf und interpretiert diese aufgrund eines stochastischen oder grammatikalischen Modells. In den Arbeiten von Young [You02, HY03, HY05, YY06] und Acero, Wang [WA05, AW04, WDA05] werden vor allem stochastische Verfahren zur semantischen Decodierung verwendet.

Ein erster Ansatz zur semantischen Decodierung von gesprochener Sprache ist die flache Decodierung. Dabei werden mithilfe von Transduktoren, welche hierarchisch angeordnet sind, semantische Konzepte beschrieben. Gerade in der deutschen

[1] Schwach kausal bedeutet, dass eine kleine Änderung im Eingangssignal („Ursache“) eine starke Änderung im Ausgangssignal („in der Wirkung“) haben kann [Rec94].

[2] Die (De-)Codierung ist definiert, als eine allgemein zugängliche Transformation [Sch03].

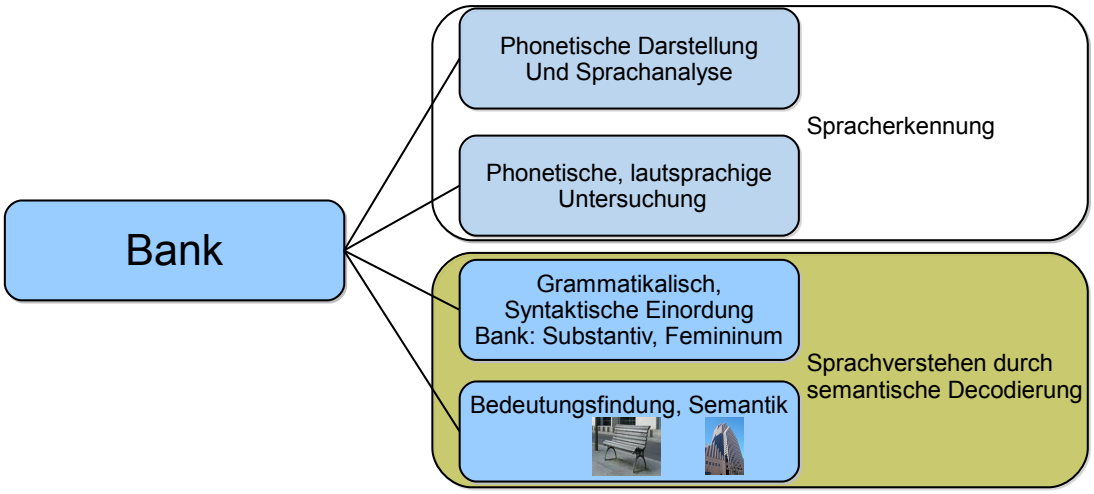


Abbildung 4.1: Verschiedene Betrachtungsarten der Sprache.

Sprache ist die Modellierung mit Transduktoren durch die Existenz mehrteiliger Prädikate problematisch.

Die nachfolgend präsentierten semantischen Decodierverfahren wurden bereits in vorausgegangenen eigenen Veröffentlichungen eingesetzt und etabliert [SSWR08, SGS+08]. Außerdem wurde das in [HY05] präsentierte System implementiert, und dessen Schwierigkeiten wurden aufgezeigt. Dieses System spiegelt den Stand der Technik (*state-of-the-art*) in der semantischen Decodierung wieder.

4.1 Verfahren zur semantischen Decodierung

Dieser Abschnitt befasst sich mit stochastischen Verfahren zur semantischen Decodierung von Wörtern in gesprochenen Äußerungen. Anschließend werden ab Abschnitt 4.3 regelbasierte und stochastische Modelle zur semantischen Decodierung erläutert. Die Funktionaliät der Modelle wird mit Experimenten (siehe Abschnitt 4.3.4) belegt und deren Ergebnisse mit den hier beschriebenen Stand-der-Technik-Verfahren verglichen.

4.1.1 Flache Decodierung

Im Folgenden wird das Prinzip der stochastischen semantischen Decodierung mit GM eingeführt. Allgemein kann das „Parsen“ einer Sequenz von Konzepten[3] $\mathbf{c} =$

[3] Allgemein beschreibt der Begriff „Konzept“ einen scharf abgegrenzten Raum. Im Bereich des Sprachverstehens wird mit einem semantischen Konzept die Bedeutung von Satzteilen beschrieben [Qui67]. Ein Konzept besteht aus einer oder mehreren Wortklassen (*Labels*).

$c_1, c_2, \ldots, c_n$ mit der Rückschlusswahrscheinlichkeit $P(\mathbf{c}|\mathbf{w})$ über einer Wortfolge $\mathbf{w} = w_1, \ldots, w_n$ beschrieben werden:

$$\begin{aligned} \hat{\mathbf{c}} &= \underset{\mathbf{c}}{\operatorname{argmax}}\, P(\mathbf{c}|\mathbf{w}) \\ &\approx \underset{\mathbf{c}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{c})\; P(\mathbf{c}) \end{aligned} \tag{4.1}$$

$P(\mathbf{c})$ beschreibt die Wahrscheinlichkeit der Verteilung der semantischen Konzepte über eine Sequenz $\mathbf{c}$. Das lexikalische Modell wird über die Wahrscheinlichkeitsverteilung $P(\mathbf{w}|\mathbf{c})$ einer gesprochenen Wortfolge $\mathbf{w}$ mit gegebener Semantik $\mathbf{c}$ beschrieben (siehe Gleichung 4.1).

In Abbildung 4.2 wird ein „Tagging“ Modell, welches einem HMM entspricht, zur semantischen Decodierung nach Pieraccini dargestellt [PTG+92]. Der **Prolog** modelliert dabei die ersten T_P Zeitpunkte und der **Epilog** die letzten T_E Zeitpunkte einer Beobachtung $w_1, \ldots, w_T$. Das Wiederholen des **Chunks** über $T_C = T - T_P - T_E$ Zeitschlitze wird auch als das Ausrollen eines dynamischen Bayes'schen Netzwerkes (DBN) bezeichnet.

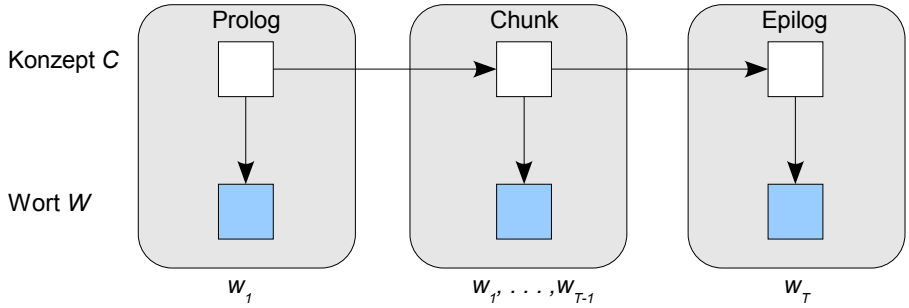


Abbildung 4.2: Flaches Erkennungskonzept realisiert als ein Dynamisches Bayes'sches Netz, welches einem „Tagging“ mit einem „Hidden Markov Modell“ entspricht. Der Zustand der Konzepte $\mathbf{c}$ modelliert den zeitlichen Ablauf der Markovkette. In einem zweiten stochastischen Prozess emittiert der Zustand zu jedem Zeitpunkt ein Wort w_t.

Das DBN aus Abbildung 4.2 repräsentiert mit einem Prolog, Chunk und Epilog ein GM [Bila], welches mit der Verbundwahrscheinlichkeit beschrieben werden kann:

$$P(\mathbf{c}) = \prod_{t=1}^{T} P(c_t|c_{t-1}) \tag{4.2}$$

Aus den verborgenen Zuständen $\mathbf{c}$ werden Wortfolgen $w_t \in \mathcal{W}$ einer sprachlichen Äußerung generiert bzw. absorbiert. Diese werden als lexikalisches Modell bezeichnet, und deren Verbundwahrscheinlichkeit lässt sich wie folgt faktorisieren:

$$P(\mathbf{w}|\mathbf{c}) = \prod_{t=1}^{T} P(w_t|c_t) \tag{4.3}$$

Nachteile:

- Durch die **flache** Decodierung ist es nicht möglich, semantische Abhängigkeiten über längere Distanzen zu „parsen“, wie sie beispielsweise bei mehrstelligen Prädikaten häufig vorkommen.
- Ebenso können regelhafte semantische Beschreibungen sehr schwer in dieses Modell integriert werden.
- Des Weiteren ist mit diesem Verfahren keine Wortklassenebene l bzw. weitere Konzeptebene c vorgesehen.

4.1.2 Semantische Kaskadierung mit Transduktoren

Ein Transduktor $\mathbf{T}$ ist ein in der Ausgabefunktion erweiterter Mealy[4]-Automat. Jelinek [JS04] verwendet die kaskadierten Transduktoren, um aus einer Wortfolge $w_1, \ldots, w_T$ die bestmögliche Wortklasse $\mathbf{l}$, Konzeptklasse $\mathbf{c}$ und Dialogaktion $\mathbf{a}$ zu berechnen:

$$\begin{aligned} \hat{S} = [\hat{\mathbf{l}}, \hat{\mathbf{c}}, \hat{\mathbf{a}}] &= \underset{\mathbf{l},\mathbf{c},\mathbf{a}}{\operatorname{argmax}}\, P(\mathbf{l}, \mathbf{c}, \mathbf{a}|\mathbf{w}) \\ &\approx \underset{\mathbf{l},\mathbf{c},\mathbf{a}}{\operatorname{argmax}}\, P(\mathbf{w}|\mathbf{l})\ P(\mathbf{l}|\mathbf{c})\ P(\mathbf{l}|\mathbf{a})\ P(\mathbf{a}). \end{aligned} \tag{4.4}$$

In Gleichung 4.4 wird mit der FORMEL VON BAYES eine Kaskadierung in die Wahrscheinlichkeiten $P(\mathbf{w}|\mathbf{l})$, $P(\mathbf{l}|\mathbf{c})$, $P(\mathbf{c}|\mathbf{a})$ und $P(\mathbf{a})$ approximiert. Diese beschreiben in [JS04] folgende vier hierarchische Ebenen:

$P(\mathbf{a})$**:** Das Dialogmodell beschreibt die Abhängigkeiten zwischen den aufeinanderfolgenden Dialogschritten.

$P(\mathbf{c}|\mathbf{a})$**:** Das Konzeptmodell beschreibt die situationsbedingte Abhängigkeit der Konzepte von der Aufgabenstellung (Dialogzustand).

$P(\mathbf{l}|\mathbf{c})$**:** Das semantische Wortklassenmodell beschreibt mögliche Wortklassen abhängig vom semantischen Konzept.

$P(\mathbf{w}|\mathbf{l})$**:** Das lexikalische Modell beschreibt die Wahrscheinlichkeit der Worte zur Wortklasse.

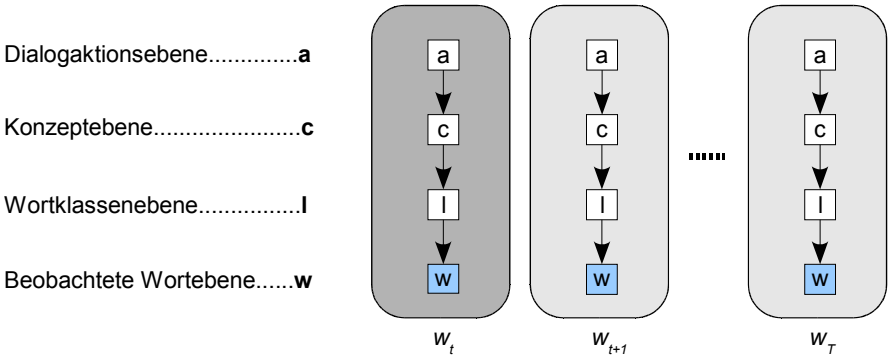


Abbildung 4.3: Einfache semantische Kaskadierung nach [JS04]: In jeder Kaskade weisen gewichtete stochastische Transduktoren $\mathbf{T}$ einem beobachteten Wort w_t einer Folge $\mathbf{w}$ die wahrscheinlichste Wortklasse $l_t \in \mathcal{L}$, eine semantische Kategorie $c_t \in \mathcal{C}$ und eine Aktion $a_t \in \mathcal{A}$ eines Dialogmanagers zu. Aufgrund der limitierten Möglichkeit in der on-line Erkennung werden im Rahmen dieser Arbeit gewichtete Transduktoren zur semantischen Decodierungen nicht weiter verfolgt.

In Abbildung 4.3 wird die vereinfachte Modellierung aus Gleichung 4.4 zur semantischen Decodierung eines Wortes w_t mithilfe von kaskadierten Transduktoren dargestellt.

Vor- und Nachteile:

- Gegenüber der flachen Decodierung hat sich die mehrstufige Kaskadierung ($\mathbf{a}, \mathbf{c}$ und $\mathbf{l}$) bewährt [HY05].
- Definierbare Zustandsübergänge δ in den Transduktoren ermöglichen die Verwendung von Grammatiken.
- Für jedes Wort w_i muss ein semantisches Konzept $c_i \in \mathcal{C}$ und eine passende Dialogaktion $a_i \in \mathcal{A}$ decodiert werden.
- Es ist nach wie vor nicht möglich semantische Abhängigkeiten über längere Distanzen zu „parsen“, wie sie beispielsweise bei mehrstelligen Prädikaten häufig vorkommen.

Deshalb wird im nächsten Abschnitt (siehe Abschnitt 4.2) mit dem Hidden-Vector-State (HVS)-Modell ein Verfahren vorgestellt, welches mittels einer hierarchischen

[4]Ein Mealy-Automat ist ein endlicher Automat, dessen Ausgabe von seinem Zustand und seiner Eingabe abhängt. Anschaulich bedeutet das, dass jeder Kante im Zustandsdiagramm ein Ausgabewert zugeordnet wird [HU00]

Modellierung auch mehrstellige Wörter (Prädikate) semantisch decodieren kann. He – Young [HY05] zeigten, dass dieses Modell aufgrund der hierarchischen Struktur und der semantischen Repräsentation als Baum gegenüber den vorgestellten Transduktoren besser abschneidet.

4.2 Das Hidden-Vector-State Modell

In einem HVS-Modell wird eine Decodierung in Form eines semantischen Baumes realisiert. Der HVS-Parser ist eine Erweiterung der bisher erwähnten flachen „Parser" mit Einbeziehung eines Kellerautomaten, der die bisher verwendeten semantischen Informationen speichert. Für jedes beobachtete Wort w_t wird ein passender Zustandsvektor c_t, welcher den Weg vom Wortklassenknoten l_t bis zum Wurzelknoten ST beschreibt, erstellt (siehe Abbildung 4.4). Diese Form der Decodierung ist gegenüber den Transduktoren (siehe Abschnitt 4.1.2) für verschachtelte Sätze und insbesonders für mehrteilige Prädikate besser geeignet [HY05]. Im Rahmen dieser Arbeit wird das HVS-Modell als „State-of-the-Art"-Modell für die semantische Decodierung mittels regelbasiertem und stochastischem Ansatz verwendet.

In Abbildung 4.4 wird passend zum „Parsetree" der jeweilige Inhalt des Zustandsvektors $\mathbf{c}_t$ zum Zeitpunkt t dargestellt. So sind z. B. im Wort *München* folgende semantische Wortklassen $l_t = c_t[1]$ codiert:

$$\mathbf{c}(\textit{München}) = [\texttt{STADT}, \texttt{NACH}, \texttt{ZURÜCK}, \texttt{TRANSPORT}, \texttt{ST}]$$

Der „Parsetree" $\mathbf{b}$ in Abbildung 4.4 kann auch als ein Markov-Modell 1. Ordnung mit den verborgenen Zuständen des Vektors $\mathbf{c}$ betrachtet werden. Jeder Vektor $\mathbf{c}_t$ des Wortes w_t ist äquivalent zu einem Zustand s_t eines endlichen Kellerautomaten (*push-down automaton*) $\mathcal{M}$, nach folgender Definition:

$$\begin{aligned}
&\mathcal{M} = (\mathcal{S}, \Sigma, \Gamma, \delta, s_0, \#, F) \text{ mit} \\
&\mathcal{S}, \text{ einer endliche Menge an Zuständen,} \\
&\Sigma, \text{ dem Eingabealphabet,} \\
&\Gamma, \text{ dem Kelleralphabet,} \\
&\delta \subseteq (\mathcal{S} \times (\Sigma \cup \{\mathcal{E}\}) \times \Gamma) \rightarrow (\mathcal{S} \times \Gamma^*), \text{ den Zustandsübergängen} \\
&s_0 \in \mathcal{S}, \text{ dem Startzustand} \\
&\# \in \Gamma, \text{ dem Anfangssymbol im Keller und} \\
&F \subseteq \mathcal{S}, \text{ der endlichen Menge an Endzuständen.}
\end{aligned} \tag{4.5}$$

Die Zustandsvektoren $\mathbf{c}_t$ werden in einen Kellerautomaten $\mathcal{M}$ abgebildet, welcher mithilfe von Kellerspeicheroperationen $\texttt{pop}(0), \ldots, \texttt{pop}(I)$ zu einem endlichen Netzwerk ausgerollt werden kann. Dabei beschreibt eine $\texttt{pop}(i)$ Operation das Entfernen von i Wortklassen (*Labels*) auf den Vektor $\mathbf{c}$. Durch eine Begrenzung von z. B. $I = 4$ bleibt das Netzwerk handhabbar. Mit einem $\texttt{push}()$-Befehl wird in jedem

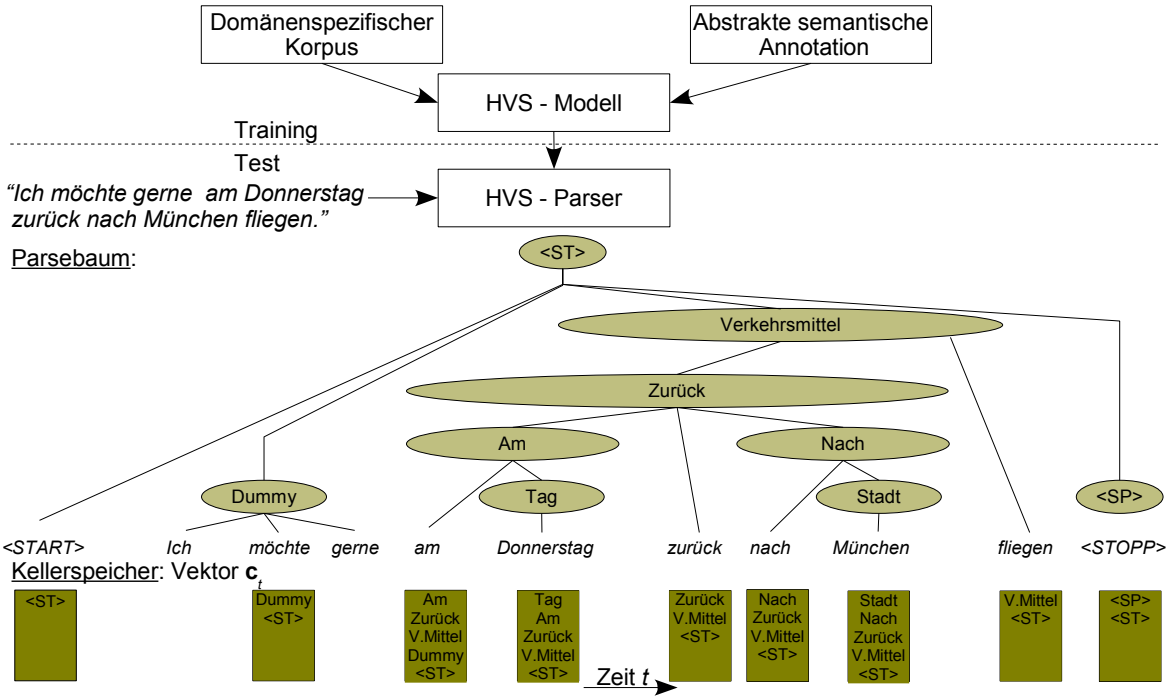


Abbildung 4.4: Beispiel für die semantische Decodierung mittels HVS-Modell [HY05]: Der „Parsetree“ ist semantisch äquivalent zu den Werten der Vektoren $\mathbf{c}_t$. Dieser repräsentiert einen Kellerspeicher, welcher durch Restriktionen (z. B. Rechtslinearität) an `push(i)` bzw. `pop()`-Operation handhabbar bleibt. Der Kellerautomat $\mathcal{M}$ kann zu einem endlichen Automaten (Netzwerk) ausgerollt werden, und deren Transitionen können mithilfe von Beispielsätzen trainiert werden.

Zeitschritt t die aktuelle Wortklasse l_t auf den Kellerspeicher $\mathbf{c}_t$ gelegt. Aus den restriktiven Kellerspeicheroperationen lassen sich rechts- (bzw. links-) verzweigende „natürlichgesprochene Sprachen“ ableiten [Phi95].

Als nachteilig hat sich bei der Transformation des „Parsetrees“ in eine Sequenz $\mathbf{C} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_T)$ die semantisch gleiche Beschreibung an Zustandsvektoren bei ambigen Wörtern erwiesen. Mit einer Folge $N = (n_1, n_2, \ldots, n_t)$ an Kellerspeicheroperationen berechnet sich die Verbundwahrscheinlichkeit

$$P(N, \mathbf{C}, W) = \prod_{t=1}^{T} \underbrace{P(n_t | W_1^{t-1}, \mathbf{C}_1^{t-1})}_{\nu} \underbrace{P(c_t[1] | W_1^{t-1}, \mathbf{C}_1^{t-1}, n_t)}_{\gamma} \underbrace{P(w_t | W_1^{t-1}, \mathbf{C}_1^{t})}_{\omega} \quad (4.6)$$

mit einer Sequenz $\mathbf{c}_t = (c_t[1], c_t[2], \ldots, c_t[D_t])$ an semantischen Konzepten der Kellerspeichertiefe D_t. Das oberste Kellerspeicherelement $c_t[1]$ entspricht der Wortklasse l_t und das unterste Kellerspeicherelement $c_t[D_t]$ beinhalten den Startknoten ST (siehe

Messungen:	FST	HVS
Recall [%]	86,71	89,82
Precision [%]	84,84	88,75
F-Score [%]	85,77	89,28

Tabelle 4.1: Nach [HY05] kann mit dem HVS-Modell gegenüber dem FST-Modell besonders bei längeren Sätzen bessere Erkennungsraten in der ATIS-Domäne erreicht werden.

Abbildung 4.4). Das Produkt $W_1^{t-1}\mathbf{C}_1^{t-1}$ beschreibt die semantische Decodierung zum Zeitpunkt $t-1$.

Das Kellerspeicherelement D_t berechnet sich iterativ aus dem vorigen Kellerspeicherelement $D_{t-1}+1-n_t$. Die Zahl 1 bedeutet, dass immer ein neues Element auf den Kellerspeicher gelegt wird muss, nachdem $n_t \leq I$ Elemente vom Kellerspeicher genommen werden. Nach [HY05] kann das HVS-Modell (siehe Gleichung 4.6) mit folgenden Approximationen

$$\nu = P(n_t|W_1^{t-1}, \mathbf{C}_1^{t-1}) \approx P(n_t|\mathbf{c}_{t-1}) \tag{4.7}$$

$$\gamma = P(c_t[1]|W_1^{t-1}, \mathbf{C}_1^{t-1}) \approx P(c_t[1]|c_t[2\ldots D_t]) \tag{4.8}$$

$$\omega = P(w_t|W_1^{t-1}, \mathbf{C}_1^{t}) \approx P(w_t|\mathbf{C}_t) \tag{4.9}$$

berechnet werden.

4.2.1 Experimente und Ergebnisse

Für die Evaluierung wurde der ATIS-Korpus verwendet (siehe Kapitel 3). Nach [HY05] stellt sich gegenüber dem Transduktoren (*finite-state transducer*) (FST)-Modell eine relative Verbesserung von $4,9\%$ (*F-Score*) ein:

Mithilfe der implementierten Teilformeln (siehe Gleichung 4.7, 4.8 und 4.9) und des Trainingsalgorithmus (siehe Anhang 1.1) konnten im Rahmen dieser Arbeit folgende Grenzen bei der Decodierung der Modelle 1. Ordnung erreicht werden:

- Durch die systembedingte Links-rechts-Decodierung ist eine Decodierung von Datum, Uhrzeit etc. wesentlich komplexer als mit einfachen Grammatikregeln.
- Die Anzahl der Zustände entspricht annähernd einer Vergrößerung um den Faktor 23 (120 Zustände aus dem FST-Modell gegenüber den $2\,799$ Zuständen aus dem HVS-Modell).
- Das System bleibt nur bis zu einer maximalen Kellerspeichertiefe[5] von $D=3$ berechenbar.

[5]Bezeichnet die Anzahl der maximalen Speicherlemente (Objekte) eines Kellerspeichers. Der Kellerspeicher (Stack) arbeitet nach dem Last In – First Out-Prinzip (LIFO).

Das implementierte HVS-Modell wird aufgrund der Problematik bei der Decodierung von grammatikalisch einfach zu beschreibenden Wortgruppen, wie z. B. Datum, Uhrzeit etc. hier nicht weiter verfolgt. Die Ergebnisse aus Tabelle 4.1 dienen als Referenzsystem (*Baseline System*) für die weiteren Eigenentwicklungen im Bereich der semantischen Decodierung.

4.3 Kombinationen stochastischer Verfahren mit kontextfreien Grammatiken

Neben den stochastischen Verfahren (z. B. HVS Modell, siehe Abschnitt 4.2) eignen sich kontextfreie Grammatiken für die semantische Decodierung. In kommerziellen Systemen werden ECFG bei PHOENIX-Parsern eingesetzt — sie erlauben dort das Überspringen von unbekannten Wörtern und das Parsing von Satzteilen [WI94].

4.3.1 Erweiterte kontextfreie Grammatiken

Die im Theorieteil (siehe Kapitel 2) beschriebenen ECFG werden hier modifiziert verwendet. Neben den nicht-terminalen Zeichen N dürfen nun auch Quantoren der RE für die terminalen Zeichenketten Σ verwendet werden. Die Quantoren $?, +, *$ definieren das null- oder einfache, ein- oder mehrfache bzw. null- oder mehrfache Vorkommen terminaler Zeichen Σ.

Im folgenden Beispiel sollen aus natürlich gesprochenen Sätzen des NADIA-Korpus der Abflug- und der Zielort ermittelt werden. Es werden die Präpositionen `von`, `nach` und alle zugehörigen Synonyme mit den Labels `VON` bzw. `NACH` versehen. Die „semantische Grammatik“ wird wie folgt definiert:

$$\begin{aligned} \texttt{S} &= \texttt{DUMMY}^*.(\texttt{AUSGANGSORT} \mid \texttt{ZIELORT}).\texttt{DUMMY}^* \\ \texttt{AUSGANGSORT} &= \texttt{VON.DUMMY}^*.\texttt{STADT} \\ \texttt{ZIELORT} &= \texttt{NACH.DUMMY}^*.\texttt{STADT} \end{aligned} \tag{4.10}$$

Im Rahmen dieser Arbeit werden die manuell erstellten Grammatiken mittels einer hierarchischen Baumdarstellung in Übergangsnetze transformiert und mit herkömmlichen Verfahren (z. B. Viterbi-Algorithmus) decodiert. Eine äquivalente **hierarchische** Baumdarstellung von Gleichung 4.10 ist in Abbildung 4.5 zu sehen.

4.3.2 Hierarchische Übergangsnetzwerke

Aus den semantischen Grammatikregeln (ECFG) wird ein hierarchisches Übergangsnetzwerk **N** aufgespannt. Dieses Netzwerk dient als einheitliche Modellierungsform (*Framework*) von sowohl **grammatikalischen** als auch **stochastischen** Verfahren. So

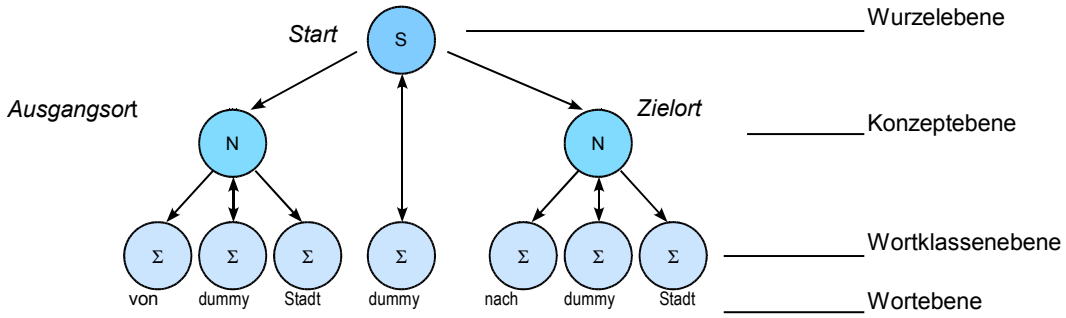


Abbildung 4.5: Transformation von semantischen Grammatiken (siehe Gleichung 4.10) in hierarchische Übergangsnetzwerke: Der „Parsetree“ liefert eine intuitive Darstellung der semantisch zu decodierenden Einheiten (Wort-, Wortklassen-, Konzept- und Wurzelebene). Die Knotenverbindungen sind äquivalent zu den Kanten des neu gebildeten Übergangsnetzwerks [SGS+08].

können die HMM für die akustisch-phonetische Modellierung direkt in die Netzwerkhierarchie integriert werden. Die integrierte Modellierung von akustischen und syntaktischen, natürlich gesprochenen Daten wird in den Arbeiten von Thomae und Lieb beschrieben [Tho06, Lie06].

Bei dem hierarchischen Übergangsnetzwerk (siehe Abbildung 4.6) handelt es sich um ein **Mealy**-Netzwerk. Das bedeutet, dass in den Transitionen die informationstragenden, semantischen Werte hinterlegt sind.

Die Struktur eines regelbasierten hierarchischen Übergangsnetzwerks wird manuell, z. B. mittels ECFG (siehe Abschnitt 4.3), durch einen Experten eingegeben. Im Gegensatz zu stochastischen Übergangsnetzwerken kann bei regelbasierten Systemen mithilfe von wenigen Grammatikregeln eine relativ hohe Abdeckung der Zielsprache erreicht werden. Eine weitere Steigerung des Abdeckungsgrades kann durch Hinzufügen weiterer Grammatikregeln erzielt werden. Damit wird das Übertragungsnetz komplexer und inkonsistenter mit der Konsequenz, dass bei unerwartet gesprochenen Äußerungen ein höherer Erkennungsfehler zu erwarten ist [SGS+08]. In Abbildung 4.7 erfolgt die beispielhafte Übertragung einer erweiterten kontextfreien Grammatik in ein hierarchisches Übergangsnetzwerk. Die Zustände sind ausgehend vom obersten Konzept durchnummeriert. Im Zustand 2 wird ein neues Unterkonzept mit den Zuständen $20-22$ aufgerufen. Erst wenn dieser Zustandsautomat abgearbeitet ist, d. h. der Endzustand wurde erreicht, wird das Unterkonzept verlassen und zum Dachkonzept zurückgekehrt. Die Zustandsübergänge erfolgen mit erkannten Labels, hier sind dies `D` *(dummy)*, `ST` *startzustand*, `E` *(endzustand)* und `FROM` *fromloc*, `TO` *toloc*, `C` *city* für die Unterkonzepte.

Das Übergangsnetzwerk (siehe Abbildung 4.7) wird mittels eines Transitionsalgorithmus in eine große statistisch gewichtete Matrix $\mathbf{T}$ überführt.

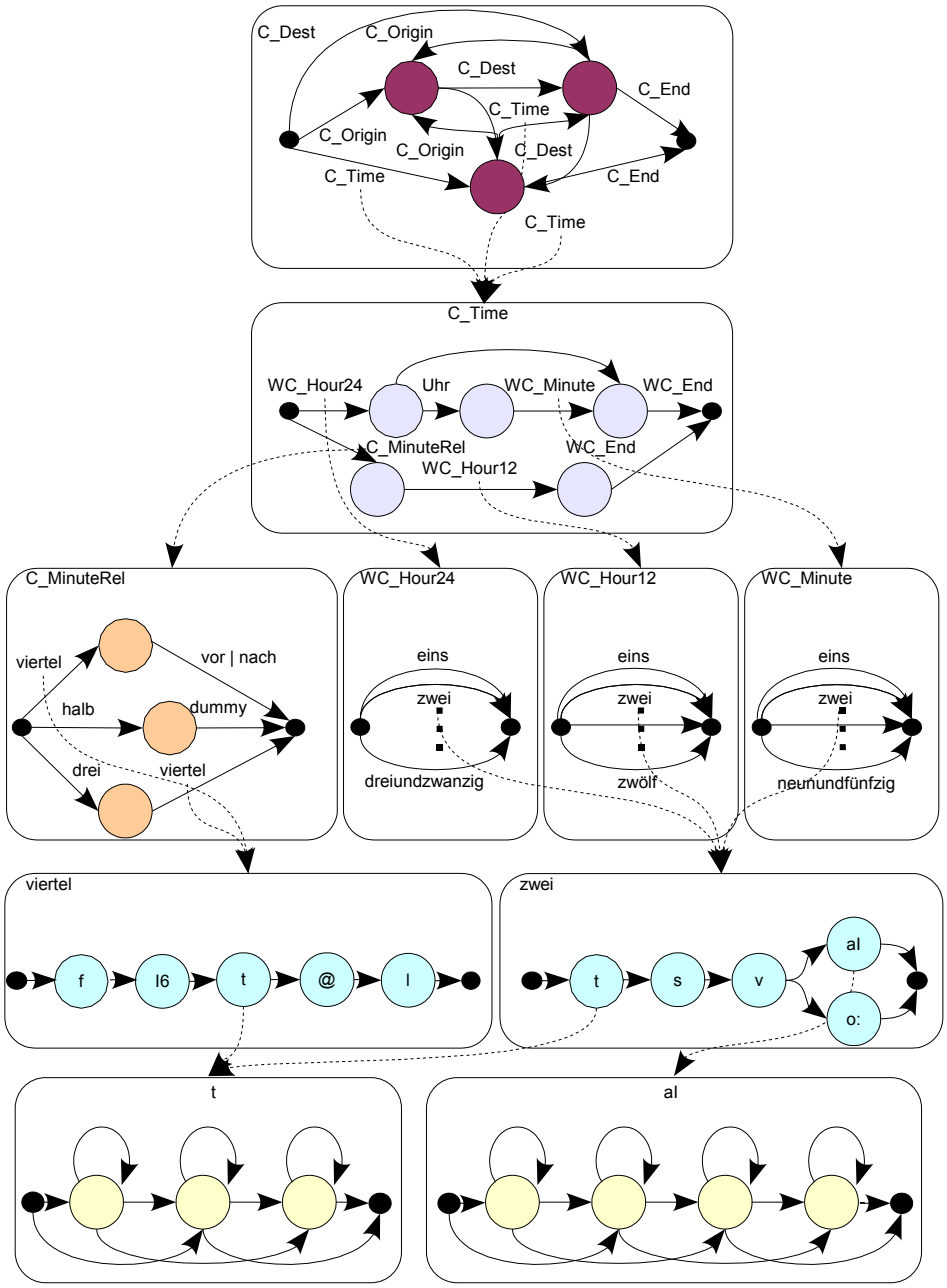


Abbildung 4.6: Beispiel eines implementierten hierarchischen Transitionsnetzwerks. Durch die gleichzeitige, integrierte Suche in allen Hierarchieebenen wird zu jedem Zeitpunkt t versucht, das Erkennungsergebnis des Spracherkenners semantisch zu decodieren.

Geringerer Annotierungsaufwand durch Kombination von Trainingsdaten mit ECFG

Im Gegensatz zur integrierten Modellierung von Thomae und Lieb [Tho06, Lie06] und zum HVS-Modell [HY05] basiert das Netzwerk hier nicht auf aufwendig hierar-

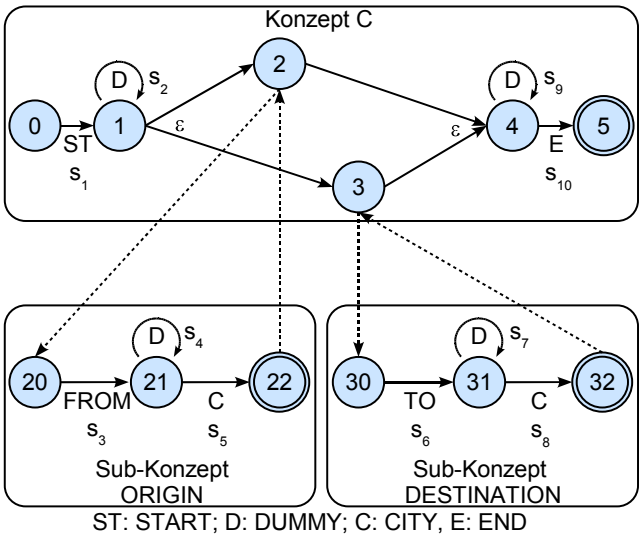


Abbildung 4.7: Schematische Darstellung eines nicht-deterministischen hierarchischen endlichen Automaten, welcher aus erweiterten, kontextfreien Grammatiken (ECFG) gebildet wird.

chisch annotierten Trainingsdaten, sondern auf einfach zu erstellende Grammatiken, welche mithilfe von Experten gebildet werden.

Die Zuordnung eines Wortes zu einer Wortklasse $W \mapsto L$ wird hier aus den Trainingsdaten gelernt. Diese Beziehung kann unabhängig von der Anwendung trainiert werden, da meist die Zuordnung eines Wortes W zu einer Wortklasse L semantisch bedeutungslos ist. Zudem ist diese Zuordnung leicht zu annotieren. Neben dem geringeren Annotationsaufwand wirkt sich auch die höhere Flexibilität des Systems in den verschiedenen Anwendungsmöglichkeiten aus.

4.3.3 Transitionsmatrix

Die Struktur des endlichen Automaten wird in eine Zustandsübergangsmatrix $\mathbf{T} = [t_{ij}]_{N \times N}$, mit $t_{ij} = P(l_j|l_i)$ (siehe Abschnitt 3.3) mit definierten Zuständen $s_1, \ldots, s_N$ abgebildet [SGS+08]. Die Transitionen δ_T werden bei ambigen Grammatiken mithilfe eines Trainingskorpus statistisch gewichtet.

Das generierte Übergangsnetzwerk (siehe Abbildung 4.7) kann beispielhaft in eine äquivalente Matrix mit 10 Zuständen $s_1, \ldots, s_{10}$ überführt werden.

$$\mathbf{T} = \begin{array}{c|cccccccccc} & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} \\ \hline s_1 & 0 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ s_2 & 0 & 1/3 & 1/3 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ s_3 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ s_4 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ s_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ s_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ s_7 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ s_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ s_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ s_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{4.11}$$

Die in der Trainingsphase errechneten statistischen Gewichtungen summieren sich in jeder Zeile i der Matrix zu

$$G(t_i) = \sum_{j=1}^{N} t_{ij} = 1. \tag{4.12}$$

Somit werden in $\mathbf{T}$ alle regelbasiert definierten Transitionen beschrieben. Jeder Zustand (Zeile der Matrix) s_i zeigt auf ein terminales Zeichen, welches in einem Zustandsvektor eingebettet ist. Der Vektor $\mathbf{l}$ (siehe Gleichung 4.13) beschreibt die auf Wortebene existierende Korpusannotation. Für ein Satzende wird die Bezeichnung (*Label*) `ENDE` gewählt. Aus diesem Grund existieren in der letzten Zeile der Matrix $\mathbf{T}$ keine Transitionen. Die Zahl 1 in der letzten Spalte der letzten Zeile der Matrix $\mathbf{T}$ drückt aus, dass der Endzustand erreicht wurde und dass der Automat in diesem verbleibt.

$$\begin{aligned} \mathbf{l} = (&\texttt{START}, \texttt{DUMMY}, \texttt{VON}, \texttt{DUMMY}, \texttt{STADT}, \\ &\texttt{NACH}, \texttt{DUMMY}, \texttt{STADT}, \texttt{DUMMY}, \texttt{ENDE}) \end{aligned} \tag{4.13}$$

Hierarchische Struktur

Die aus Gleichung 4.10 überführte Transitionsmatrix $\mathbf{T}$ repräsentiert alle terminalen Symbole der ECFG in einer hierarchischen Struktur, welche in der nachfolgenden Liste $\mathbf{H}$ dargestellt werden.

$$\begin{aligned} \mathbf{H} = (&1 \mapsto \texttt{S}; 2 \mapsto \texttt{S,DUMMY}; 3 \mapsto \texttt{S,ORIG,FROM}; 4 \mapsto \texttt{S,ORIG,DUMMY}, \\ &5 \mapsto \texttt{S,ORIG,CITY}; 6 \mapsto \texttt{S,DEST,TO}; 7 \mapsto \texttt{S,DEST,DUMMY}; \\ &8 \mapsto \texttt{S,DEST,CITY}; 9 \mapsto \texttt{S,DUMMY}; 10 \mapsto \texttt{S}) \end{aligned} \tag{4.14}$$

Die hierarchische Strukturliste (siehe Gleichung 4.14) kann zusammen mit der Matrix $\mathbf{T}$ in ein Transitionsdiagramm mit zwei aufeinanderfolgenden Zeitpunkten

t und $t+1$ dargestellt werden. Die semantische Decodierung eines `DUMMY`-Wortes kann sowohl im Konzept `ZIELORT`, als auch im `AUSGANGSORT` stattfinden (siehe Abbildung 4.8).

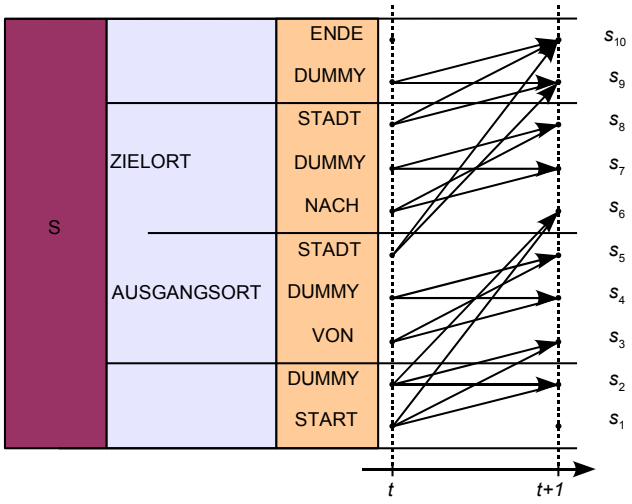


Abbildung 4.8: Mögliche Zustandsübergänge $t-1 \mapsto t$ werden in einem Transitionsdiagramm dargestellt. Hier werden die Übergänge aus Gleichung 4.10 beispielhaft mithilfe der codierten Zustände gebildet.

Decodierung

Die Decodierung einer „ungeparsten“ Wortfolge $\mathbf{w}$ erfolgt dynamisch, dargestellt in einem Trellisdiagramm (siehe Abbildung 4.9). Auf der Zeitachse (Abszisse) werden die besterkannten Wörter vom Spracherkenner aufgetragen und auf der Ordinate die verfügbaren Wortklassen $\mathbf{l} = (l_1, \ldots, l_T)$. Mit den Wortklassen werden ebenso die semantischen Konzepte, welche aus den Grammatikregeln stammen, auf der Ordinate aufgetragen.

Die Decodierung erfolgt mittels dynamischer Programmierung [SBBW04], d. h. der beste Weg $\hat{\mathbf{q}} = (\hat{q_1}, \ldots, \hat{q_T})$ wird durch Konkatenation der Transitionsdiagramme (siehe Abbildung 4.8) im aufgespannten Suchraum $\mathcal{S}$ gefunden. Die Suche des besten Pfades führt vom Startknoten zum Zielknoten und erstreckt sich über die gesamte Satzlänge.

Integration des Spracherkenners: In diesem Decodierverfahren kann das Erkennungsergebnis eines Spracherkenners einbezogen werden. Damit werden aufgrund der bisherigen semantischen Decodierung und der zu erwartenden grammatikalischen Satzstellung schlechte oder gar falsche Erkennungsergebnisse des Spracherkenners mithilfe des Suchstrahls eingeschränkt bzw. korrigiert. Die Erkennung der

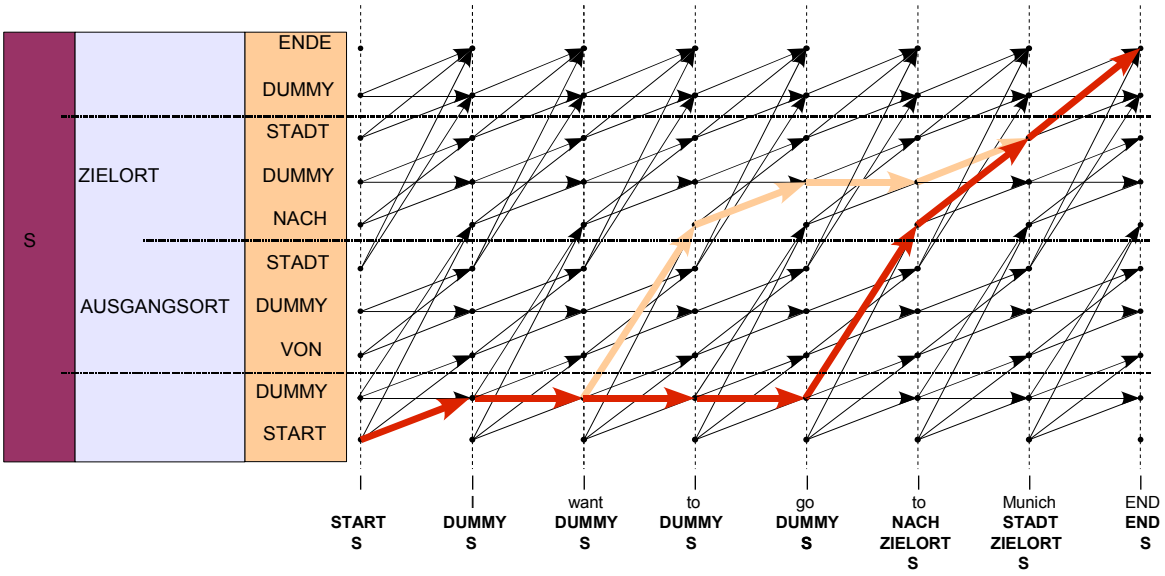


Abbildung 4.9: Durch „zeitliches Ausrollen“ der in Abbildung 4.8 dargestellte Transition ensteht ein Trellisdiagramm. Jeder vollständige Pfad durch den Trellis (d. h. von $t = 0$ bis $t = T - 1$) entspricht einer eindeutigen semantischen Zuordnung aller gesprochenen Wörter. Hier führen zwei Pfade vom Endzustand (oben rechts) in den Anfangszustand (unten links) führen. Der beste Pfad wird mittels des Backtracking-Algorithmus gefunden wird [SGS+08].

besten Wort-Wortklassen-Beziehung $P(l_i|w_t)$ aus einer Worthypothese w_t und einer gegebener Wortklasse $l_i \in \mathcal{L}$ wird mithilfe des „Satzes von Bayes“ erreicht [Rüg99]. Die Parameter in Gleichung (4.15) werden mit einem auf der Wortebene l annotierten Korpus gelernt.

$$P(l_i|w_t) = \frac{P(w_t|l_i) \cdot P(l_i)}{P(w_t)} \tag{4.15}$$

In dem englischen Beispielsatz, `I want to go to Munich`, hat die Präposition `to` verschiedene Bedeutungen; einmal beschreibt es mit „`to go`“ die Handlung, ein andermal bestimmt es den Ort. In diesem einfachen Beispiel wird das Prinzip der hierarchischen Decodierung in einem TRELLIS-Diagramm (siehe Abbildung 4.9) verdeutlicht. Dazu werden in der y-Achse die terminalen Symbole $\mathbf{l}$ mit der hierarchischen Struktur $\mathbf{H}$ dargestellt, und die x-Achse repräsentiert die Wortfolge $\mathbf{w}$ im Zeitverlauf t.

$$\mathbf{w} = (\texttt{START}, \texttt{I}, \texttt{want}, \texttt{to}, \texttt{go}, \texttt{to}, \texttt{Munich}, \texttt{END}).$$

Viterbi-Algorithmus: Der aus der Spracherkennung bekannte und bereits eingeführte Viterbi-Algorithmus (siehe Abschnitt 2.2.4) wird hier modifiziert für die

Erkennung semantischer Slots verwendet. Die Variable $\delta_t(i)$ beschreibt die Vorwärtswahrscheinlichkeit vom Zustand $q_t \mapsto q_t + 1$. Sie berechnet sich aus dem Produkt der Emissionswahrscheinlichkeit der Wortklasse l_i und dem Argument des Maximums der Zustandsübergangswahrscheinlichkeit t_{ij} und dessen Knotenwahrscheinlichkeit $\delta_{t-1}(i)$. Mit $\psi_t(i)$ wird die Rückschlusswahrscheinlichkeit bestimmt.

Nach Gleichung 4.15 wird in $\delta_1(i)$ für ein gesprochenes Wort w die beste Wortklasse l_i initialisiert. Die Rückschlusswahrscheinlichkeit wird für alle N-Wortklassen mit 0 initialisiert, d. h.

$$\begin{matrix} \delta_1(i) = P(l_i|w_1) \\ \psi_1(i) = 0 \end{matrix} \text{ , } 1 \leq i \leq N. \tag{4.16}$$

Mit der Erkennung der besten Wortklasse l_i zum Zeitpunkt t und der mittels ECFG definierten Semantik können alle weiteren „semantischen“ Pfade **rekursiv** berechnet werden.

Rekursion:

$$\begin{matrix} \delta_t(j) = \max\limits_{1 \leq i \leq N} [\delta_{t-1}(i) \cdot t_{ij}] \cdot P(l_j|w_t) \\ \psi_t(j) = \operatorname{argmax}\limits_{1 \leq i \leq N} [\delta_{t-1}(i) \cdot t_{ij}] \cdot P(l_j|w_t) \end{matrix} \text{ , } \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{4.17}$$

Terminierung:

$$\hat{q}_T = \operatorname*{argmax}_{1 \leq i \leq N} \delta_T(i) \tag{4.18}$$

Der beste Weg wird durch „Backtracking“ gefunden:

$$\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1}). \tag{4.19}$$

Mit den aufgedeckten Zuständen aus der Rückverfolgung des besten Pfades (*Backtracking*) kann mit diesem Verfahren die beste hierarchische Struktur eines Beispielsatzes $\mathbf{w}$ durch die Hierarchieliste $\mathbf{H}$ abgeleitet werden. Der beste Weg, als (→) in Abbildung 4.9 gekennzeichnet, des beobachteten Satzes $\mathbf{w}$ führt zu einer semantisch sinnvollen Decodierung.

Der Viterbi-Algorithmus schließt den Alternativpfad, als (→) gekennzeichnet, aufgrund der geringen relativen Häufigkeit (Emissionswahrscheinlichkeit) $P(l_i|w)$ im Trainingskorpus aus. In diesem Pfad, wird im Gegensatz zum besten Pfad das erste Wort `to` als Präposition des Ortes decodiert. Demzufolge werden dann die Wörter `go` und `to` semantisch als `DUMMY`-Wörter decodiert, ehe das Wort `Munich` wieder als `CITY` erkannt wird.

4.3.4 Experimente

Für die Versuchsdurchführung werden die hierarchisch annotierten Daten des deutschsprachigen NADIA-Korpus verwendet (siehe Abschnitt 3.1). Die hierarchische Annotierung dient zugleich als Referenzsystem (*Baseline System*) der Experimente.

Bewertungsmaße

Ein Satz w wird akzeptiert, wenn die automatisch generierte hierarchische Struktur mit dem Referenzsystem übereinstimmt. Dazu wird das Bewertungsschema aus Abschnitt 2.6 mit der vollständigen semantischen Decodierung eines gesprochenen Satzes und der semantischen Decodierung von natürlich gesprochenen Wortgruppen konkretisiert.

Überprüfung auf Satzebene Ein Satz w wird als falsch erkannt, wenn ein einzelnes Wort w_i der semantischen Decodierung c sich von dem Referenzsystem unterscheidet. Dies wird mit der Satzakzeptanzrate

$$\mathrm{AR}_{\mathrm{Satz}} = \frac{\text{Anzahl der semantisch richtig decodierten Sätze}}{\text{Anzahl der Sätze}} \tag{4.20}$$

gemessen.

Die Genauigkeit (*Precision*) berechnet sich aus dem Verhältnis an richtig erkannten semantischen Konzepten, die tatsächlich in den Trainingsdaten vorhanden waren, zu allen (richtig und falsch) erkannten semantischen Ereignissen.

$$\mathrm{Precision}_{\mathrm{Satz}} = \frac{\text{richtig erkannte Konzepte}}{\text{richtig erkannte Konzepte} + \text{falsch erkannte Konzepte}} \tag{4.21}$$

Die Trefferquote (*Recall*) berechnet sich aus dem Verhältnis an richtig erkannten semantischen Konzepten zu allen (richtig und nicht) erkannten semantischen Konzepten.

$$\mathrm{Recall}_{\mathrm{Satz}} = \frac{\text{richtig erkannte Konzepte}}{\text{richtig erkannte Konzepte} + \text{nicht erkannte Konzepte}} \tag{4.22}$$

Überprüfung auf Wortebene Um nun die automatische Decodierung auf Wortklassenebene l zu überprüfen, wird eine Wortakzeptanzrate $\mathrm{AR}_{\mathrm{Wort}}$ eingeführt:

$$\mathrm{AR}_{\mathrm{Wort}} = \frac{\text{Anzahl der semantisch richtig decodierten Wörter}}{\text{Anzahl der Wörter}} \tag{4.23}$$

Die Genauigkeit (*Precision*) ist der Anteil an erkannten Ereignissen, die tatsächlich in den Daten vorhanden waren, zu allen erkannten semantischen Ereignissen:

$$\mathrm{Precision}_{\mathrm{Wort}} = \frac{\text{richtig erkannte Wortklassen}}{\text{richtig erkannte Wortklassen} + \text{falsch erkannte Wortklassen}} \tag{4.24}$$

Die Trefferquote (*Recall*) berechnet sich aus dem Verhältnis richtig erkannter Wortklassen, zu allen (richtig und nicht) erkannten Wortklassen.

$$\mathrm{Recall}_{\mathrm{Wort}} = \frac{\text{richtig erkannte Wortklassen}}{\text{richtig erkannte Wortklassen} + \text{nicht erkannte Wortklassen}} \tag{4.25}$$

Mithilfe der Genauigkeit (*Precision*) und der Trefferquote (*Recall*) kann die semantische Exaktheit (*goal detection accuracy*) analog zu Abschnitt 2.6 bestimmt werden.

Die besten drei Grammatiken $\mathcal{G}_{\{1,2,3\}}$ (zu finden in Anhang 1.2) mit den Wort- AR_{Wort} und Satzerkennungsraten AR_{satz} werden in Tabelle 4.2 dargestellt. Zur Betrachtung der Güte des Systems werden die Grammatiken $\mathcal{G}_{\{1,2,3\}}$ mit dem bekannten F-Score (siehe Abschnitt 2.6) dargestellt.

Perplexität Die Perplexität $\xi(\cdot)$ ist ein Maß für die Qualität des Modells und ist mit der Kreuzentropie in der Exponentialfunktion zur Basis 2 definiert. Für weniger komplexe Grammatiken erwartet man, dass die Zahl der möglichen Konzeptzuständen $\mathbf{c}$ und damit die Kreuzentropie zunimmt, dies bedeutet gleichzeitig eine Zunahme der Perplexität. Dies zeigt sich auch in den verwendeten Grammatiken, die Perplexitäten verhalten sich umgekehrt proportional zu den Erkennungsraten:

$$\xi(\mathcal{G}_1) < \xi(\mathcal{G}_2) < \xi(\mathcal{G}_3) \tag{4.26}$$

$$AR_{\text{Wort}}(\mathcal{G}_1) \gg AR_{\text{Wort}}(\mathcal{G}_2) \gg AR_{\text{Wort}}(\mathcal{G}_3) \tag{4.27}$$

	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$
AR_{Satz}[%]	$92,8$	$94,7$	$95,4$
AR_{Wort}[%]	$95,5$	$96,9$	$97,8$
$N(s)$	89	98	113
Genauigkeit (*Accuracy*) [%]	$93,1$	$94,0$	$95,1$
Fehlerrate (*Error*) [%]	$6,9$	$6,0$	$4,9$
F-Score (*Measure*) [%]	$93,3$	$95,0$	$95,6$

Tabelle 4.2: F-Score für die semantische Decodierung mit dem vorgestellten stochastischen Verfahren und erweiterter kontextfreier Grammatiken. Die Erkennung von R_{Satz} wird als richtig gewertet, wenn für einen gesprochenen Satz alle semantischen Informationen richtig detektiert werden. Die Erkennung von R_{Wort} wird als richtig gewertet, wenn ein einzelnes Wort der richtigen semantischen Gruppe zugeordnet wird.

Die Verbesserung der F-Score (siehe Abbildung 4.10) resultiert aus der Erweiterung der Grammatikregeln $\mathcal{G}$. Im $3.$ Versuch wurde der Regelblock um die Uhrzeit erweitert (siehe Anhang 1.2). Dies führte zu einer deutlichen Verbesserung des Erkennungsergebnisses mit der Konsequenz eines weiteren Anstiegs der Zustandsanzahl $N(s)$. Für eine dahinterliegende Datenbankabfrage (z. B. SQL) muss der komplette Satz semantisch korrekt decodiert werden.

Die möglichen Zustandsübergänge wurden auf Wortebene mit einem Trainingsset $\mathcal{S}_{\text{Train}}$ mit $N(\mathcal{S}_{\text{Train}}) = 1\,065$ annotierten Sätzen aus dem NADIA-Korpus trainiert. Die erzielten Ergebnisse basieren auf einem zu $\mathcal{S}_{\text{Train}}$ disjunkten Testset $\mathcal{S}_{\text{Test}}$ mit $N(S_{\text{Test}}) = 300$ Sätzen. Es wird mit der besten Grammatik $\mathcal{G}_3$ eine Wortakzeptanzrate $\text{AR}_{\text{Wort}} = 97,8\%$ und $\text{AR}_{\text{Satz}} = 95,6\%$ erreicht. Daraus berechnet sich ein F-Score von $95,6\%$ für Wörter w und $95,4\%$ für ganze Sätze $\mathbf{w}$.

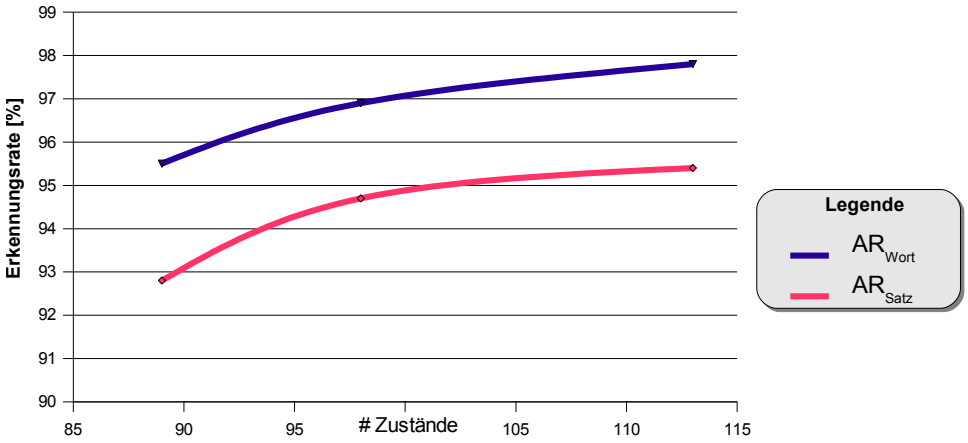


Abbildung 4.10: Vergleich von Satz- und Worterkennungsrate in Abhängigkeit der Anzahl der Zustände der erweiterten kontextfreien Grammatik.

Mit der Erhöhung der Anzahl der Zustände $N(s)$ in der Grammatik bis an das Transduktor-Modell (siehe Abschnitt 4.1.2) werden die (Satz- und Wort-) Erkennungsraten stetig verbessert.

4.4 Graphische Modelle

Die vorliegenden Modelle der automatischen Spracherkennung mittels stochastischer Verfahren und ECFG werden in diesem Abschnitt durch GM modelliert, erweitert und deren Ergebnisse mit den bereits eingeführten Gütekriterien (siehe Kapitel 2) verglichen. Der Unterschied zu Abschnitt 4.3 ist, dass in den Experimenten zwei semantische Ebenen mit stochastischen Verfahren modelliert werden: Konzepte und Wortklassen. Die im Rahmen der semantischen Decodierung verwendeten GM lassen sich auf theoretischer Basis mit faktoriellen HMM[6] vergleichen [GJ97].

[6]Im faktoriellen HMM beobachten zwei verborgene stochastische Prozesse unabhängig voneinander ein Oberservierung $\mathbf{o}$.

4.4.1 Stochastische Modellierung der semantischen Decodierung

Das System kennt nur das aktuelle Wort w und muss daraus auf das unbekannte Konzept c und Label (Wortklasse) l schließen.

Graphisches Modell 1 (GM1)

Das GM aus [SGS+08], hier als GM1 bezeichnet, berücksichtigt die stochastische Bindung der diskreten, beobachtbaren Wortknoten w_t mit der Wortklasse l_t und der Konzeptklasse c_t. Ebenso berücksichtigt das Modell die stochastische Bindung der versteckten Wortklasse l_t mit der Konzeptklasse c_t. Dies ist in Abbildung 4.11 durch die Wahrscheinlichkeitsverbindungen zwischen dem **diskreten Beobachtungsknoten** w und den restlichen **diskreten** Knoten verdeutlicht. Aus Abbildung 4.11 kann

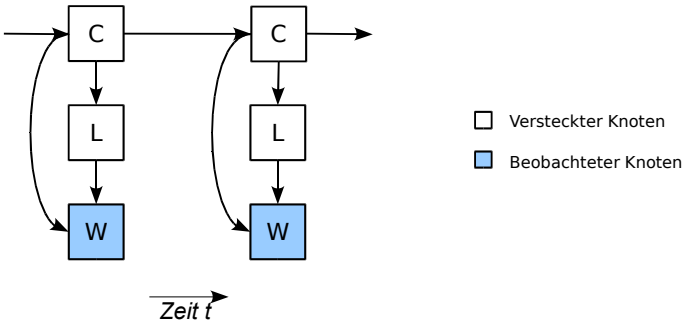


Abbildung 4.11: Diskrete Modellierung nach [SGS+08]. GM1 beschreibt die stochastische Bindung der Konzepte $c_{t-1} \rightarrow c_t$.

die folgende Verbundwahrscheinlichkeit $P(GM1)$ abgeleitet werden:

$$P(GM1) = P(c_1) \cdot P(l_1|c_1) \cdot P(w_1|c_1, l_1) \cdot \prod_{t=2}^{T} P(c_t|c_{t-1}) \cdot P(l_t|c_t) \cdot P(w_t|c_t, l_t) \quad (4.28)$$

Jeder Knoten erhält eine bedingte Wahrscheinlichkeit, immer in Abhängigkeit seiner Elternknoten. Die bedingte Wahrscheinlichkeit $P(c_t|c_{t-1})$ stellt die Abhängigkeit des Konzeptes vom Konzept des vorigen Wortes dar. Dies ist die einzige Abhängigkeit zwischen zwei aufeinanderfolgenden Zeitschlitzen $t-1 \rightarrow t$.

Alle Abhängigkeiten innerhalb eines Zeitschlitzes wurden durch Sprachmodelle (siehe Abschnitt 4.1.2) modelliert. Ähnlich zum faktoriellen HMM existieren zwei verborgene Zustände (l und c), jedoch sind deren stochastische Bindungen hier anders modelliert; bei einem bestimmten Wort w hat jede Wortklasse l und jedes Konzept c eine bestimmte Wahrscheinlichkeit. Die Parameter des Modells werden aus hierarchisch annotierten Trainingsdaten, welche in Kapitel 3 beschrieben sind, gelernt. Mithilfe des Viterbi-Algorithmus [Vit67] wird die wahrscheinlichste Folge von Konzepten **c** und Wortklassen **l** decodiert.

In Tabelle 4.3 werden die Ergebnisse, die mit dem Graphischen Modell GM1 erzielt wurden, dargestellt. Die Erkennungsraten RR und das F-Score aller zehn Gruppen sowie in der letzten Spalte eine nach der Anzahl der enthaltenen Wörter gewichtete Durchschnittszahl, liegen noch deutlich unter dem Referenzsystem, dem HVS-Modell.

Gruppe [%]	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	Ø
RR für c	89,5	91,3	89,4	91,4	91,6	91,7	88,6	91,3	89,8	92,3	**90,7**
RR für l	92,4	93,1	92,3	94,2	94,6	93,8	92,8	92,6	92,4	94,3	**93,2**
F für c	89,4	91,1	89,4	91,5	92,1	91,6	88,8	91,6	89,8	92,3	**90,8**
F für l	92,5	93,1	92,5	94,3	95,0	94,1	93,2	93,0	92,6	94,5	**93,5**

Tabelle 4.3: Ergebnisse von GM1 mit 10-facher Kreuzvalidierung erzielt.

Die Schwächen des Modells sind, dass Abhängigkeiten von Wörtern über zwei oder mehr Zeitschritte $t-2-n \longrightarrow t$ für $n \in \mathbb{N}^+$ nicht berücksichtigt werden. Das Konzept kann aber auch abhängig sein von der Wortklasse l des Wortes des Zeitpunktes $t-1$. Die Modellierung von Out-Of-Vocabulary (OOV)-Wörtern, die nur einmal oder gar nicht in den Trainingsdaten vorkommen, wird in diesem Modell nicht berücksichtigt. Für eine großteils fehlerfreie semantische Decodierung von Standardsätzen eines Auskunftssystems (z. B. Flugauskunft) reicht das einfache GM1 aus.

Graphisches Modell 2 (GM2)

Bei GM2 handelt es sich um eine einfache Erweiterung des GM1 . Mithilfe einer hinzugefügten **Kante** $l_{t-1} \rightarrow c_t$ wird zusätzlich die Abhängigkeit des Wortes w_t einer Konzeptklasse c_t von der vorigen Wortklasse l_{t-1} modelliert.

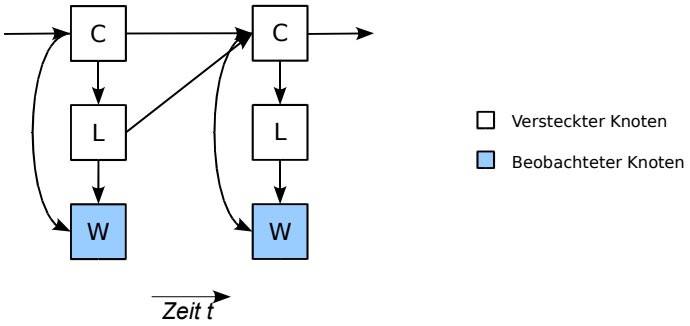


Abbildung 4.12: Deterministische Modellierung nach [SGS+08]. GM2 beschreibt gegenüber GM1 zusätzlich die stochastische Bindung $l_{t-1} \rightarrow c_t$.

Aus Abbildung 4.12 kann die Verbundwahrscheinlichkeit $P(GM2)$ folgendermaßen faktorisiert werden: Der Faktor $P(c_t|c_{t-1}, l_{t-1})$ drückt die Abhängigkeit des Konzeptes c_t vom vorigen Konzept c_{t-1} und der vorigen Wortklasse l_{t-1} aus. Durch den zusätzlichen Faktor l_{t-1} wird das Modell in Abbildung 4.11 zu einem faktoriellen HMM Decodiersystem (siehe Abbildung 4.12) erweitert.

$$P(GM2) = P(c_1) \cdot P(l_1|c_1) \cdot P(w_1|c_1, l_1) \cdot \prod_{t=2}^{T} P(c_t|c_{t-1}, l_{t-1}) \cdot P(l_t|c_t) \cdot P(w_t|c_t, l_t) \tag{4.29}$$

Mit der stochastischen Bindung wird dem Konzept c_t eines Wortes w_t ein zusätzlicher Elternknoten, die Wortklasse l_{t-1} des vorherigen Worts, hinzugefügt. Dadurch konnten die Erkenungsleistungen (RR und F-Score) gegenüber GM1 absolut um $2,3\%$ verbessert werden (siehe Tabelle 4.4). Für die Wortklasse wurde hier eine absolute Verbesserung von $0,6\%$ erzielt.

Gruppe [%]	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	Ø
RR für c	91,9	93,0	91,3	94,0	94,2	92,7	92,8	93,5	92,6	93,9	**93,0**
RR für l	93,4	93,4	92,8	94,5	94,5	93,8	93,9	93,7	93,5	94,3	**93,8**
F für c	92,0	93,0	91,4	94,3	94,7	93,0	93,2	93,9	92,9	94,0	**93,2**
F für l	93,4	93,4	93,0	94,7	94,9	94,1	94,3	94,1	93,7	94,5	**94,0**

Tabelle 4.4: Ergebnisse von GM2 mit 10-facher Kreuzvalidierung erzielt.

Durch die zusätzliche Abhängigkeit kann die Fehlerrate bei der Erkennung von Konzepten c nochmals verringert werden.

Graphisches Modell 3 (GM3)

Eine weitere Verbesserung der Erkennungsrate der semantischen Decodierung konnte durch die stochastische Bindung des Wortes w_{t-1} an das Konzept c_t des „faktoriellen Sprachmodells“ erreicht werden (siehe Abbildung 4.13). Das Konzept c_t ist in diesem Modell von allen drei Variablen $c_{t-1}, l_{t-1}, w_{t-1}$ des vorigen Zeitpunktes $t-1$ abhängig.

Gegenüber GM2 wird in GM3 der Faktor $P(c_t|c_{t-1}, l_{t-1})$ zu $P(c_t|c_{t-1}, l_{t-1}, w_{t-1})$ erweitert. Das Konzept c_t ist abhängig vom Konzept c_{t-1}, der Wortklasse l_{t-1} und des beobachtbaren Wortes w_{t-1} des Zeitpunktes $t-1$. Damit wird in Abbildung 4.13 die Verbundwahrscheinlichkeit $P(GM3)$ wie folgt faktorisiert:

$$P(GM3) = P(c_1) \cdot P(l_1|c_1) \cdot P(w_1|c_1, l_1) \cdot \prod_{t=2}^{T} P(c_t|c_{t-1}, l_{t-1}, w_{t-1}) \cdot P(l_t|c_t) \cdot P(w_t|c_t, l_t) \tag{4.30}$$

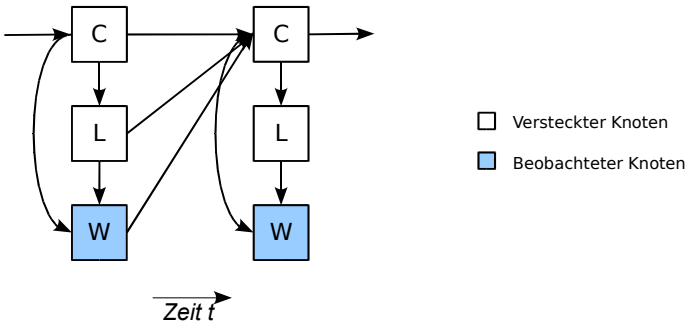


Abbildung 4.13: Diskretes Modell zur semantischen Decodierung nach [SGS+08]. GM3 beschreibt gegenüber GM2 zusätzlich die stochastische Bindung $w_{t-1} \rightarrow c_t$.

Mit den Veränderungen von Modell GM2 zu GM3 konnten die Erkennungsergebnisse (RR und F-Score) für die Konzepte c absolut um ca. $0,5\%$ verbessert werden. Die Ergebnisse für die Wortklassen l blieben fast identisch, zum Teil wurden sie in manchen Gruppen sogar verschlechtert, weil eine gewisse Sättigung erreicht wurde. Im Durchschnitt gab es aber eine leichte Steigerung. Bei dem Modell GM3 sind die Erkennungsraten für die Wortklassen um $0,6\%$ besser als die Erkennungsraten für die Konzepte.

Gruppe [%]	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	Ø
RR für c	93,1	93,4	92,9	92,6	94,9	93,8	92,6	93,3	93,3	94,9	**93,5**
RR für l	94,5	92,9	93,8	93,8	95,2	94,8	93,7	93,5	94,1	94,8	**94,1**
F für c	93,3	93,4	93,1	92,7	95,2	94,0	93,0	93,6	93,5	95,0	**93,7**
F für l	94,7	92,9	94,0	94,0	95,6	95,0	94,1	93,9	94,3	94,9	**94,3**

Tabelle 4.5: Ergebnisse von GM3 mit 10-facher Kreuzvalidierung erzielt.

Es wurden weitere Abhängigkeiten, wie z. B. die Abhängigkeit der Wortklasse l_t vom Konzept c_{t-1}, in verschiedenen Modellen untersucht und getestet. Diese führte aber zu keiner Ergebnisverbesserung.

Des Weiteren wurden Abhängigkeiten über mehrere Zeitschritte t untersucht. Auch das Modell das die Abhängigkeit des Konzeptes c_t von c_{t-2}, l_{t-2} oder w_{t-2} mit berücksichtigt, wodurch sich die Berechnungsdauer erhöhte aber keine weitere Verbesserung brachte. Die Ergebnisse waren alle insgesamt schlechter als das GM3. Auch Abhängigkeiten des Konzepts c_t von zwei Variablen aus dem „Frame“ $t-2$ führten zu keiner weiteren Verbesserung.

4.4.2 Regelbasierte Multi-Netze zur semantischen Decodierung

Ein Bayes'sches Multi-Netz ist ein Netz, in dem die Kanten erscheinen oder verschwinden, je nachdem welche Werte bestimmter Knoten im Graphen haben. In [GH96] wird dieses Verhalten als asymmetrische Unabhängigkeitsaussage bezeichnet.

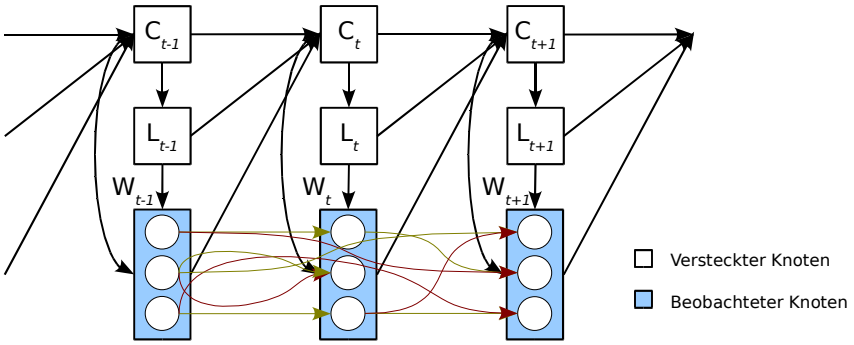


Abbildung 4.14: Switching Parents: Auswahl eines regelbasiert-definierten Wörterbuchs $\mathcal{W}$ zum Zeitpunkt t entsprechend dem vorliegenden Label l_t der Wortklasse.

Bayes'sche Multi-Netze [Bil00] oder schaltende Elternknoten (*switching parents*) (SPT) sind ein neuartiges Feature von GMTK, mit welchem ein Wechsel (*switching*) der Elternknoten abhängig vom bedingten momentanen Wert anderer Eltern erlaubt wird. In einem Entscheidungsbaum (*Decision Tree*) wird für jeden Zustand der „Switching Parents" gespeichert, welche „Conditional Parents" verwendet werden.

Mit bedingten Elternknoten (*conditional parents*) (CPT) kann eine Auswahl des Wortes $w_t \in \mathcal{W}$ aufgrund der vorliegenden Wortklasse l_t getroffen werden.

Im konkreten Fall werden die Grammatikregeln aus Abschnitt 4.3 in Algorithmus 2 als Entscheidungsbaum dargestellt.

Algorithmus 2 Entscheidungsbaum mit „Switching Parents" in GMTK realisiert.

```
wordClassToWordmapping
1
0 7 11 21 12 25 26 27 default
-1 0
-1 1
-1 2
-1 3
-1 4
-1 5
-1 6
```

In der dritten Zeile in Algorithmus 2 steht die Zahl 0 für den ersten Elternknoten, die Zahl 7 gibt an, wie viele Zustände von der Elternvariable unterschieden werden. Hier sind es sechs Zustände (11, 21, 12, 25, 26 und 27) und das „default“, welches für alle sonstigen Zustände verwendet wird. In den folgenden Zeilen wird jedes Blatt im Entscheidungsbaum beschrieben; z. B. beschreibt hier der Index 2 den „SparseCPT“ der Wortklasse $l = \texttt{Wochentag}$. Auf gleiche Weise werden die restlichen Wortklassen l mit grammatikalischen Regeln aus Abschnitt 4.3 definiert.

Graphisches Modell 4 (GM4)

In den bisherigen Modellen sind die Zustandsübergangswahrscheinlichkeiten durch stochastische Sprachmodelle realisiert. In diesem Modell werden deterministische Übergänge mithilfe von SPT realisiert.

Auf das ATIS-Szenario konkretisiert, bedeutet dies, dass beispielsweise auf die Wortklasse `FromLoc` das Konzept `Origin` mit der definierten stochastischen Bindung

$$P(C_t = \texttt{Origin} | L_{t-1} = \texttt{FromLoc}) = 1, \tag{4.31}$$

folgt. Ferner wurden aufgrund „Sparse-data“ im ATIS-Korpus OOV-Wörter in der jeweiligen Wortklasse l mittels Grammatiken (siehe Abschnitt 4.3) statistisch gleich verteilt; es gibt z. B. 60 Möglichkeiten, die Minuten in der Uhrzeit anzugeben. Jedes dieser Ziffernwörter der Wortklasse `N60` besitzt die gleiche Wahrscheinlichkeit $1/60$. Ähnlich wurden die Wortklassen `Monat`, `Tag`, `Datum`, `Wochentag` und `Stunden` modelliert.

Gruppe [%]	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	Ø
RR für c	92,8	93,6	92,7	92,9	95,2	94,2	93,0	93,8	93,7	95,4	**93,7**
RR für l	94,6	93,8	94,9	94,3	95,8	95,2	94,4	94,6	94,7	95,9	**94,8**
F für c	92,9	93,6	92,9	93,0	95,6	94,5	93,4	94,2	93,9	95,6	**94,0**
F für l	94,8	93,8	95,1	94,5	96,1	95,5	94,8	95,0	94,9	96,0	**95,1**

Tabelle 4.6: Ergebnisse von GM4 mit 10-facher Kreuzvalidierung erzielt.

Diese hybride Form der Modellierung, sowohl stochastisch als auch regelbasiert, ist der entscheidende Vorteil gegenüber der Modellierung mit HVS (siehe Abschnitt 4.2).

4.5 Darstellung der Ergebnisse

Die verschiedenen Methoden der semantischen Decodierung werden hier zusammengefasst und verglichen.

4.5.1 Messmethoden und Korpusbeschreibung

Die Ergebnisse in Abschnitt 4.4 und Abschnitt 4.4.2 werden nach Abschnitt 2.6 mit F-Score, der Erkennungsrate RR für die semantische Decodierung eines Wortes w_i in ein Konzept c_i und eine Wortklasse l_i bewertet und basieren auf den ATIS-Korpus (siehe Abschnitt 3.1). Dabei wird durch eine 10-fache Kreuzvalidierung sichergestellt, dass Trainings- $\mathcal{S}_{\text{Train}}$ und Testsets $\mathcal{S}_{\text{Test}}$ disjunkte Teilmengen von $\mathcal{S}$ sind. Im Falle der Evaluierung am ATIS-Korpus wurden zehn Gruppen zu je 84 Sätzen gebildet. Jeder Satz aus dem ATIS-Korpus wurde folglich einmal als unbekannter Satz dem Modell zur Klassifikation vorgelegt und analog zu Abschnitt 2.6 mit folgendem Kriterium bewertet:

TP (*True Positive*): Anzahl der relevanten Wörter, die richtig klassifiziert werden.

FN (*False Negative*): Anzahl der relevanten Wörter, die falsch klassifiziert werden.

FP (*False Positive*): Anzahl der Dummy-Wörter, die falsch (als relevant) klassifiziert werden.

TN (*True Negative*): Anzahl der Dummy-Wörter, die richtig klassifiziert werden.

Aus den zehn Teilergebnissen $\mathcal{S}_{\text{Test}}$ der Kreuzvalidierung wird eine Gesamterkennungsrate berechnet. In den Einzelergebnissen wird die unterschiedliche Anzahl an Wörtern $|W|$ je Testset $\mathcal{S}_{\text{Test}}$ passend gewichtet.

In nachfolgender Tabelle 4.7 werden die Eigenschaften der zehn Testsets $\mathcal{S}_1, \ldots, \mathcal{S}_{10}$ mit den Kardinalitäten der Wörter, der nicht-bedeutungstragenden „Dummy"-Konzepte, den semantischen Konzepten c und Wortklassen l dargestellt. Diese Kardinalitäten kommen durch die unterschiedliche Anzahl an Wörtern je Satz im ATIS-Task zustande.

Gruppe	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$	$\mathcal{S}_9$	$\mathcal{S}_{10}$
Anzahl Wörter	876	967	973	839	921	903	933	950	910	919
Konzept Dummy	412	473	458	396	407	419	420	431	436	436
Sonstiges Konzept	464	494	515	443	514	484	513	519	474	483
Wortklasse Dummy	412	474	457	396	408	419	420	430	435	436
Sonstige Wortklasse	464	493	516	443	513	484	513	520	475	483

Tabelle 4.7: Eigenschaften der „Testsets" $\mathcal{S}_1, \ldots, \mathcal{S}_{10}$ mit je 84 Sätzen aus dem ATIS-Task. In das Gesamtergebnis fließen die unterschiedliche Anzahl an Wörtern $N(w)$ der Einzelergebnisse mit ein. Des Weiteren werden alle Wörter nach nicht-bedeutungstragendem Konzept „Dummy" und den semantischen Konzepten c bzw. Wörtern l unterschieden.

4.5.2 Vergleich der Ergebnisse

In Tabelle 4.8 werden die Ergebnisse der F-Score aller vier Modelle noch einmal zusammenfassend dargestellt.

Gruppe	Ø(GM_1)		Ø(GM_2)		Ø(GM_3)		Ø(GM_4)	
RR für c	90,7		93,0		93,5		93,7	
RR für l	93,2		93,8		94,1		94,8	
F für c	90,8	92,2	93,2	93,6	93,7	94,0	94,0	94,6
F für l	93,5		94,0		94,3		95,1	

Tabelle 4.8: Vergleich der durchschnittlichen Erkennungsergebnisse der erstellten Graphischen Modelle.

Zusätzlich zu den in den Tabellen 4.3 bis 4.6 präsentierten Ergebnissen sind die durchschnittlichen Erkennungsraten (*F-Score*) aus Konzept c und Wortklasse l berechnet. Daraus ist die stetige Verbesserung der Modelle in der semantischen Decodierung ersichtlich: GM2 verbesserte die Ergebnisse im Vergleich zu GM1 um $1,4\%$. GM3 verbesserte sich gegenüber Modell GM2 um $0,4\%$ und Modell GM4 decodiert um zusätzliche $0,6\%$ besser als GM3.

4.5.3 Graphische Darstellung

Die dargestellten Ergebnisse in Abbildung 4.15 zeigen, dass die semantische Decodierung mit ECFG das Referenzsystem FST bei etwa gleicher Zustandsanzahl relativ um $5,6\%$ übertrifft. Darüber hinaus zeigen die Experimente, dass die regelhafte semantische Decodierung gegenüber der stochastischen Decodierung weniger Zustände benötigt, deren Erkennungsumfang aber auch dementsprechend reduziert ist.

Ein weiterer Vergleich aus Abbildung 4.15 zeigt, dass die semantischen Erkennungsergebnisse der GM Versuche unterhalb des in [HY05] vorgestellten Referenzsystems liegen. Die Gründe liegen in der Verwendung von wesentlich mehr Zuständen im HVS Modell (siehe Abbildung 4.15). Daraus resultiert ein größerer Decodieraufwand mit höherer Zeitkomplexität.

Die Vergleichszahlen basieren auf der Satzebene unter Verwendung der in Kapitel 3 beschriebenen Sprachmodelle.

4.6 Zusammenfassung des Kapitels

In diesem Kapitel wurden vier Systeme zur semantischen Decodierung im Kontext der Graphischen Modelle vorgestellt und mit einem Referenzsystem (HVS-Modell) verglichen. Das Hauptproblem ist, dass für die Entwicklung eines stochastischen

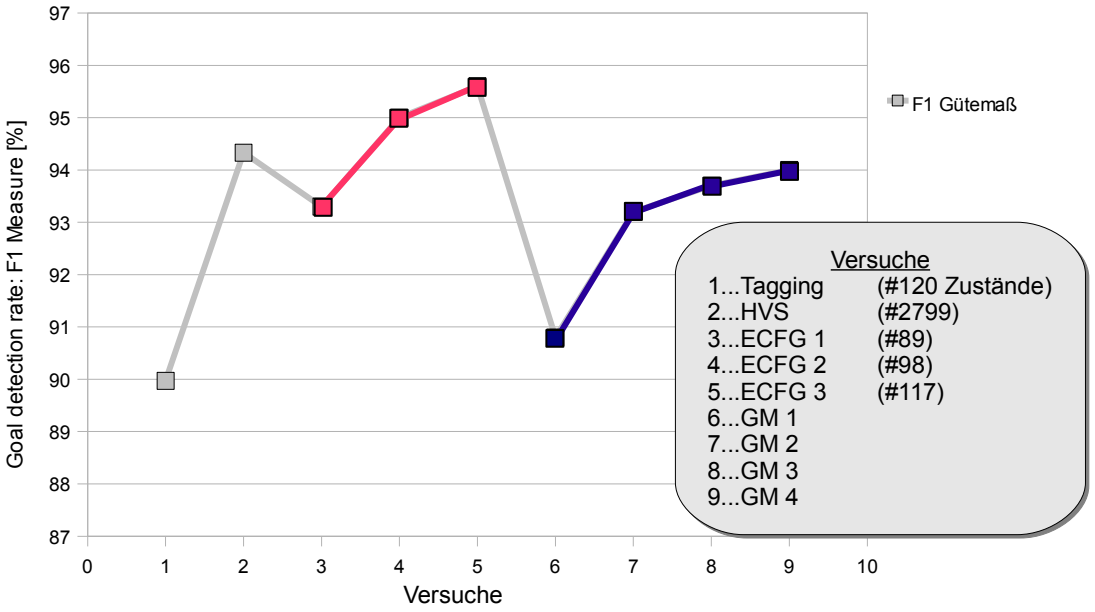


Abbildung 4.15: Graphische Zusammenfassung der Ergebnisse der in diesem Kapitel durchgeführten Experimente. Dargestellt sind semantische Exaktheit (*goal detection accuracy*) der Versuche $1 - 9$ auf dem ATIS-Korpus.

Modells genügend Trainingsdaten vorliegen müssen, um beispielsweise die Abschätzung der Modellparameter vornehmen zu können. Allerdings sind diese vollständig annotierten Trainingsdaten meist nicht vorhanden. Deshalb wurde versucht, aufgrund eines unvollständig annotierten Trainingskorpus (nur auf Wortebene annotiert bzw. wenig Trainingsbeispiele), Satzphrasen bzw. natürlich gesprochene Sätze semantisch zu decodieren.

Dazu wurde das HVS-Modell als Referenzsystem ausführlich analysiert. Gegenüber dem FST (*Tagging*)-Modell kann dies die hierarchische Struktur und somit die semantische Bedeutung über ganze Sätze erfassen. Nachteilig erwies sich die Einschränkung auf links- bzw. rechts- verzweigende Sprachen. Dies führt besonders in der deutschen Sprache bei der semantischen Decodierung von Uhrzeiten zu Schwierigkeiten. In [HY05] wurde gezeigt, dass auf dem ATIS-Korpus eine Erkennungsrate nach F-Score von $89,28\%$ erzielt wurde.

Bei kürzeren und regelhaft ausgesprochenen Satzphrasen wie z. B. Uhrzeiten eignet sich besonders der ECFG-Ansatz. Bei der Erkennung von Uhrzeiten, Abflugs- und Zielorten können auf der NADIA-Datenbank mit einer semantischen Grammatik von 113 Zuständen eine Satzerkennungsrate von $95,4\%$ und eine Worterkennungsrate von $97,8\%$ erzielt werden. Diese Ergebnisse wurden auf einem nur auf Wortebene annotierten Korpus (NADIA) erzielt. Durch eine flexible Anpassung der Grammatiken kann die Anwendungsdomäne, besonders für die Erkennung von regelhaft ausgesprochenen Satzphrasen schnell gewechselt werden. Allerdings zeigt

sich das System bei längeren natürlich gesprochenen Sätzen als anfällig.

Deshalb wurde im weiteren Verlauf des Kapitels ein System mit GM zur Interpretation von natürlich gesprochener Sprache vorgestellt. Im Gegensatz zu Grammatiken muss für die semantische Decodierung ein hierarchisch annotierter Trainingskorpus vorliegen. Hier wurden die Annotationen des ATIS-Korpus verwendet und insgesamt 26 Wortklassen und 12 semantische Konzepte decodiert. Dabei erzielte das beste GM nach einer 10-fachen Kreuzvalidierung mit je 90% Trainings- und 10% Testbeispielen eine Erkennungsrate nach F-Score von $93,7\%$ bei semantischen Konzepten und $95,1\%$ bei Wortklassen. Damit konnte gegenüber einem Referenzsystem eine relative Verbesserung von $3,7\%$ bzw. $8,7\%$ erzielt werden.

Mit den erzielten Ergebnissen wurde gezeigt, dass das System der GM gegenüber dem HVS Modell auf dem ATIS-Korpus bessere Ergebnisse erzielt. Dies ist auf die Verwendung von SPT zurückzuführen und mit der bereits erwähnten Einschränkung des HVS Modells zu erklären. Durch eine durchgängige graphische Modellierung kann außerdem das Ergebnis an das übergeordnete Dialogmanagementsystem weitergereicht werden (siehe Kapitel 5). Ebenso eignet sich das System der GM für eine Rückführung des Wissens aus der Dialogsteuerung in den Spracherkenner (siehe Kapitel 2).

5

Dialogmanagementsystem

Das Dialogmanagementsystem hat die Aufgabe, die Abfolge der Zustände, in denen sich ein Dialog befinden kann, zu bestimmen und daraus einen **zukünftigen Dialogzustand** abzuschätzen. Dabei ist das Ziel, einen so genannten „natürlichen Dialog“ zu realisieren.

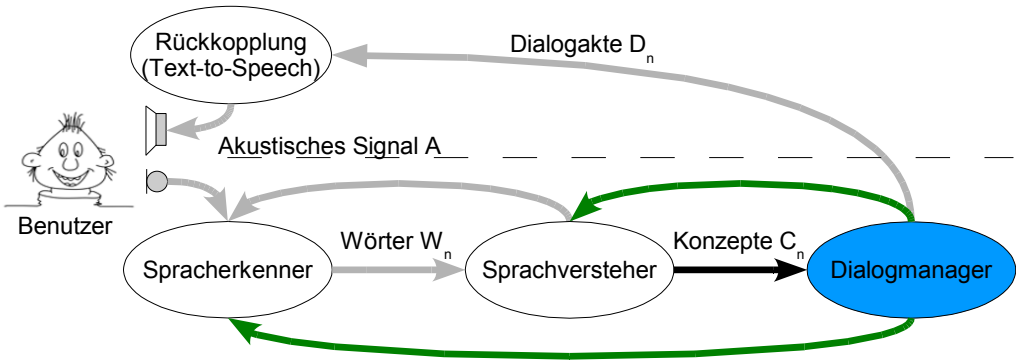


Abbildung 5.1: Das Dialogmanagementsystem erhält „semantische Slots“ von der semantischen Decodierung (Sprachversteher) und führt kontextbezogene Informationen der Spracherkennung und der semantischen Decodierung zurück. Mit der Integration der kontextbezogenen Informationen des Dialogzustands sind sowohl im Spracherkenner als auch in der semantische Decodierung bessere Erkennungsergebnisse zu erwarten.

Das Dialogmanagement bestimmt aus der semantischen Decodierung (siehe Kapitel 4) die Dialogstrategie und damit, wie das System auf (Sprach-)Eingaben des Benutzers reagiert (siehe Abbildung 5.1). Dabei werden externe Quellen, wie z. B. das Wissen über den Benutzer oder des Dialogverlaufs zur Entscheidungsfindung herangezogen. Der Dialog wird mit einer Antwort, welche an eine Sprachsynthese (*text-to-speech*) (TTS) weitergeleitet wird, vervollständigt. Die Antwortgenerierung ist praktisch das Gegenstück zur Sprachanalyse [Rus94].

In diesem Kapitel wird zunächst der POMDP [You06] als ein Referenzsystem für das Dialogmanagement vorgestellt. Aus den daraus resultierenden Erkenntnissen wird im Anschluss ein erweitertes Hidden-Markov-Modell für das Dialogmanagement (HMD), basierend auf GM, vorgestellt. Die beiden Systeme werden abschließend auf Basis des ATIS-Korpus, mit $21\,650$ gesprochenen Benutzeräußerungen in $1\,585$ natürlich gesprochenen Dialogen evaluiert. Die nachfolgend präsentierten Dialogmanagementverfahren wurden bereits in einer Veröffentlichung vorgestellt und etabliert. [SMS+09].

5.1 Markoffsche Entscheidungsprozesse

Die POMDP liefern Rahmenmodelle (*Framework*), deren Parameter mit einer Auswertung von natürlichsprachlichen Dialogen trainiert werden [You06]. Die prinzipielle Arbeitsweise erfolgt in zwei Phasen:

Training:

- Training des Modells anhand eines annotierten Trainingskorpus. Das Ziel dieses Schritts ist es, die Wahrscheinlichkeiten (Parameter) der Matrizen $\mathbf{T}$,$\mathbf{R}$,$\mathbf{A}$ festzulegen.
- Die Markoffschen Entscheidungsprozesse (*Markov Decision Processes*) (MDP) oder die POMDP werden mithilfe Approximationsalgorithmen (z. B. SARSOP [KHL08]) berechnet. Daraus resultiert eine Berechnungsvorschrift für Dialoge (*dialog policy*).

Laufzeit:

- Es erfolgt eine Aktualisierung des Belief-Vektors $\mathbf{b}$.
- Die Auswahl der Dialogaktion a nach der Berechnungsvorschrift.

Der POMDP ist eine Form der Markov-Prozesse. Die Annahme, die diesen Prozessen zugrunde liegt, ist, dass der Folgezustand nur vom aktuellen Zustand, nicht aber von vorhergehenden Zuständen abhängt. Die Zusammenhänge der verschiedenen Markov-Prozesse sind nach [Cas05] in Abbildung 5.2 dargestellt.

Der POMDP lässt sich als ein Markoffschen Entscheidungsprozesse (*Markov Decision Processes*) (MDP) darstellen, dessen Zustände nicht beobachtbar sind, in ähnlicher Weise wie das HMM die Zustände der Markov-Kette versteckt und nur die Beobachtung ausgibt, die mit bestimmten Wahrscheinlichkeiten den Zuständen zugeordnet werden kann. Der große Unterschied vom POMDP zum MDP ist also, dass das System nicht sicher weiß, in welchem Zustand es sich befindet, es kann nur anhand der Beobachtungen, die es macht, für jeden Zustand eine bestimmte Wahrscheinlichkeit ermitteln [Cas05].

		Können die Zustandsübergänge kontrolliert werden?	
		Nein	Ja
Sind die Zustände verborgen?	Nein	Markov Kette	MDP
	Ja	HMM	POMDP

Abbildung 5.2: Übersicht über die Markov-Prozesse: Die Markov-Kette oder ein einfach stochastischer Prozess, (links oben) enthält sichtbare Zustände, deren Übergänge sind jedoch nicht kontrollierbar. Das HMM oder der doppelt stochastische Prozess, (links unten) besteht aus versteckten Zuständen, deren Zustandsübergänge nicht kontrollierbar sind. Äquivalent verhalten sich die MDP (rechts oben) zu den POMDP (rechts unten): Darin sind die Zustandsübergänge kontrollierbar, jedoch sind im POMDP die Zustände wie im HMM versteckt.

5.1.1 Entscheidungsprozess nach Markov (MDP)

Ein MDP besteht aus einem 4-Tupel $M = (\mathcal{S}, \mathcal{A}, \mathbf{T}, \mathbf{R})$, dessen Variablen definiert sind als:

Zustand $\mathcal{S}$ beschreibt die Menge der Zustände in einem für den Dialog definierten Zustandsraum.

Aktion $\mathcal{A}$ beschreibt die Menge an möglichen Dialogaktionen.

Zustandsübergang Matrix $\mathbf{T}$, beschreibt die Übergangswahrscheinlichkeit, von einem Zustand s_i in den nächsten Zustand s_j zu wechseln.

Belohnung Matrix $\mathbf{R}$, beschreibt die direkte Belohnung eines Dialogschrittes, also der Übergang von Zustand s in den Zustand s'. Ein einzelner Belohnungsschritt wird im Folgenden mit $r(s, s')$ beschrieben.

MDP sind zeitdiskret wie Markov Modelle oder HMM. Der Markov Prozess bezieht sich dabei auf die Übergangswahrscheinlichkeiten der Zustände.

Wahl der bestmöglichen Aktion: In jedem Zeitschritt wird eine Belohnung der Transition berechnet. Das Ziel der Belohnung ist es, eine Berechnungsvorschrift (*policy*) $\pi := S \rightarrow A$ zu finden, um die angesammelten Belohnungen r aller Zustände s mit dem bedingten Erwartungswert

$$E\left[\sum_{t=0}^{\infty} \gamma^t \, r_t\right] , \tag{5.1}$$

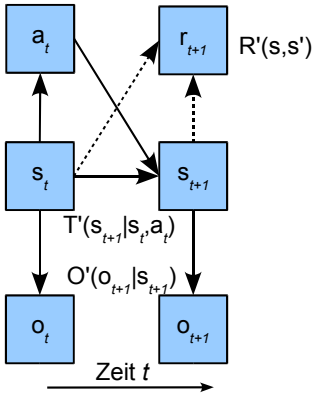


Abbildung 5.3: Markov Decision Process, realisiert als Graphisches Modell.

zu maximieren. Dabei bezeichnet γ den Discountfaktor, der zwischen 0 und 1 liegt. Dieser berechnet weniger die Belohnung, sondern viel mehr die Länge eines Dialoges. Die Idee ist dabei, dass das System näher am Ziel liegende Lösungen bevorzugt [WY06]. Welche Aktion a in einem bestimmten Zustand s gewählt wird, beschreibt ein stochastisches Regelwerk (*policy*). Dieses Regelwerk wird in einem Vektor $\mathbf{A}$ des MDP gespeichert.

Lernvorgang: Der Lernvorgang bestimmt die Wahrscheinlichkeiten eines Zustandsüberganges $s \rightarrow s'$ unter der Bedingung, dass eine bestimmte Antwort (Aktion) a gegeben wurde. Diese werden in der Matrix $\mathbf{T} : \mathbf{S} \times \mathbf{A}$ mit einer diskreten WDF über der Zustandsmenge $\mathcal{S}$ gespeichert. Die Parameter von $\mathbf{T}$ werden im Rahmen dieser Arbeit durch einfaches Auszählen der Transitionen im Trainingskorpus bestimmt. Die trainierte Dialogstrategie kann durch den „Q-Learning“[1]-Algorithmus während des Betriebes verfeinert werden, welche im Rahmen dieser Arbeit nicht stattgefunden hat.

Die Berechnung der Belohnungsmatrix $\mathbf{R}$ kann entweder durch vorhandene Belohnungsdaten im Trainingskorpus oder, im Hinblick auf die fehlende Annotation im ATIS-Korpus, manuell erfolgen.

Berechnung von MDPs: Der Algorithmus besteht aus zwei Basisschritten, welche für alle Zustände $s \in \mathcal{S}$ wiederholt werden, bis keine weiteren Verbesserungen eintreten: In einem ersten Schritt werden in $V(s)$ die durschnittlichen Belohnungen

[1]Q-Learning ist eine verstärkende Lernstrategie (*Reinforcment Learning*), die eine Aktion-Wert-Funktion, welche die zu erwartenden Aktion a in einem Zustand s beschreibt [SB05]. Dieses Verfahren findet häufig Anwendung in der Spieltheorie.

(*Rewards*) bestimmt[2], welche aus der Berechnungsvorschrift (*policy*) π resultieren,

$$V_\pi(s) = R(s, \pi_t(s)) + \gamma \sum_{s' \in S} T(s, \pi_t(s), s')\, V_\pi(s'). \tag{5.2}$$

Eine neue Berechnungsvorschrift $\pi(s) = a$ kann mithilfe Gleichung 5.2 und Gleichung 5.3 bestimmt werden:

$$\pi_v(s) = \underset{a}{\text{argmax}} \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s')\, V_\pi(s') \right]. \tag{5.3}$$

Mit Gleichung 5.3 kann eine neue Wertfunktion $V(s)$ nach Gleichung 5.2 berechnet werden. Diese Schritte werden so lange wiederholt, bis $V(s)$ eine Schwelle (*Threshold*) t unterschreitet.

5.1.2 Partiell-beobachtete Entscheidungsprozesse nach Markov

Der POMDP unterscheiden sich von den MDP, dadurch dass die Zustände $s_0, \ldots, s_N$ nicht beobachtbar sind; dem System steht nur die bekannte Beobachtungsmatrix $\mathbf{O}$ zur Verfügung. In Abbildung 5.2 ist der Zusammenhang von POMDP zu den MDP und der HMM zu den Markov-Modellen dargestellt. Die Erweiterung des Markov-Modells erlaubt die Modellierung von Unsicherheiten im Eingangssignal. Ähnlich zum HMM versucht das System anhand der Beobachtung $\mathbf{o}$ (z. B. semantische Slots) auf den am besten passenden Zustand s_i zu schließen. Der POMDP besteht aus einem 6-Tupel $P = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbf{B}, \mathbf{T}, \mathbf{R})$ mit

- eine endliche Menge an **Zuständen** $\mathcal{S}$,
- eine endliche Menge an **Aktionen** $\mathcal{A}$,
- eine endliche Menge an **Beobachtungen** $\mathcal{O}$,
- eine **Belief-Matrix** B ,
- eine **Zustandsübergangsmatrix** $\mathbf{T} = [t_i j]_(\mathcal{S} \times \mathcal{S})$,
- eine **Belohnungsmatrix** für jede Dialogaktion $\mathbf{R} = [r_i j]_(\mathcal{S} \times \mathcal{A} \times \mathcal{S})$.

[2]Dieser Schritt wird mithilfe der dynamischen Programmierung berechnet und als Rückwärtsinduktion bezeichnet [Bel57].

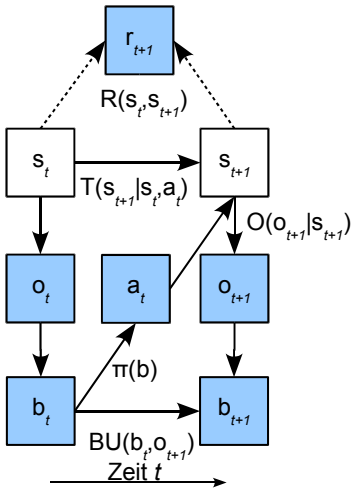


Abbildung 5.4: Teilweise beobachtbarer Markov Decision Process, realisiert als Graphisches Modell.

Aktualisierung des Belief-Vektors Der Belief-Vektor ist ein interner Zustand des Systems. Er beinhaltet die Wahrscheinlichkeiten jedes Zustandes s zu einem bestimmten Zeitpunkt t. Weil die realen Zustände nicht beobachtbar sind, wird der Belief-Vektor anhand von Beobachtungen o und Aktionen a berechnet. Der erste Belief-Vektor wird aus einem Startzustand s_0 definiert, die folgenden Zustände (*belief states*) werden mit

$$\begin{aligned} b_{t+1}(s_{t+1}) &= P(s_{t+1}|o_t, a_t, b_t) \\ &= \frac{P(o_t|s_{t+1}, a_t, b_t) \cdot P(s_{t+1}|a_t, b_t)}{P(o_t|a_t, b_t)} \\ &= \frac{P(o_t|s_{t+1}, a_t) \cdot \sum_{s \in \mathcal{S}} P(s_{t+1}|a_t, b_t, s_t) \cdot P(s_t|a_t, b_t)}{P(o_t|a_t, b_t)} \\ &\approx \frac{O(s_{t+1}, a_t, o_t) \cdot \sum_{s \in \mathcal{S}} T(s_{t+1}, a_t, s_t) \cdot b(s_t)}{P(o_t|a_t, b_t)} \end{aligned} \tag{5.4}$$

iterativ berechnet [KLC98].

Berechnung der Belohnungsfunktion r Zu jedem Zeitpunkt t erhält das System eine Belohnung, welche sich aus der ausgeführten Aktion a und der Berechnungsvorschrift b bestimmen lässt. Die über einen Zeitraum kumulierte Belohnungsmatrix wird bestimmt, mit

$$\mathbf{R} = \sum_{t=0}^{\infty} k^t \cdot r(\mathbf{b}_t, a_t) = \sum_{t=0}^{\infty} k^t \sum_{s \in S} b_t(s) \cdot r(s_t, a_t), \tag{5.5}$$

wobei k einen geometrischen Discountfaktor darstellt.

Berechnungsvorschrift π durch Bestimmung der Maxima Während die Berechnungsvorschrift (*policy*) π in dem MDP die Zustände mit den Aktionen a des Dialogmanagements verbindet, beschreibt $\hat{\pi}$ in den POMDP die Verbindung des Schätzvektors $\mathbf{b}$ (siehe Gleichung 5.4) mit den Aktionen a (siehe Abbildung 5.4), sodass die Rückgabe maximiert wird. Der dominierende Unterschied zu dem MDP ist, dass der Schätzvektor $\mathbf{b}$ kontinuierlich ist und nahezu beliebig viele Aktionen a auslösen kann.

Die optimale Berechnungsvorschrift ist stückweise linear und konvex [KLC95]. Wie in Abbildung 5.5 dargestellt ist, wird aus den Schätzvektoren $\mathbf{v_i}(s)$, der am besten passende Wert bestimmt, mit

$$V^{\hat{\pi}}(b) = \max_{v_i \in V}(v_i \cdot \mathbf{b}). \tag{5.6}$$

Mit dem Argument des Maximums kann daraus für die Berechnungsvorschrift π die nächste Dialogaktion $a_i \in \mathcal{A}$ mit

$$\hat{\pi}(b) = a(\underset{i}{\text{argmax}}(v_i \cdot \mathbf{b})), \tag{5.7}$$

bestimmt werden.

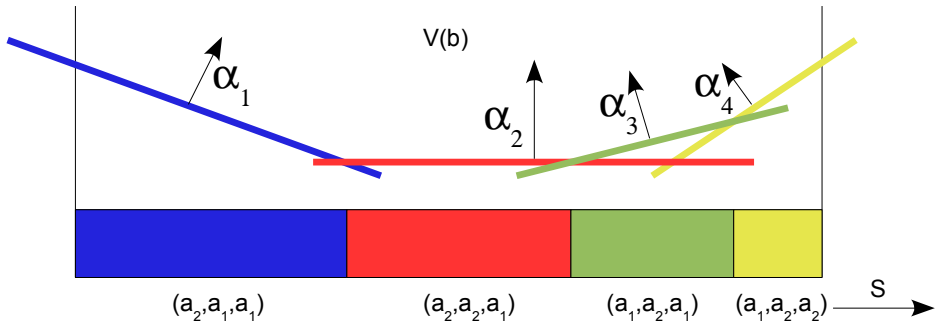


Abbildung 5.5: Die optimale Wertfunktion $V(b)$ des Schätzvektors $\mathbf{b}$ ist stückweise linear und konvex. Über dem Maximum wird jeweils ein Teilbereich von $V(\mathbf{b})$ ausgewählt und eine Berechnungsvorschrift π^* mit zugehöriger Aktion a berechnet.

Approximationsalgorithmus Mithilfe des „Successive Approximations of the Reachable Space under Optimal Policies“ (SARSOP)-Approximationsalgorithmus [KHL08] können im Gegensatz zur vollständigen Berechnung der Dialogvorschrift mit Sondik's Algorithmus [SS73] mehr Zustände in angemessener Zeit berechnet werden. Dieser Algorithmus hat sich zum Standard für die Berechnung von POMDP-Modellen entwickelt und wird auch im Rahmen dieser Arbeit verwendet (siehe Algorithmus 3).

Algorithmus 3 Rekursive Berechnung der Wertfunktion im POMDP Dialogmanagementmodell.

for all $b \in \mathcal{B}$ **do**
  $V_0 = 0$
end for
$t = 0$
repeat
  $t = t + 1$
  for all $b \in \mathcal{B}$ **do**
    $V_t(b) = \max_{a \in A} \left[r(b,a) + \gamma \sum_{b' \in B} \tau(b,a,b')\, V_{t-1}(b') \right]$
  end for
until $|V_t(b) - V_{t-1}| < \epsilon$, $\forall b \in \mathcal{B}$

Berechnung der Dialogpolicy mithilfe SARSOP-Algorithmus [KHL08] Weitere Approximationsalgorithmen werden hier nicht verwendet, da das POMDP im Rahmen dieser Arbeit als ein Referenzmodell betrachtet wird.

5.2 Hidden-Markov-Modell für das Dialogmanagement

Für mixed-initative Dialoge werden Benutzerziele g (siehe Abschnitt 3.2.4) abgeschätzt, um daraus eine SQL-Datenbankabfrage zu ermöglichen. In Abbildung 5.6 wird das GM eines mixed-initative Dialogmanagementsystems mit beobachtbarem Prolog s_0 und Epilog s_T gezeigt. Die verborgenen Zustände $s_1, \ldots, s_t, s_{t+1}, \ldots, s_T$ des ausrollbaren Chunks [BZ02], erlauben zu jedem Zeitschritt t eine Beobachtung o_t semantischer Slots. Dadurch kann der Dialog von $N(d) = 3$ (aufgrund der Definition des GMs) bis zu einer Dialoglänge T variieren.

Analog zum Belief-State b des POMDP-Modells (siehe Abschnitt 5.1) werden die Beobachtungen o mithilfe eines „Future Flags“ f in vergangene $o_{0:t-1}$, aktuelle o_t und zukünftige Beobachtungen $o_{t+1:T}$ unterteilt. Der vergangene Teil $o_0, \ldots, o_{t-1}$ wächst mit jeder vom Benutzer gegebenen Antwort o_t. Die aktuelle Beobachtung o_t repräsentiert die in einem Zeitschlitz t gegebene Benutzerantwort. Der zukünftige Teil schätzt die fehlenden semantischen Slots ab, ausgehend von der vergangenen und der aktuellen Beobachtung $o_{0:t}$ bis zum nächstgelegenen Benutzerziel g.

Das HMD besteht aus einem 5-Tupel $H = (\mathcal{S}, \mathbf{T}, \mathcal{O}, \mathcal{A}, \mathcal{F})$ mit

- eine endlichen Menge an **Zuständen** $\mathcal{S}$,
- der **Zustandsübergangsmatrix** $\mathbf{T}(s, a, s')$, die eine endliche Menge an Zustandsübergangswahrscheinlichkeiten ist,
- einer endlichen Menge an **Beobachtungen** $\mathcal{O} = \{o_1, \ldots, o_n\}$,
- einer endlichen Menge an **Aktionen** $\mathcal{A}$ und

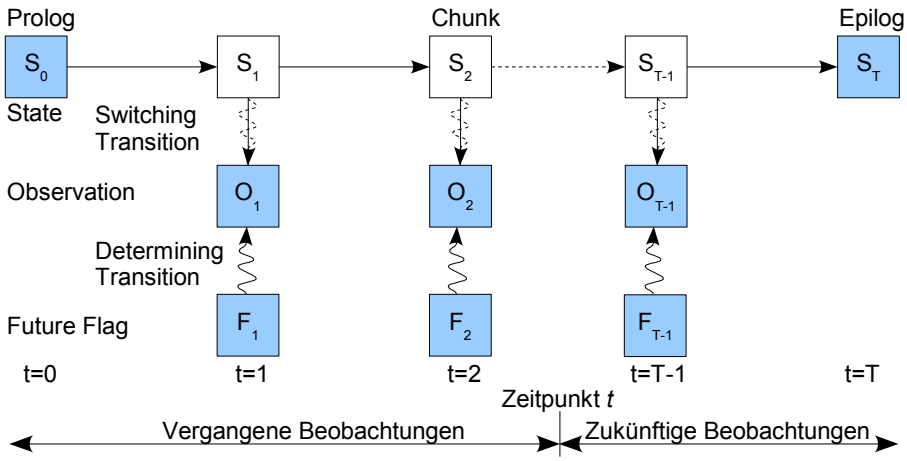


Abbildung 5.6: Das Graphische Modell kann mithilfe des Future Flags vergangene, gegenwärtige und geschätzte zukünftige semantische Informationen beobachten. Diese Aufteilung erlaubt die Verwendung verschiedener Wahrscheinlichkeitsmatrizen.

- einer **Zukunftsflag** F, welche deterministisch über vergangene und zukünftige Beobachtungen entscheidet.

5.2.1 Modellierung

Die Zustandsübergangsmatrix $\mathbf{T}$ ist markoffsch, d. h. die Berechnung erfolgt nur mit der Information des Vorgängerzustands. Die Beobachtungen aus dem vergangenen Teil werden mit den Zuständen s entsprechend der Beobachtungsmatrix $\mathbf{O}_p$ verbunden. Die Beobachtungsmatrix $\mathbf{T}_p$ wird mithilfe von GMTK aus dem Trainingskorpus $\mathcal{S}_{\text{train}}$ bestimmt.

Da zum Zeitpunkt t keine Informationen über zukünftige Beobachtungen $o_{i>t}$ vorliegen, werden sie gleich verteilt $\mathcal{P}(o_i) = {}^1/_{|o_f|}$ der Beobachtungsmatrix $\mathbf{O}$ hinzugefügt. Somit kann jeder verborgene Zustand s eine Beobachtung $o_{i>t}$ mit der gleichen Wahrscheinlichkeit eingehen.

Der Vorteil dieser Modellierung liegt darin, dass zu jedem Zeitpunkt t ein gültiges HMD vorliegt und für jedes t mithilfe des Viterbi-Algorithmus decodiert werden kann. Zum Zeitpunkt $t = 0$ liegen keine Beobachtungen O vor, d. h. die Viterbi-Decodierung erfolgt nur durch die Übergangsmatrix $\mathbf{T}$ und mit dem geschätzten Benutzerziel g_1, welches nach $\mathcal{S}_{\text{train}}$ als Erstes erreicht werden soll. Mithilfe von Viterbi wird zum Zeitpunkt t der optimale Pfad durch den Dialog d berechnet; durch die Extrahierung der nächsten Beobachtung o_{t+1} wird eine weitere Frage an den Benutzer des Systems gestellt. Die Anzahl möglicher Beobachtungen o zum Zeitpunkt $t + 1$ wird durch die Bedingung beschränkt, dass durch die Transition nur ein semantischer Slot hinzugefügt werden darf (siehe Abbildung 5.7). Somit ist gewähr-

leistet, dass das System nur eine neue Information von dem Benutzer verlangt. Dem Dialogmanager müssen dabei nur minimale Veränderungen hinzugefügt werden.

Es bleibt die Frage zu klären, wie ein Benutzerziel (*user goal*) abgeschätzt werden kann. Im ATIS-Korpus werden folgende Regeln für Benutzerziele definiert:

- Ein Benutzerziel enthält alle semantischen Slots, die bislang beobachtet werden konnten (kumulative Beobachtung).
- Ein Benutzerziel enthält eine minimale Reihe an Beobachtungen, woraus eine SQL-Datenbankanfrage erfolgt (regelbasierte Definition).

Dadurch ist gewährleistet, dass das Benutzerziel nicht überschätzt wird. In Anlehnung an [HNR68] kann es also nie vorkommen, dass das reale Benutzerziel weniger semantische Slots enthält als das geschätzte.

5.2.2 Suche des besten Dialogpfades

Die verborgenen Zustände $s \in \mathcal{S}$ des HMD basieren auf den in Abschnitt 3.2 eingeführten semantischen Slots. Im Rahmen dieser Arbeit besteht $\mathcal{S}$ aus $N(\mathcal{S}) = 2^8 = 256$ Zuständen, die eine spezielle Kombination der semantischen Slots spezifizieren. Die Anzahl der Beobachtungen $o \in \mathcal{O}$ ist $N(\mathcal{O}) = N(\mathcal{S})$. Sie sind binär codiert und stammen aus der semantischen Decodierung (siehe Kapitel 4).

Die Grundidee besteht nun darin, dass aufgrund der Eingaben aus der semantischen Decodierung das System die bestmögliche Sequenz dynamisch aus den möglichen Dialogzuständen s_d auswählt. Dazu eignen sich die Verfahren aus der dynamischen Programmierung und der Decodierung mithilfe des Viterbi-Algorithmus [Vit67].

In Abbildung 5.7 werden alle Dialogpfade in einem Trellisdiagramm dargestellt. Durch die Restriktion, dass keine Informationen verloren gehen können, erhöht sich der Dialogzustand s_i monoton mit jedem Zeitschritt t.

Neuabschätzung der semantischen Slots

Der Viterbi-Algorithmus sucht und findet den besten Dialogpfad mit jeder neuen Beobachtung o_t zu jedem Zeitpunkt t. Das heißt, dass für die vergangenen Beobachtungen $o_1, \ldots, o_{t-1}$ die semantischen Slots $s_1, \ldots, s_{t-1}$ mit jeder neuen Information neu berechnet werden können. Dies führt unter Umständen dazu, dass verschieden gut erkannte Beobachtungen in o_t nach mehreren Zeitschritten $t + x$ durch den am besten passenden Dialogpfad eindeutig bestimmt werden.

Durch die Verwendung zweier Transitionsmatrizen $\mathbf{T}_p$ und $\mathbf{T}_f$ werden für den zukünftigen Teil der Beobachtungen verschiedene Restriktionen definiert:

- Nur ein semantischer Slot s_i soll pro Zeitschritt t hinzugefügt werden.

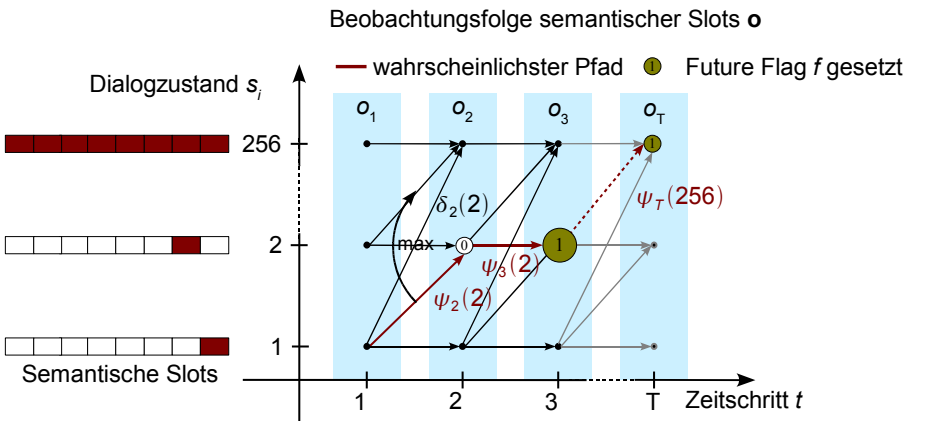


Abbildung 5.7: Trellisdiagramm eines GM-basierten Dialogmanagementmodells mit dessen zu befüllenden semantischen Slots. Der beste Pfad wird mithilfe des Viterbi-Algorithmus decodiert. Der geschätzte Pfad $t + 1$ bis zum Benutzerziel T wird mit dem Future Flag f angezeigt.

- Mit dem Erreichen eines bekannten Benutzerziels g wird automatisch der Endzustand des Dialoges erreicht.

- Der Benutzer arbeitet kooperativ, d. h. es sind keine Fehlereingaben zu erwarten.

Für $\mathbf{T}_p$ werden mithilfe der Addition einer Einheitsmatrix $\mathbb{1}$ Wiederholungen der semantischen Slots erlaubt. Die Matrix $\mathbf{T}_f$ bleibt unverändert. Die Transitionen werden nach den in Kapitel 3 beschriebenen Verfahren aus dem Trainingskorpus ermittelt.

Auswirkung und Erkennung von Fehlern

Zudem erlaubt diese Modellierung, mögliche Fehler in der semantischen Decodierung (siehe Kapitel 4) oder der Spracherkennung zu korrigieren. Konkret bedeutet dies, dass unter Umständen nicht immer der semantische Slot mit der höchsten Konfidenz ausgewählt wird.

Ein Dialogmanager sucht stets den besten Pfad zu einem Benutzerziel (*user goal*). Aufgrund dessen kann es vorkommen, dass nicht zu jedem Zeitpunkt t der am besten erkannte semantische Slot ausgewählt wird. In diesen Fällen werden vom Dialogmanagementsystem Fehler in der semantischen Decodierung erkannt und eine Korrektur dieser vollzogen. Vom System werden dann gezielte Dialogaktionen $a \in \mathcal{A}$ in jedem Zeitschritt t zur nochmaligen Abfrage des semantischen Slots ausgelöst.

5.2.3 Arbeitsweise des Dialogmanagementmodells

Zu jedem Zeitpunkt t bringt der Benutzer neue semantische Slots als Beobachtung o in das Modell ein. Das Dialogmodell nimmt ständig Beobachtungen $o \in \mathcal{O}$ auf und mit dem stetig zunehmenden Informationsgewinn wird das nächstgelegene Benutzerziel (*user goal*) geschätzt.

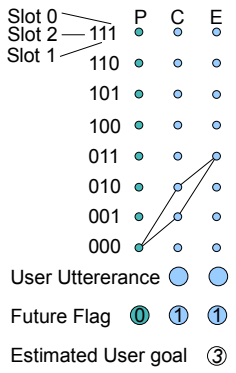
Mit dem Erreichen eines Benutzerziels ist der Dialog aber nicht zwangsläufig zu Ende; der Benutzer wird entweder nach einem neuen Dialogziel gefragt, oder es werden ihm die Daten aus der Datenbankabfrage zur Verfügung gestellt. Durch die Definition von Zwischenzielen wird eine robuste Dialogführung auch mit Fehlervorkommen in der semantischen Decodierung gewährleistet.

Im folgenden Beispiel soll die Arbeitsweise des HMD-Modells verdeutlicht werden. Das Modell beinhaltet drei semantische Slots: Abflugsort (*Origin*), Zielort (*Destination*) und Datum (*flight date*). Die Benutzerziele werden manuell definiert, sie ermöglichen eine SQL-Datenbankabfrage. Im trivialen Beispiel wäre eine Datenbankabfrage über einen Flug mit dem Abflugs- und Zielort möglich. Dieser Zustand $s = 0\,1\,1$ wird dann als ein Benutzerziel definiert werden.

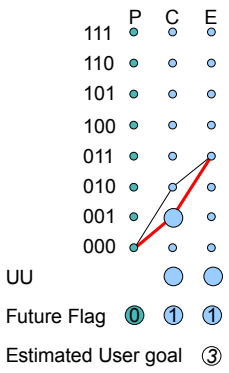
Initialisierung: Zu Beginn des Dialogs findet keine Beobachtung o statt. Dennoch kann das System mögliche Benutzerziele anvisieren: Das minimale Benutzerziel ist $s = 0\,1\,1$ (An- und Abflugort). Aus dem Startzustand $s = 0\,0\,0$ kann das erste Benutzerziel über zwei verschiedene Wege in zwei Zeitschritten erreicht werden (siehe Abbildung 5.8a). Mit dem Viterbi-Algorithmus wird der bestmögliche Weg zum anvisierten Benutzerziel berechnet. Um dieses Ziel zu erreichen, muss $s = 0\,0\,1$ mit einer gezielten Frage befüllt werden. Die korrespondierende Aktion a_1 ist demzufolge „Frage nach Slot 0“.

Erster Schritt: Möglicherweise wird aufgrund einer Fehlerkennung oder durch eine falsche Benutzereingabe Slot 2 ($s = 1\,0\,0$) decodiert. Zuerst wird wieder das nächstgelegene Benutzerziel berechnet. Im konkreten Fall wird das Benutzerziel $s = 1\,1\,1$ anvisiert. Die Länge der benötigten Dialogschritte wird berechnet; es existieren eine vergangene und eine gegenwärtige Beobachtung. Mithilfe zweier zukünftiger Beobachtungen kann das Benutzerziel erreicht werden. Dies führt zu einer Gesamtlänge von $N(d_s) = 4$ Dialogschritten unter der Bedingung, dass zu jedem Zeitschritt t nur eine Frage an den Benutzer gestellt werden kann (siehe Abbildung 5.8c). Nach Ausführung des Viterbi-Algorithmus (siehe Abbildung 5.8d) ist der wahrscheinlichste aktuelle Zustand nun $s = 1\,0\,0$, und der zukünftige Pfad führt über $s = 1\,1\,0$ zum Benutzerziel. Durch Subtraktion $1\,1\,0 - 1\,0\,0 = 0\,1\,0$ erhält man die nächste Aktion a_2: „Frage nach Slot 1“.

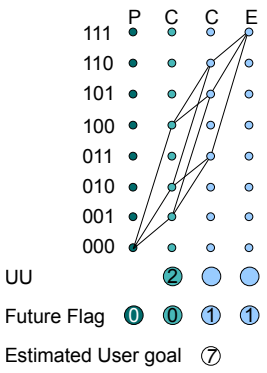
Zweiter Schritt: In diesem Schritt erhält das System vom Benutzer den erwarteten semantischen Slot. Aus den kumulativen Beobachtungen resultieren Slot 1 und



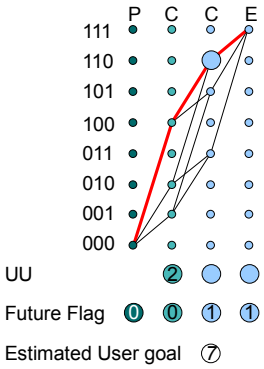
(a) Schritt: 1.1



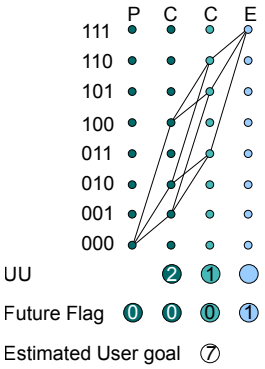
(b) Schritt: 1.2



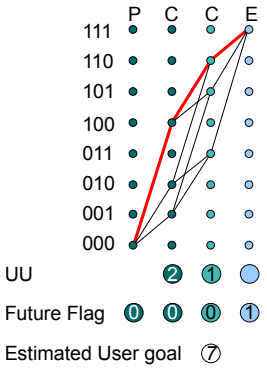
(c) Schritt: 2.1



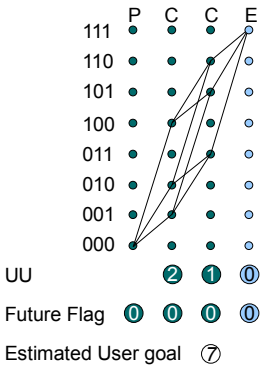
(d) Schritt: 2.2

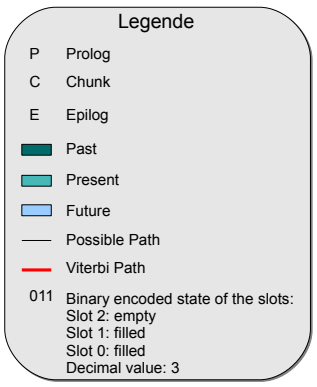


(e) Schritt: 3.1



(f) Schritt: 3.2





(g) Schritt: 4.1

Abbildung 5.8: Beispiel einer Dialog-Decodierung mithilfe des GM-basierten Dialogmanagers.

Slot 2. Daraus ergibt sich, dass das nächstgelegene Benutzerziel (Zustand $s = 1\,1\,1$) nicht verändert werden muss. Die Differenz vom jetzigen Zustand zum Benutzerziel beträgt „Eins“. Es muss also mit einer gewissen Wahrscheinlichkeit ein zusätzlicher Slot abgefragt werden. Die vergangenen Beobachtungen sind: $s_0 = 0\,0\,0$, $s_1 = 1\,0\,0$ und die aktuelle Beobachtung ist mit großer Wahrscheinlichkeit: $s_2 = 1\,1\,0$. Die Gesamtlänge ist nach wie vor $N(d) = 4$. Aus den möglichen Pfaden (siehe Abbildung 5.8e) wird mithilfe des Viterbi-Algorithmus der beste Pfad ausgewählt (siehe Abbildung 5.8f). Durch Subtraktion $1\,1\,1 - 1\,1\,0 = 0\,0\,1$ erhält man die nächste Aktion a_3: „Frage nach Slot 0“.

Dritter Schritt: Aus der Benutzeräußerung wird Slot 0 decodiert. Wie in jedem Schritt überprüft das Dialogmanagementsystem, ob das Benutzerziel erreicht wurde. Dazu wird dem letzten „aktuellen Zustand“ der aktuell beobachtete Slot addiert. In diesem Beispiel wird mit dem Zustand $s_2 = 1\,1\,0$ und dem semantischen Slot $0\,0\,1$ das Benutzerziel $s_3 = 1\,1\,1$ erreicht. Dadurch kann eine SQL-Datenbankabfrage erfolgen und dessen Ergebnis kann dem Benutzer präsentiert werden. In einem System mit mehreren Slots wird der Benutzer zuvor gefragt, ob er ein weiteres Benutzerziel anstrebt. Dann wird der Dialog fortgesetzt, ansonsten ist der Dialog hier zu Ende.

5.3 Experimente

Das neue Verfahren wird mit dem in Abschnitt 5.1 eingeführten Referenzsystem auf Basis der automatischen Flugauskunft ATIS (siehe Kapitel 3) verglichen. Um ein aussagekräftiges und vergleichbares Ergebnis zu erhalten, wurden die Eingaben der semantischen Slots simuliert. Die Simulationsumgebung existiert in einem *C++* „Framework“ und kommuniziert mit den Funktionen des Dialogmanagers. Dazu werden verschiedene, im folgenden erläuterte Tests durchgeführt.

Manuelle Tests Mithilfe der implementierten Benutzeroberfläche werden die semantischen Informationen vom Benutzer ausgewählt und den zu testenden Systemen übermittelt. Der Benutzer kann nach Belieben den Systemen (HMD,POMDP) semantische Informationen übergeben. Das System versucht dann, den Benutzer mithilfe der geschätzten Benutzerziele (*user goals*) durch den Dialog zu führen. Damit können situative Tests durchgeführt werden. Für eine exzessive und objektive Evaluierung der Systeme sind jedoch automatische Testverfahren notwendig.

Automatisierte Tests Das folgende Verfahren erläutert die automatisierte Testumgebung, welche zu reproduzierbaren Ergebnissen führt. Für diesen Zweck wird ein Modell des Benutzers erstellt.

5.3.1 Benutzermodell

Ein einfaches Benutzermodell besteht aus zwei Teilbereichen:

- Eingabe von semantischen Informationen,
- Reaktion auf die vom System gestellte Frage.

Ein Benutzerziel (*user goal*) g beschreibt ein bestimmtes Zwischenziel im Dialogverlauf. Gültige Zwischenziele sind Teilmengen aller semantischen Slots $g \subset S$. Jede Variation von g erhält dabei genügend Information, um dem Benutzer eine konkrete Frage stellen zu können. Ein Benutzerziel g aus dem ATIS-Korpus erfüllt z. B. folgende logische Bedingung:

$$(\text{Abflugort} \wedge \text{Ziel} \wedge (\neg\text{Preis})) \vee (\text{Abflugort} \wedge \text{Ziel} \wedge \text{Preis} \wedge \text{HinRück}) \tag{5.8}$$

Der zweite Teilbereich erfasst die Klassifikation der Systemantworten und die Simulation der Reaktion des Benutzers.

Für diesen Zweck wurde ein „gutartiger" Benutzer implementiert, welcher auf die vom System gestellte Frage die richtige Antwort liefert. Die „Gutartigkeit" kann, je nach Test, variiert werden. Für den Fall, dass das System eine Liste von Eingabemöglichkeiten (z. B. Städtenamen) liefert, wird eine zufällige Auswahl aus dieser simuliert. Mit der simulierten Benutzerantwort wird ein weiterer semantischer Slot in Richtung Benutzerziel (*user goal*) befüllt.

Annahmen Eine Annahme im Benutzermodell ist, dass das Benutzerziel während eines Dialoges unverändert bleibt. Nach [You06] treten veränderte Benutzerziele in einem Dialog selten auf. Eine weitere Annahme ist, dass im Bereich der freien Auswahl von Benutzerzielen die WDF zum einen anhand einer Gleichverteilung und zum anderen basierend auf der statistischen Verteilung des ATIS-Trainingskorpus, benutzt wird (siehe Abbildung 5.9)

Die Annahmen müssen aufgrund der gering vorhandenen Mengen an Information im ATIS-Korpus getroffen werden. Ebenso basiert die Simulation unsicherer Spracheingaben auf Annahmen aufgrund nicht vorhandener Trainingsdaten (siehe Kapitel 3).

5.3.2 Simulation unsicherer Spracheingaben

Um unsichere Spracheingaben oder Fehler des Spracherkenners zu simulieren, werden die Eingaben in Form von semantischen Slots systematisch verrauscht (Verrauschungsprozess). Die Verrauschung findet nach der Benutzereingabe und vor dem zu evaluierenden Dialogmanagementsystem statt (siehe Abbildung 5.9). Der Prozess bewirkt, dass die Benutzereingaben realitätsnah, also nicht eindeutig sind. Wenn eine gestörte Benutzereingabe o_x vorliegt, wählt der „Verrauschungsprozess"

zufällig einen semantischen Slot s_i und übergibt ihn an das Dialogmanagementsystem.

Mit dem „Verrauschungsprozess“ wird das Dialogmanagementsystem auf die automatische Korrektur von Fehler E_{Sys} getestet. Die bewusste Veränderung der semantischen Decodierung wird hier als simulierte Fehlerrate E_{Sim} bezeichnet. Die Ausgaben des „Verrauschungsprozesses“ sind die Beobachtungen der zu evaluierenden Systeme. E_{Sys} wird als Fehlerrate des Dialogmanagementsystems für die nachfolgende Darstellung der Ergebnisse verwendet.

Die Wahrscheinlichkeit des „Verrauschungsprozesses“ wird als Fehlerrate E_{Sim}, wie sie aus der semantischen Decodierung stammen könnte, dem Dialogmanagementsystem weitergereicht. Der aktuelle Fehlerbetrag E_{Sys} ist das Resultat der gemessenen Beobachtungen an semantischen Slots.

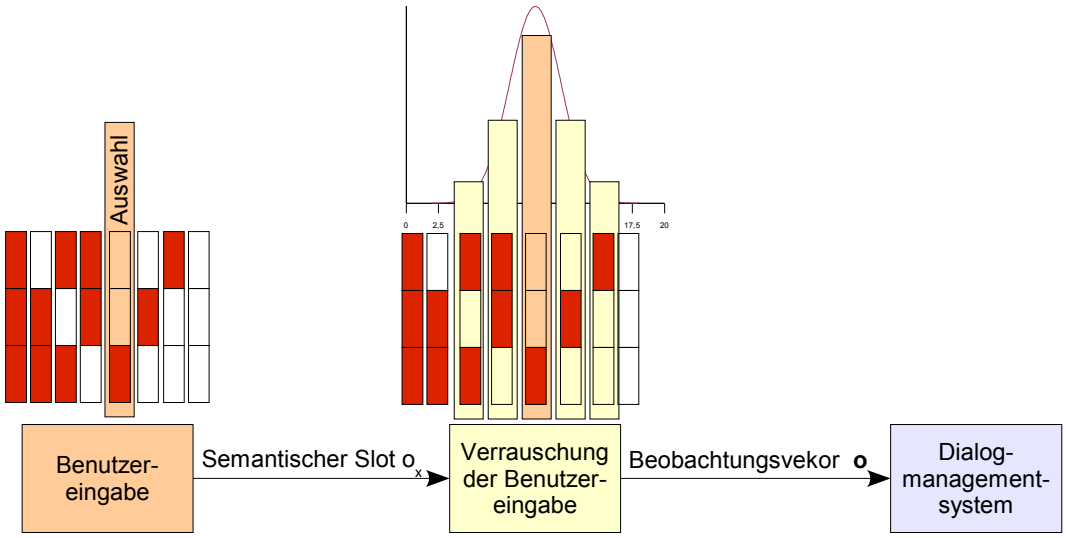


Abbildung 5.9: Über die manuelle oder automatisierte Benutzereingabe wird der Eingabewert o_x durch die semantisch am nächsten liegenden Nachbarn mithilfe der Normalverteilung in einen Beobachtungsvektor o übergeführt. Der „Verrauschungsprozess“ simuliert unsichere Benutzereingaben, welche beispielsweise aus dem Spracherkenner resultieren können.

5.4 Evaluierung

In diesem Abschnitt wird beschrieben, wie die wichtigsten Funktionen der GM-basierten Dialogmanagementprozesse getestet werden. Diese sind:

die Fragefunktion des Systems (*ask function*): diese kontrolliert den Dialogfluss und legt die Statistik an,

die Benutzerzielfunktion (*user goal generation function*): diese wählt am Beginn eines Dialoges ein zufälliges Benutzerziel, simuliert die Benutzerauswahl und dessen freie Auswahlmöglichkeit und

die Verrauschungprozessfunktion (*noise process function*): sie simuliert die Benutzereingabe durch unvollendete Spracheingaben.

5.4.1 Stabilitätstest

Das Ziel des Stabilitätstests ist es, zu zeigen, dass der im Rahmen dieser Arbeit entwickelte Dialogmanager prinzipiell arbeitet, auch mit verrauschten Benutzereingaben. Dabei wird eine einheitliche WDF $P(s_i) = {}^1/_{N(s)}$ über allen Benutzerzielen (*user goals*) verwendet. Mithilfe der automatisierten Tests wird die erfolgreiche Terminierung des Systems getestet.

Ein weiteres Ziel ist die Ermittlung der benötigten Dialogschritte $N(d_s)$. Im Vergleich zu den Dialogen im ATIS-Korpus wird hier eine Überlänge festgestellt, wenn vom System mehr Benutzerziele als nötig vom Benutzer abgefragt werden. Der Benutzer klassifiziert diese Systeme als nicht zielgerichtet.

Für den Stabilitätstest wurden alle getesteten $N(d) = 200$ automatisch generierten Dialoge erfolgreich zu Ende geführt, sodass eine SQL-Datenbankabfrage an das System simuliert werden konnte. Das Ergebnis zeigt, dass das HMD-System unter Störeinfluss (Spracherkennung, semantische Decodierung) sehr robust ist. Unendliche Dialogabläufe ergeben sich nur, wenn der Benutzer unkooperativ ist und immer denselben semantischen Slot beantwortet.

Ein weiteres Ergebnis ist, dass im Vergleich der annotierten Dialoge aus dem ATIS-Korpus das HMD-System in $66,5\%$ der automatisch generierten Dialoge die optimalen Benutzerziele durchläuft und in $33,5\%$ der getesteten Dialoge der Dialog mit mehr Dialogschritten als notwendig zu Ende geführt werden. Dadurch ist der Benutzer mit mehr Fragen als für eine SQL-Datenbankabfrage notwendig konfrontiert. Die durchschnittliche Dialoglänge beträgt $\varnothing N(d_s^{\text{system}}) = 8$ Dialogschritte, wohingegen im Durchschnitt $N(d_s^{\text{opti}}) = 6$ Dialogschritte ausreichend sind [SRWR09].

5.4.2 Automatische Fehlererkennung und Korrektur

Anstelle einer einheitlichen WDF (siehe Stabilitätstest) werden 20% der Gesamtdialoge aus dem ATIS-Korpus $\mathcal{S}_{\text{Train}} = 0,2 \cdot \mathcal{S}_{\text{Ges}}$ für ein stochastisches Benutzermodell $\mathcal{M}$ verwendet. Damit werden die freie Listenauswahl und das Benutzerziel statistisch angereichert. Die Verwendung einer korpusbasierten WDF ist für dieses Experiment wichtig, da das System nur jene Fehler erkennen und korrigieren kann, welche dem Trainingskorpus $\mathcal{S}_{\text{Train}}$ bekannt sind.

Die simulierte Fehlerrate aus dem „Verrauschungsprozess“ wird wie im Stabilitätstest auf $E_{\text{Sim}} = 5\%$ belassen. Nachfolgend werden die in Tabelle 5.1 verwende-

ten Definitionen für die Gegenüberstellung der verwendeten Modelle $\mathcal{M}_{\text{HMD}}$ und $\mathcal{M}_{\text{GMTK}}$ erläutert.

Die Klassifikation eines Dialogschrittes d_s wird als richtig gewertet, wenn das Dialogmanagementsystem die gleiche semantische Information decodiert wie auf dem fehlerfreien Eingangssignal,

$$d(s_{\text{in}}, s_{\text{dec}}) = \begin{cases} 1 & \text{wenn } s_{\text{in}} = s_{\text{dec}}, \\ 0 & \text{sonst.} \end{cases} \tag{5.9}$$

Ein Dialogschritt d_s wird als richtig gewertet, wenn das Eingangssignal „verrauscht“ ist und das Dialogmanagementsystem den Dialogschritt richtig klassifiziert,

$$d_{\text{korr}}(s_{\text{noisy}}, s_{\text{dec}}) = \begin{cases} 1 & \text{wenn } s_{\text{noisy}} = s_{\text{dec}}, \\ 0 & \text{sonst.} \end{cases} \tag{5.10}$$

Aus den Gleichungen 5.9 und 5.10 folgen die Definitionen analog zu Abschnitt 2.6: Richtig Positiv (TP), Falsch Positiv (FP), Richtig Negativ (TN) und Falsch Negativ (FN) sind wie folgt definiert

$$TP = \{d(s_{\text{in}}, s_{\text{dec}}) = 0\} \wedge \{d_{\text{korr}}(s_{\text{noisy}}, s_{\text{dec}}) = 0\} \tag{5.11}$$
$$FP = \{d(s_{\text{in}}, s_{\text{dec}}) = 1\} \wedge \{d_{\text{korr}}(s_{\text{noisy}}, s_{\text{dec}}) = 0\} \tag{5.12}$$
$$FN = \{d(s_{\text{in}}, s_{\text{dec}}) = 0\} \wedge \{d_{\text{korr}}(s_{\text{noisy}}, s_{\text{dec}}) = 1\} \tag{5.13}$$
$$TN = \{d(s_{\text{in}}, s_{\text{dec}}) = 1\} \wedge \{d_{\text{korr}}(s_{\text{noisy}}, s_{\text{dec}}) = 1\} \tag{5.14}$$

Damit wird eine Falsch Positiv Rate (FPR) und eine Falsch Negativ Rate (FNR) (siehe Kapitel 2) nach

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{5.15}$$
$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{5.16}$$

berechnet.

Die Fehlerkorrektur ist für den realen Einsatz von Bedeutung, da eine falsche Klassifikation beispielsweise ein falsches Flugticket lösen kann. Die Kriterien für einen falschen Dialog sind, dass ein semantischer Slot durch den „Verrauschungsprozess“ mit einem anderen ersetzt wurde. Die Kriterien für einen erfolgreich durchgeführten mit Fehlern behafteten Dialog sind, dass sich die decodierte Antwort von der Beobachtung unterscheidet. Nach einer Simulation von $N(d) = 1\,990$ Dialogen wurde folgendes Resultat erzielt.

Die Spezifität klassifiziert die Rate der „richtigen“ (nicht-verrauschten) Dialoge, die auch „richtig“ klassifiziert werden. Da beide Systeme mit unterschiedlichen

		HMD	POMDP
Anzahl der Dialoge $N(d)$		1 990	2 000
Benutzeräußerungen	$N(u)$	8 717	12 901
Durchschnittliche Dialoglänge $\varnothing N(d_s)$		$4,38$	$6,45$
Korrekte Klassifikationsrate d_{corr}		1 761	1 688
F-Score TP		115	56
F-Score FP		114	121
F-Score FN		179	256
F-Score TN		1 582	1 567
Sensitivity (SES)	[%]	$0,57$	$0,38$
Specifity (SPC)	[%]	$0,81$	$0,81$
False positiv rate (FPR) [%]		$0,07$	$0,07$
False negativ rate (FNR) [%]		$0,61$	$0,82$
Relevanz (PPV) [%]		$0,50$	$0,32$
Segreganz (NPV) [%]		$0,90$	$0,86$
False Discovery Rate (FDR) [%]		$0,15$	$0,19$
True Discovery Rate $(1 - \text{FDR})$ [%]		$0,85$	$0,81$

Tabelle 5.1: Fehlererkennungsraten von HMD und POMDP Dialogmanagementmodellen im Vergleich [SMS+09].

„Verrauschungsprozessen“ arbeiten, wird zu Vergleichszwecken die Spezifität in beiden Modellen auf $\text{SPC}_{\text{POMDP}} = \text{SPC}_{\text{HMD}} = 0,81$ gesetzt.

Die **Sensitivität** beschreibt die Rate fehlerhaften (verrauschter) Dialogen, die erkannt werden. Gegenüber dem POMDP-System ($\text{SES}_{\text{POMDP}} = 0,38$) kann mit dem HMD-System ($\text{SES}_{\text{HMD}} = 0,57$) die Sensitivität um etwa 50% erhöht werden.

Die erzielten Ergebnisse werden in Abbildung 5.10 mithilfe der „Empfänger Operationscharakteristik“ (*Receiver Operating Characteristic*) (ROC) visualisiert. Beide Verfahren übertreffen ein lineares Dialogmanagementmodell (ground truth) und eignen sich zur Erkennung (Korrektur) von Fehlern während des Dialogverlaufes. In einem Vergleich mit gleicher Spezifität ($SPC = 0,81$) schneidet das HMD-System gegenüber dem POMDP-System in der Sensitivität (SES) relativ um ca. 50% besser ab.

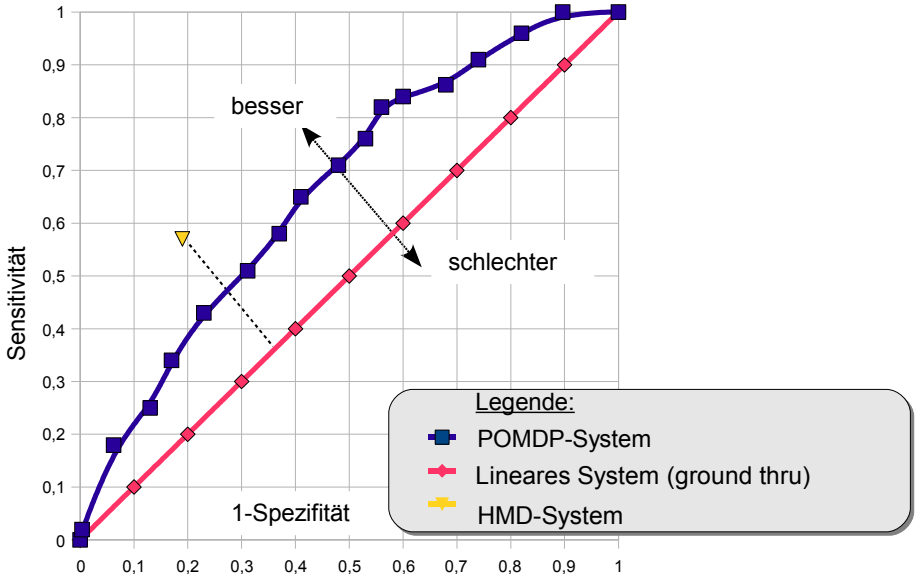


Abbildung 5.10: ROC-Diagramm: Vergleich auf Basis der Dialogmanagement-modelle (HMD und POMDP) auf Basis des ATIS-Szenarios.

5.5 Zusammenfassung des Kapitels

In diesem Kapitel wurden zwei **Mixed-Initiative**-Dialogmanagementsystem vorgestellt, implementiert und evaluiert. Im POMDP-System sind zur Laufzeit nur „einfache“ Tabellenabfragen und Multiplikationen nötig. Beim Entwurf der Dialogvorschrift (*policy*) ist ein extrem hoher (NP-Vollständig [HLR08]) Rechenaufwand nötig.

Das HMD-System berechnet die Dialogvorschrift direkt zur Laufzeit und bleibt somit für viele Anwendungsbereiche flexibel einsetzbar. Da es mit einem GM mo-

delliert wurde, ist es erweiterbar und es ist ein Zusammenschluss mit den bereits eingeführten Verarbeitungsstufen (Spracherkenner und semantische Decodierung) möglich. Damit kann das HMD-System Fehler aus der semantischen Decodierung erkennen und korrigieren. In einem Experiment wurde gezeigt, dass mit dem HMD-System verglichen zum POMDP-System tatsächlich eine bessere Erkennung von Fehlern aus den vorigen Verarbeitungsstufen möglich ist.

.

6

Verteilte Modellierung des Dialogmanagements

In diesem Kapitel werden die stochastischen und regelbasierten Methoden zur Beschreibung eines Dialogmanagementsystems (siehe Kapitel 5) in einem hybriden multi-modalen „Framework“ eingesetzt. Im nächsten Abschnitt erfolgt die einheitliche Beschreibung aller Komponenten des Dialogs mit GMs. In Abschnitt 6.3 werden die einzelnen Verarbeitungsstufen aufgrund der Parallelisierungsmöglichkeit in Agenten auf einem verteilten Agentensystem realisiert. Die Architektur, Implementierung und Evaluierung wurde bereits in einer Veröffentlichung vorgestellt und etabliert. [SSRW09].

6.1 Graphentheoretische Modellierung eines Multiagenten Dialogmanagementsystems

Nach [ZMK09] wird das POMDP-Modell (siehe Kapitel 5) in ein multiagenten Framework eingebettet. Dazu existieren eine Reihe von möglichen Zuständen $\mathcal{S}$, wobei der aktuelle Zustand $s \in \mathcal{S}$ nicht direkt durch einen Agenten beobachtet werden kann. Jeder Agent j verarbeitet die Daten aus einer Reihe möglicher Beobachtungen $\mathcal{O}^j$ und kann eine Aktion a_t^j entsprechend möglicher Aktionen $\mathcal{A}^j$ des Agenten j auslösen.

Die Aktionen der Agenten in dem multiagenten Framework zum Zeitpunkt t werden im Vektor $\mathbf{a}_t = (a_t^i, \ldots, a_t^j)$ zusammengefasst. Für die Berechnung der „Policy“ π des Agenten j wird eine Sequenz $0, \ldots, t$ an vergangenen Aktionen $a_{o:t}^j = (a_0^j, \ldots, a_t^j)$ benötigt. Die „Policy“ wird durch

$$\pi(a_t|\mathbf{h}_{0:t}) = \prod_j \pi^j(a_t^j|h_{0:t}^j) \tag{6.1}$$

beschrieben, wobei die Vergangenheit (*history*) eines Agenten j zum Zeitpunkt t durch $h_{0:t}^j = (a_{0:t}^j, o_{1:t}^j)$ definiert wird.

Die Zustände des auf Multiagenten basierenden POMDP-Modells werden durch eine Anfangsverteilung $p_0(s)$ und eine Zustandsübergangsverteilung $p(s_t|s_{t-1}, \mathbf{a}_{t-1})$ definiert (siehe Abbildung 6.1). Die Beobachtung der jeweiligen Verarbeitungsstufen in einem Zustand s_t durch einen Agenten j berechnet sich nach $p(o_t^j|s_t, \mathbf{a}_{t-1})$. Die Verbundwahrscheinlichkeit wird nach

$$p(s_{0:t}, \mathbf{h}_{0:t}) = p_0(s_0) \prod_{i=0}^{t-1} \pi(\mathbf{a}_i|\mathbf{h}_{0:i})\, p(s_{i+1}|s_i, \mathbf{a}_i)\, p(\mathbf{o}_{i+1}|s_{i+1}, \mathbf{a}_i) \tag{6.2}$$

beschrieben, mit $p(\mathbf{o}_{t+1}|s_{t+1}, \mathbf{a}_t) = \prod_j p(o_{t+1}^j|s_{t+1}, \mathbf{a}_t)$ (siehe Abbildung 6.1).

In Gleichung 6.2 sind mit der Dialogvergangenheit (*dialog history*) $\mathbf{h}$, den Aktionen $\mathbf{a}$, den Zuständen s und den Beobachtungen in Form von semantischen Slots o alle Variablen des POMDP-Dialogmanagementsystems, realisiert in einem verteilten Framework, beschrieben.

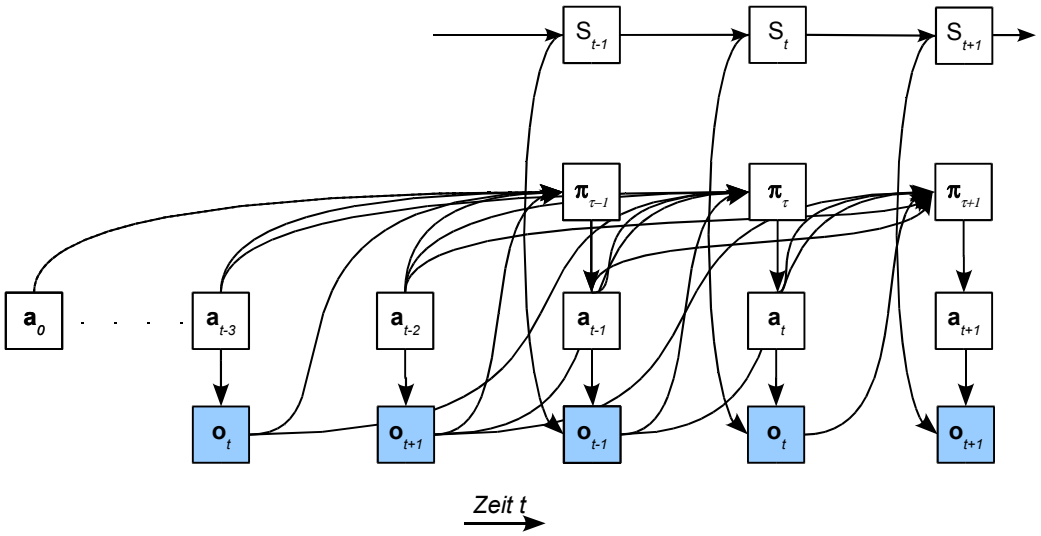


Abbildung 6.1: Graphisches Modell eines Dialogmanagementsystems für den Vorgang in einem Agenten: In jedem Zustand s findet zu jedem Zeitpunkt t eine Beobachtung statt. Aufgrund einer Berechnungsvorschrift (dialog policy) π wird zu jedem Zeitpunkt t eine Aktion a ausgelöst. π wird mit der Dialogvergangenheit $\mathbf{h}$, bestehend aus n Aktionen a und n Beobachtungen von semantischen Slots o, berechnet.

6.2 Graphentheoretisches Gesamtmodell

Die Verfahren aus Kapitel 2 bis Kapitel 6 werden hier in einem einheitlichen, graphentheoretischen Modell zusammengefasst. Somit werden verschiedene For-

schungsgebiete, wie z. B. Akustik, Phonetik, Syntax, Semantik und Dialogmanagement mit jedem Zeitschritt t einstufig verarbeitet. In jeder Verarbeitungsstufe können verschiedene, in dieser Arbeit beschriebene GM verwendet werden, wodurch eine hohe Flexibilität des Systems erreicht wird. In Abbildung 6.2 ist das aus den Experimenten hervorgegangene beste GM in einem einstufigen Modell zusammengefasst.

Zwischen den Verarbeitungsstufen, welche hier als Agenten realisiert werden, existieren fest definierte Schnittstellen, sodass die Modelle jederzeit ausgetauscht werden können. Durch diese 1-stufige Modellierung ergeben sich folgende Vorteile:

- Zugriff auf eine einheitliche Wissensbasis aus syntaktischem, semantischem Wissen und der Dialogvergangenheit.
- Integrierte Wissensverarbeitung ohne Informationsverlust in den Übergängen der Verarbeitungsstufen.
- Austauschbarkeit der Module und Parallelisierung einzelner Verarbeitungsstufen.

Eine Realisierung des gesamten Graphischen Modells (siehe Abbildung 6.2) ist aufgrund der Architektur von GMTK derzeit nicht möglich [Bil04]. Dennoch kann jede einzelne (GM-basierte) Verarbeitungsstufe als Agent realisiert werden.

Die Austauschbarkeit (während der Laufzeit) und Parallelisierung der Agenten (Verarbeitungsstufen) wird hier in einem Framework verteilter Agenten realisiert. Aus dem graphentheoretischen Modell (siehe Abbildung 6.2) sind die Schnittstellen der einzelnen Agenten definiert. Alle Agenten greifen im Sinne des graphentheoretischen Modells auf eine einheitliche Wissensbasis zu. Damit ist eine konsistente Informationsverarbeitung von der Spracherkennung bis zum Dialogmanagement gewährleistet.

6.3 Verteilte Agenten (Agentenframework)

In diesem Abschnitt werden die Verfahren aus dem Abschnitt 6.2 in ein Multiagentensystem überführt. Die Modellierung (siehe Abbildung 6.2) erlaubt eine Verarbeitung multi-modaler Beobachtungen $\mathcal{O}$ und Aktionen $\mathcal{A}$, wie sie beispielsweise in kognitiv-technischen Systemen (z. B. einem assistiven Haushaltsroboter) gebraucht werden [SBG+08].

Im Rahmen dieser Arbeit erhält ein Dialogagent (Theorie, siehe Kapitel 2) semantische Slots eines natürlich gesprochenen Satzes, welche mithilfe eines Spracherkenners und der semantischen Decodierung (siehe Kapitel 4) ermittelt werden. Die semantische Decodierung ist hier ebenso als Agent realisiert. Die Agenten arbeiten eigenständig, lediglich die Ergebnisse werden auf einer gemeinsamen Wissensbasis im Sinne von Abbildung 6.3 ausgetauscht.

Das Agentennetzwerk wird mittels dem Java Agent DEvelopment Framework (JADE) [BCPR03] aufgebaut. JADE basiert auf der Open Agent Architecture [CM01], unterstützt Peer-to-peer Applikationen und stellt neben der graphischen Benutzeroberfläche weitere Debug Tools (SNIFFER,INTROSPECTION AGENT, siehe Anhang 1.3) zur Verfügung.

Im Gegensatz zu anderen „Frameworks“, wie z. B. Trindikit [LBHH04], unterstützt JADE mit einer zentralen Moderator Funktion (*directory facilitator*) oder auch „Gelbe Seiten“ (*yellow pages*) genannt, verteilte Anwendungen. Diese basieren auf auf einer offenen Agentenarchitektur (OAA) [CM01]. Über diesen Agenten wird die Kommunikation der einzelnen Agenten innerhalb des Frameworks gemanagt.

Um eine Flexibilität im „Framework“ zu erhalten ist es wichtig, präzise Schnittstellen, analog zu Abbildung 6.2 der Verarbeitungsstufen zu definieren. Neben den Spracheingabe- und TTS-Ausgabeagenten werden mithilfe der Java Expert System Shell (JESS) und POMDP (siehe Kapitel 5) zwei exemplarische Vertreter fundamental unterschiedlicher Methoden für das Dialogmanagement in das „Framework“ eingebunden und einer einheitlichen Wissensbasis (hier auch JESS) zur Verfügung gestellt. Um die definierten Regeln und Fakten gegenüber einer Wissensbasis zu vergleichen, benutzt JESS den effizienten **RETE**-Algorithmus[1] [For82].

Regelbasierte Agenten Das regelbasiertes Toolkit JESS [FH03] basiert auf einem „Forward-chaining“-Algorithmus und beinhaltet ein Knowledge-System, welches mittels RETE seine Regeln optimiert. Die benötigten Fakten stammen aus der semantischen Decodierung (siehe Kapitel 4) eines in einem Eingabeagenten integrierten Spracherkenners.

6.4 Zusammenfassung des Kapitels

In diesem Kapitel wurden zunächst in einer integrierten Darstellung mit GM alle Verarbeitungsstufen eines Dialogsystems vorgestellt. Jedoch ist GMTK nicht für ein einstufiges Dialogmanagementsystem geeignet, da eine integrierte Decodierung während des Sprechens derzeit nicht vorgesehen ist. Deshalb wurden hier die einzelnen Verarbeitungsstufen auf ein agentenbasierendes Framework portiert, womit eine parallele Verarbeitung der Dialogmodelle möglich ist [BCPR03]. Die Parallelisierung ist für die Skalierbarkeit des Systems sinnvoll und wurde exemplarisch in Form von zwei parallel arbeitenden Dialogmanagern (regelbasiert und stochastisch) implementiert [SSRW09]. Die Parameter des stochastischen Dialogmanagementmodells mussten dazu mit dem ATIS-Korpus eingestellt werden. Dies erfolgte analog zum POMDP-Training, welches in Kapitel 5 beschrieben wurde. Abschließend wurde

[1] RETE(lat.) Netz oder auch Netzwerk: Effizienter Algorithmus, der häufig in der Mustererkennung zur Abbildung von Systemprozessen über Regeln verwendet wird.

das System erfolgreich auf mehreren Domänen mit durchschnittlich $N(d) = 10$ Dialogschritten getestet, siehe [SBG+08, SRWR09, SSRW09, SMS+09].

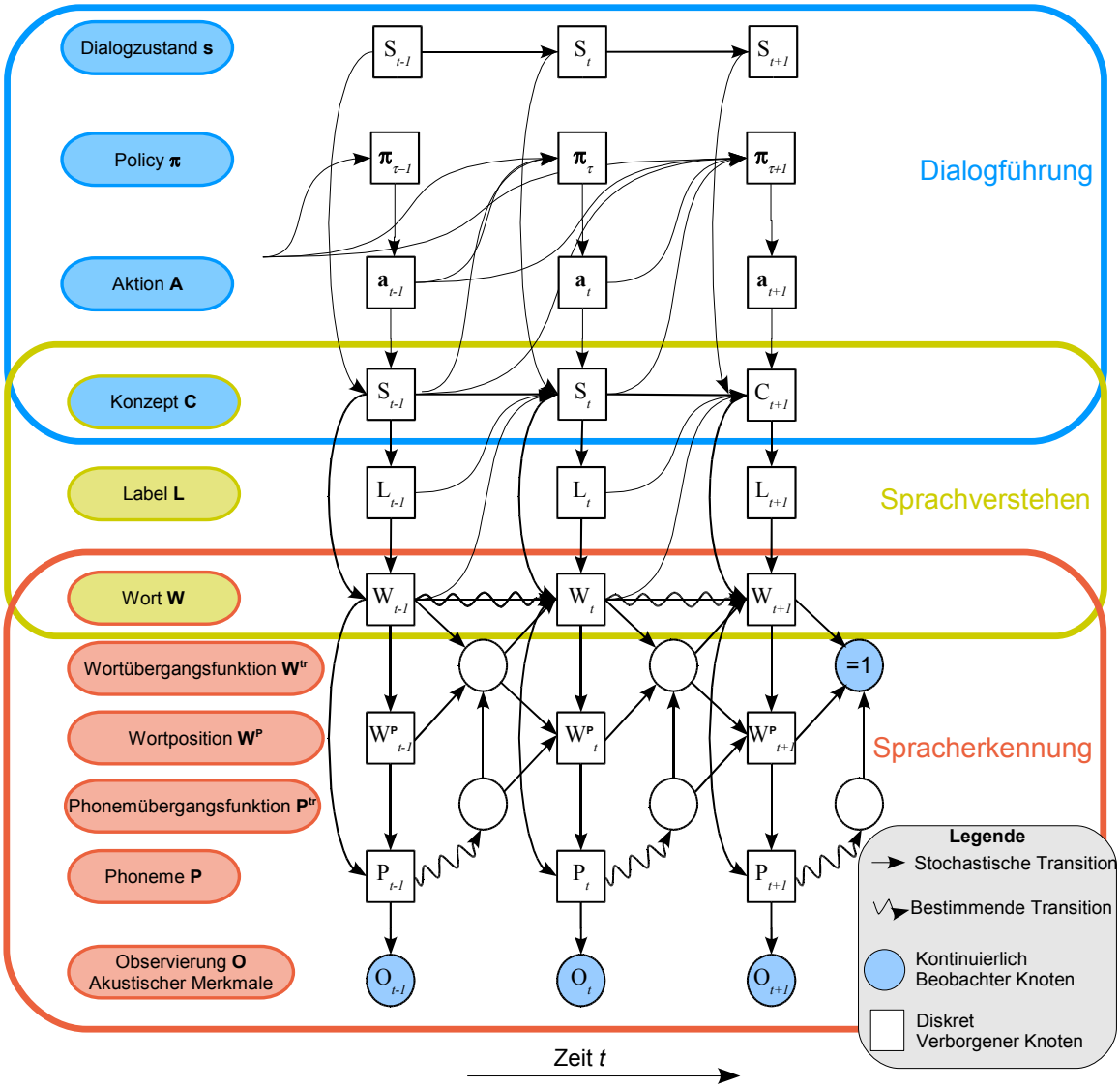


Abbildung 6.2: Graphentheoretische Zusammenfassung der einzelnen Verarbeitungsstufen. Hier sind jeweils die besten Verfahren aus der Spracherkennung, der semantischen Decodierung und dem Dialogmanagement in einem Graphischen Modell dargestellt.

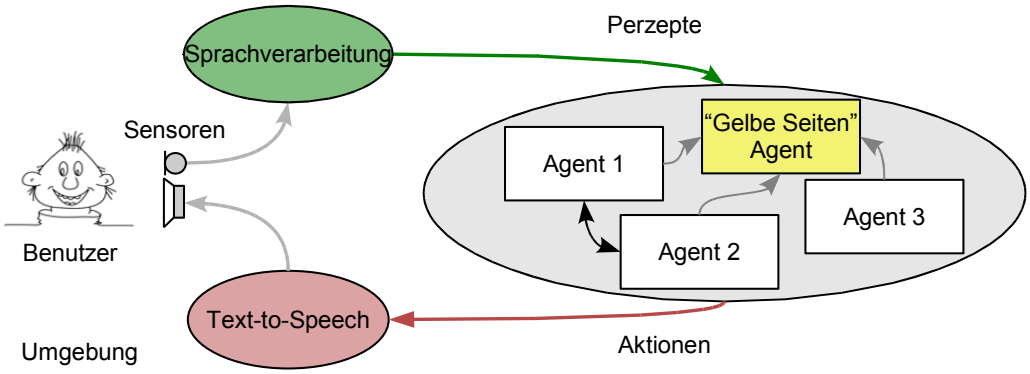


Abbildung 6.3: Konzeptionelle agentenbasierte Systemarchitektur: Agenten melden ihre Funktionalität beim „Gelben Seiten“ Agenten an. Im weiteren Verlauf kommunizieren die Agenten direkt miteinander und stellen dem Agentensystem die Ergebnisse zur Verfügung. Ein Agent verarbeitet z. B. die gesprochene Äußerung und stellt der Agentengemeinschaft die semantischen Slots bereit.

7

Zusammenfassung und Ausblick

Die vorliegende Arbeit befasste sich mit der Modellierung aller Verarbeitungsstufen des natürlichsprachlichen Mensch-Maschine-Dialogs. Aus den gesprochenen Äußerungen, die in Form von geschriebenen, aus dem Spracherkenner generierten Textphrasen vorlagen, wurden zwei verschiedene Verfahren zur Decodierung bedeutungstragender Satzphrasen entwickelt. Ausgehend von der Zuweisung von Bedeutungen in Form von „semantischen Slots“ wurden im Dialogmanagementsystem eine Dialogstrategie modelliert. Die in dieser Arbeit abgeleiteten Modelle werden einheitlich als Graphische Modelle (GM) beschrieben, womit die einzelnen Verarbeitungsstufen kombiniert werden konnten. Die Parameter des stochastischen Modells wurden mit einem Korpus eines Flugauskunftssystem trainiert.

7.1 Entwickelte Modelle und erzielte Ergebnisse

Nach einer Einführung der „Graphischen Modellierung“ am Beispiel eines Spracherkenners (Bestimmung der akustischen Parameter, Modellierung mithilfe der Hidden-Markov-Modelle (HMM), Lernverfahren und Decodierung mithilfe des Viterbi-Algorithmus), wurden drei verschiedene Modelle zur semantischen Decodierung und zur Dialogsteuerung entwickelt und experimentell untersucht. Zu diesem Zweck wurden die Ergebnisse für jede Verarbeitungsstufe (die semantische Decodierung und das Dialogmanagementsystem) mit einem implementierten Referenzsystem verglichen.

7.1.1 Semantische Decodierung

Für die semantische Decodierung wurden im Rahmen dieser Arbeit zwei Verfahren eingesetzt: Mittels erweiterten kontextfreien Grammatiken (*extended context-free grammars*) (ECFG) konnten besonders gut für regelhaft gesprochene Phrasen wie z. B. Uhrzeiten oder Flugnummern modelliert werden, und GM eigneten sich als stochastische Verfahren zur Modellierung von längeren gesprochenen Äußerungen.

Die Regeln der ECFG wurden von Experten geschrieben, wodurch die Fehlerquelle minimiert wurde. Als vorteilhaft erwiesen sich hier die nur auf „Wortebene“ benötigten Trainingsdaten, da das System flexibel und unabhängig von der Anwendungsdomäne eingesetzt werden konnte.

Längere Redewendungen wurden stochastisch mithilfe von GM modelliert. Dazu wurden vier zweistufige hierarchische Modelle vorgestellt. Sie unterscheiden sich in der statistischen Abhängigkeit des verwendeten diskreten Knotens für die semantischen Konzepte. Die optimale Zuordnung der Worte mit den verfügbaren „semantischen Slots“ erfolgt mit dem Viterbi-Algorithmus. Die besten Ergebnisse lieferte das stochastische Modell, welches mit grammatikalischen Regeln und Bedingungen kombiniert wurde. Diese Kombination beweist ihre Stärken besonders bei regelhaft gesprochenen Äußerungen.

7.1.2 Dialogmanagementsystem

Im zweiten Teil dieser Arbeit wurde ein GM für das Dialogmanagementsystem entwickelt. Zu diesem Zweck wurde mit dem „teilweise beobachtbaren markoffschen Entscheidungsprozess“ (POMDP)-Modell ein Referenzsystem für das verwendete Flugauskunftssystem erstellt. Für die Vorausberechnung möglicher Dialogstrategien wurde hier ein hoher Rechenaufwand benötigt.

Des Weiteren wurde ein Mixed-Initiative-Dialogmanagementsystem entwickelt, welches die Dialogstrategie nicht im Voraus, sondern zur Laufzeit berechnet. Durch den geringeren Rechenaufwand bleibt das System flexibel einsetzbar und die Zahl möglicher Anwendungen kann erhöht werden. Das GM konnte mit vergangenen, aktuellen und geschätzten, zukünftigen Beobachtungen von „semantischen Slots“ Pfade zum nächsten Benutzerziel berechnen. Der beste Pfad wurde hier mit einem für das Dialogmanagement implementierten Viterbi-Algorithmus gefunden.

Im letzten Teil wurde die im Rahmen dieser Arbeit entwickelte GM-basierte semantische Decodierung mit dem Dialogmanagement auf einem „Multiagenten System“ eingesetzt. Jede Verarbeitungsstufe wurde hier in Form eines Agenten implementiert. Das System basiert auf einer einheitlichen Modellierung der Verarbeitungsstufen (von der Spracherkennung bis zum Dialog) des natürlichsprachlichen Mensch-Maschine-Dialogs mit GM.

7.2 Diskussion

Die Einsetzbarkeit für unterschiedliche Aufgaben des natürlichsprachlichen Mensch-Maschine-Dialogs zeigt die große Flexibilität von GM. Im Vergleich zu vielen anderen Methoden erlauben sie eine integrierte Modellierung der einzelnen Verarbeitungsstufen. Über GM lassen sich die Probleme des natürlichsprachlichen Mensch-Maschine-Dialogs auf abstrakter Ebene modellieren und jedes Modell für sich analysieren.

Mit ihren vielfältigen Anwendungsmöglichkeiten einerseits und mit ihrer klaren und einfachen Beschreibungsform andererseits bieten GM einen einheitlichen Blickwinkel auf verschiedene Probleme. So wurden hier unterschiedlichste Herangehensweisen in der semantischen Decodierung und im Dialogmanagement einheitlich mit GM formuliert und verglichen.

Aus den Experimenten und Ergebnissen wird ersichtlich, dass sich GM prinzipiell für den Einsatz in der semantischen Decodierung und für Dialogmanagementsysteme eignen. Dafür waren jedoch erhebliche Mengen an annotierten Trainingsdaten notwendig. Im Besonderen für die semantische Decodierung und die Dialogsteuerung sind auf jeder Modellierungsebene auch Annotierungen für jede Ebene (Wortklassen, Konzepte, Benutzerziel, etc.) notwendig. Dies führt zu einem erheblichen Mehraufwand bei der Erstellung von Trainingsdaten. Als eine echte Alternative, speziell für kleine Anwendungsbereiche, boten sich die ECFG an, die zum Teil in das GM integriert wurden.

Mit der integrierten Modellierung des gesamten natürlichsprachlichen Dialogsystems wurden aber auch die Grenzen der softwaretechnischen Umsetzung mit GM erreicht: Graphische Modelle können derzeit keine Klassifizierung aus einem Informationsfluss treffen. Für das Dialogsystem bedeutet dies, dass erst zum Ende des Dialogschrittes die Entscheidung gefällt wird. Ebenso kann erst zum Ende des gesprochenen Satzes eine semantische Zuordnung stattfinden. Jede Verarbeitungsstufe verzögert also den Dialogverlauf im System.

Um dieses Problem zu lösen, wurde im Rahmen dieser Arbeit ein multiagentenbasierendes System eingesetzt, bei dem jede Verarbeitungsstufe als Agent realisiert wurde. Das GM konnte genutzt werden für die Erhöhung des Verständnisses und zum Skizzieren der Lösung für jede Verarbeitungsstufe.

7.3 Ausblick

Die hier erstellten Untersuchungen beziehen sich auf die Modellierung des natürlichsprachlichen Mensch-Maschine-Dialogs. Eine Erweiterung des Dialogs auf andere Modalitäten, wie z. B. Gesten, Emotionen und Intentionen und deren Modellierung mit GM bleibt weiterführenden Arbeiten vorbehalten.

Eine weitere Verbesserung der Ergebnisse mit einer höheren Aussagekraft zur statistischen Signifikanz könnte durch einen größeren Trainingskorpus erzielt werden. Insbesonders für die Modellierung mehrerer Ebenen sind weitere Verbesserungen in der hierarchischen Annotation erforderlich.

Mit dem verwendeten Spracherkenner konnten weder Konfidenzen noch alternativ gesprochene Äußerungen an die semantische Decodierung und das Dialogmanagement weitergeleitet werden. Durch die klaren Schnittstellen könnte hier das agentenbasierende Dialogsystem durch alternative (parallel arbeitende) Spracherkenner erweitert werden.

1

Anhang

1.1 Parameter Abschätzung und Training des HVS-Modells

Die Modellparameter werden mit λ bezeichnet. Im Hidden-Vector-State (HVS)-Modell findet eine neuerliche Abschätzung (*re-estimation*) mit dem EM-Algorithmus [DLR77] $L(\lambda) = \log P(N, \mathbf{C}, W|\lambda)$ statt.

Für ein Training wird die Hilfsfunktion Q definiert:

$$\begin{aligned} Q(\lambda|\lambda^c) &= E[\log P(N, \mathbf{C}, W|\lambda)|W, \lambda^c] \\ &= \sum_{\mathbf{C},N} P(N, \mathbf{C}|W, \lambda) \log P(N, \mathbf{C}, W|\lambda^c) \\ &= \sum_{\mathbf{C},N} P(N, \mathbf{C}|W, \lambda)\ log\ P(N, \mathbf{C}, W|\lambda^c) \end{aligned} \tag{1.1}$$

Durch Einsetzen von Gleichung (1.1) in Gleichung (4.6) lässt sich folgende Formel für das Training ableiten:

$$\hat{P}(n|\mathbf{c}) = \frac{\sum_t P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}|W, \lambda)}{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda)} \tag{1.2}$$

$$\hat{P}(c[1]|c[2 \ldots D]) = \frac{\sum_t P(\mathbf{c}_t = c|W, \lambda)}{\sum_t P(c_t[2 \ldots D] = c[2 \ldots D]|W, \lambda)} \tag{1.3}$$

$$\hat{P}(w|\mathbf{c}) = \frac{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda)\ \delta(w_t = w)}{\sum_t P(\mathbf{c}_t = \mathbf{c}|W, \lambda} \tag{1.4}$$

wobei $\delta(w_t = w) = 1$ wenn das Wort $w_t = w$ ist, ansonsten ist $\delta(w_t = w) = 0$

Die Parameterabschätzung (siehe Abbildung 1.1) der Gleichungen 1.2, 1.3 und 1.4 erfolgt mit dem Vorwärts- bzw. Rückwärtsalgorithmus. Der Vorwärtswahrscheinlichkeit α und die Rückwärtswahrscheinlichkeit β sind folgendermaßen definiert:

$$\alpha_k(t) = P(w_1, w_2, \ldots, w_t, \mathbf{c}_t = \mathbf{c}|\lambda) \tag{1.5}$$

$$\beta_k(t) = P(w_{t+1}, w_{t+2}, \ldots, w_T)|\mathbf{c}_t = \mathbf{c}_t, \lambda) \tag{1.6}$$

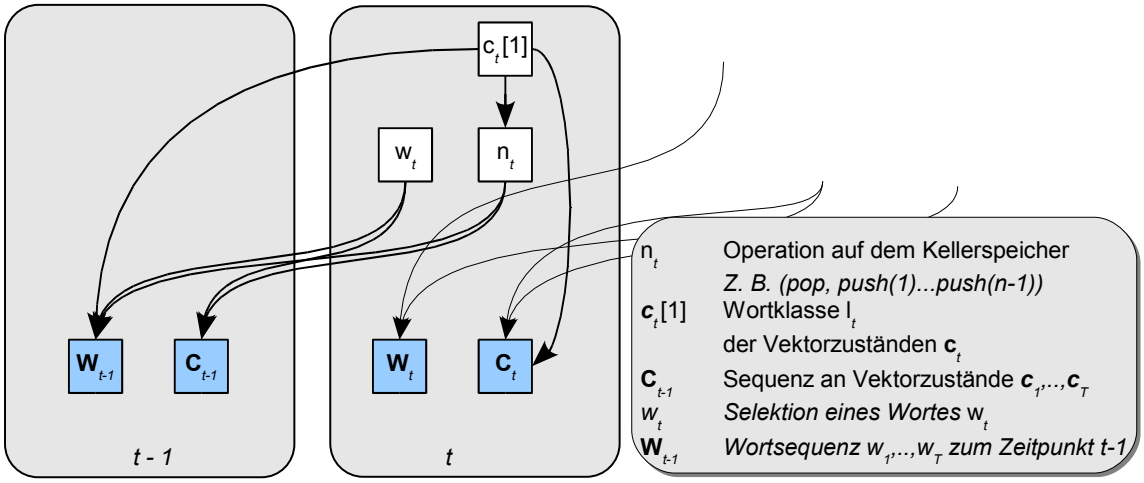


Abbildung 1.1: HVS Traingsmodell

Damit lassen sich die Wahrscheinlichkeiten für das Training ableiten, nach

$$P(n_t = n, \mathbf{c}_{t-1} = \mathbf{c}') = \alpha_{c'}(t-1)\, P(n_t = n|\mathbf{c}_{t-1} = \mathbf{c}') \cdot \cdot \sum_{\{\mathbf{c}|\mathbf{c}' \to \mathbf{c}\}} P(c[1]|c[2 \dots D])\, P(w_t|\mathbf{c}\, \beta_{\mathbf{c}}(t) \tag{1.7}$$

$$P(c_t[2 \dots D] = c[2 \dots D]) = \sum_{\{\mathbf{c}|c_t[2 \dots D] = c[2 \dots D]\}} \alpha_{\mathbf{c}}(t)\, \beta_{\mathbf{c}}(t) \tag{1.8}$$

$$P(\mathbf{c}_t = \mathbf{c}) = \alpha_{\mathbf{c}}(t)\, \beta_{\mathbf{t}}(t) \tag{1.9}$$

Die Wahrscheinlichkeit des Vorwärts- und Rückwärtsalgorithmus α bzw. β (siehe Gleichung 1.10 und 1.11) werden rekursiv berechnet, mit

$$\alpha_{\mathbf{c}}(t) = \sum_{\{\mathbf{c}' \to \mathbf{c}} \alpha_{\mathbf{c}'}(t-1)\, P(\mathbf{c}|\mathbf{c}')\, P(w_t|\mathbf{c}) \tag{1.10}$$

$$\beta_{\mathbf{c}}(t) = \sum_{\{\mathbf{c}'|\mathbf{c} \to \mathbf{c}'} P(\mathbf{c}''|\mathbf{c}\, P(w_{t+1}|\mathbf{c}'')\, \beta_{\mathbf{c}''}(t+1) \tag{1.11}$$

$$P(\mathbf{c}|\mathbf{c}') = P(\hat{n}|\mathbf{c}')\, P(c[1]|c[2 \dots D]). \tag{1.12}$$

Algorithmus 4 Vorwärts- und Rückwärtswahrscheinlichkeit im HVS-Modell

Benötigt: Beobachtungen (Wörter) $W = w_0 \dots w_T$ aus einem Traingsset TS, Hidden Vector State Netz N_{hvs}, semantischer Zustandsvektor $\mathbf{C}$
Stellt sicher: $|W| = |\mathcal{C}| = \mathbf{C}_1, \dots, \mathbf{C}_N$
function FORWARDBACKWARD(Trainingsset T, Zustandsvektor $\mathbf{C}$)
Initialisiere Knoten n_{HVS}, Kanten e_{in}, e_{out}
Initialisiere $\alpha_c == \beta_c == 0$
Initialisiere $\mathbf{c}_i$, $1 \leq i \leq N_{\text{Wörter}}$
Initialisiere Satz S
for $s \in \{1; \dots; |T|\}$ **do**
for $w \in \{1; \dots; |s|\}$ **do**
WordNum n_w = hvsnet.GETVOCABULARYNUM(w)
for $c_i^s, n_i \in \{1 \leq i \leq N_{\text{HVS}}\}$ **do**
if $n_i == n_{\text{EndNode}}$ **then**
$\mathrm{n}_i^{\text{back}} = 1$
end if
for all $e_j \in n_i$ **do**
Endnode: $n_e = \text{Node}[1](e_j)$
$\beta_{\text{back}}(n_i) = \mathcal{P}_{\text{stackOP}}(n_i|c_i)\, \mathcal{P}_{\text{trans}}(n_e)\, \mathcal{P}_{\text{wort}}(w_i|n_e)\, \beta_{\text{back}}(n_e)$ ▷ Backward Walk
$\alpha_{\text{for}}(n_e) = \alpha_{\text{for}}(n_i)\, \mathcal{P}_{\text{stackOP}}(n_e|c_e)\, \mathcal{P}_{\text{wort}}(w_i|n_e)\, \mathcal{P}_{\text{wort}}(w_i|c_i)$ ▷ Forward Walk
end for
end for
end for
end for
end function

1.2 Erweiterte kontextfreie Grammatiken

Für die Evaluation der semantischen Decodierung mittels ECFG wurden drei Experimente mit verschiedenen Konfigurationen der Grammatik $\mathcal{G}_{\{1,2,3\}}$ durchgeführt:

$$
\begin{aligned}
\mathcal{G}_1 = \; & \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* | \\
& | \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* | \\
& | \mathit{dummy}^* \textbf{ORT} \cdot \text{dummy}^* | \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^*; \\
\textbf{ORT} = \; & \textbf{DESTINATION} \cdot \textbf{ORIGIN} | \\
& | \textbf{ORIGIN} \cdot \textbf{DESTINATION} | \\
& | \textbf{ORIGIN} | \textbf{DESTINATION}; \\
\textbf{TIME} = \; & \mathit{ATime}^* \cdot \text{dummy}^* \cdot \mathit{WCHour24} \cdot \\
& \cdot (\text{dummy}^* \cdot \mathit{WCHour24} | \mathit{ATime} \cdot \mathit{WCMinute} | \mathit{ATime})^?; \\
\textbf{ORIGIN} = \; & \mathit{AOrigin}^? \cdot \text{dummy}^* \cdot \mathit{WCPlace}; \\
\textbf{DESTINATION} = \; & \mathit{ADestination}^? \cdot \mathit{dummy}^* \cdot \mathit{WCPlace};
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{G}_2 = \; & \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* | \\
& | \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* | \\
& | \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* | \\
& | \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^*; \\
\textbf{ORT} = \; & \textbf{DESTINATION} \cdot \textbf{ORIGIN} | \\
& | \textbf{ORIGIN} \cdot \textbf{DESTINATION} | \\
& | \textbf{ORIGIN} | \textbf{DESTINATION}; \\
\textbf{TIME} = \; & \mathit{ATime}^* \cdot \text{dummy}^* \cdot \\
& \cdot (\mathit{WCHour24} \cdot (\text{dummy}^* \cdot \mathit{WCHour24} | \mathit{ATime} \cdot \mathit{WCMinute})^? \cdot \\
& \cdot \mathit{ATime}^?) | ((\mathit{WCMinuteRel} \cdot \mathit{ATime})^? \cdot \\
& \cdot \text{dummy}^* \cdot \mathit{WCHour12} \cdot \text{dummy}^* \mathit{ATime}^?); \\
\textbf{ORIGIN} = \; & (\mathit{AOrigin} | \mathit{Origin})^? \cdot \text{dummy}^* \cdot \mathit{WCPlace}; \\
\textbf{DESTINATION} = \; & (\mathit{ADestination} | \mathit{Destination})^? \cdot \text{dummy}^* \cdot \mathit{WCPlace};
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{G}_3 = \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* | \\
|\text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy}^* | \\
|\text{dummy}^* \cdot \textbf{ORT} \cdot \text{dummy}^* | \text{dummy}^* \cdot \textbf{TIME} \cdot \text{dummy} *; \\
\textbf{ORT} = \textbf{DESTINATION} \cdot \textbf{ORIGIN} | \\
|\textbf{ORIGIN} \cdot \textbf{DESTINATION} | \\
\textbf{ORIGIN} | \textbf{DESTINATION}; \\
\textbf{TIME} = (\mathit{ATime}^* \cdot \text{dummy}^* \cdot \mathit{WCHour24}) \cdot \\
\cdot (\text{dummy}^* \cdot \mathit{WCHour24})^?) \cdot \mathit{ATime}^? \cdot \mathit{WCMinute}^?) | \\
|(\mathit{ATime}^* \cdot \text{dummy}^* \cdot \text{WCHour24} \cdot \mathit{ATime}) | \\
|(\mathit{ATime}^* \cdot \text{dummy}^* \cdot (\mathit{WCMinuteRel} \cdot \mathit{ATime})^? \cdot \\
\cdot \text{dummy}^* \cdot \mathit{WCHour12} \cdot \text{dummy}^* \cdot \mathit{ATime}^?); \\
\textbf{ORIGIN} = (\mathit{AOrigin} | \mathit{Origin})^? \cdot \text{dummy}^* \cdot \mathit{WCPlace}; \\
\textbf{DESTINATION} = (\mathit{ADestination} | \mathit{Destination})^? \cdot \text{dummy}^* \cdot \mathit{WCPlace};
\end{aligned}
$$

1.3 Implementierung des HMD Dialogmodells

Die Implementierung der GM erfolgte mit GM-Software (*Graphical Model Toolkit*) (GMTK) [BZ02]. In der Masterdatei (siehe Code 1.1) wird die Struktur des Graphischen Modells festgelegt.

```
1  #ifndef COMMON_PARAMS
2  #define COMMON_PARAMS
3  #define SLOT_STATE_SIZE   256
4  #define OBSERVATION_GAUSS_SIZE    256
5  #define FUTURE_SIZE       2
6
7  GRAPHICAL_MODEL model_modifiedHMM
8  frame: 0
9  {
10    variable: slotstate
11              {
12       type: discrete observed 0:0 cardinality SLOT_STATE_SIZE;
13       switchingparents: nil;
14       conditionalparents: nil using DenseCPT("observation_startCPT");
15              }
16 }
17 frame: 1
18 {
19    variable: slotstate
20    {
21       type: discrete hidden cardinality SLOT_STATE_SIZE;
22       switchingparents:   nil;
23       conditionalparents: slotstate(-1)
24         using DenseCPT("slot_transitionCPT");
25    }
26    variable: future
27    {
```

```
28          type: discrete observed 1:1 cardinality FUTURE_SIZE;
29          switchingparents: nil;
30          conditionalparents: nil using DenseCPT("future_startCPT");
31      }
32      variable: observations
33      {
34          type: discrete observed 0:0 cardinality OBSERVATION_GAUSS_SIZE;
35          switchingparents: future(0)       using mapping("futureDT");
36          conditionalparents: slotstate(0)
37            using DenseCPT("observation_gaussCPT")
38            |
39            slotstate(0) using DenseCPT("uniformCPT");
40      }
41  }
42  frame: 2
43  {
44      variable: slotstate
45      {
46          type: discrete observed 0:0 cardinality SLOT_STATE_SIZE;
47          switchingparents: nil;
48          conditionalparents: slotstate(-1)
49            using DenseCPT("slot_transitionCPT");
50      }
51  }
52  chunk 1:1
```

Algorithmus 1.1: GMTK Masterdatei

1.4 Implementierung des Agentensystems

In diesem Abschnitt wird die prototypische Realisierung des verteilten Agentensystems vorgestellt. Trotz unterschiedlicher Aufgabenstellung wurde hier auf die Wiederverwendbarkeit der Agenten geachtet, deren Aufgaben nachfolgend beschrieben werden:

Der Gelbe Seiten Agent dient als Informationshändler; er kennt die Fähigkeiten und Verfügbarkeit aller Agenten.

Der Generische Agent beschreibt die allgemeinen Aufgaben des Agenten innerhalb Java Agent DEvelopment Framework (JADE), wie z. B. Registrierung beim **Gelbeseiten Agent**.

Der JESS Agent wird aus dem **Generischen Agent** abgeleitet und beschreibt den Dialogablauf mit einem Regelwerk (siehe Abschnitt 6.3).

Der POMDP Agent wird ebenfalls vom **Generischen Agent** abgeleitet; dieser Agent beschreibt die Realiserung eines stochastisches Dialogmodells (siehe Abschnitt 5.1).

Der Spracherkenner Agent ermöglicht per Transmission Control Protocol (TCP), die Anbindung verschieder Spracherkenner.

Der Sprachausgabe Agent ermöglicht die Anbindung externen Sprachsynthese (*text-to-speech*) (TTS) System an die Agentengemeinschaft.

Das Ziel dieses Abschnittes ist es, die Systemarchitektur mit den eingeführten Agenten zu beschreiben, um Dialoge aus dem „Air Travel Information System“ (ATIS)-Szenario, mit einer durchschnittlichen Dialoglänge von $N(d) = 10$ zu führen. Dieser Nutzungsfall (*Use Case*) wurde mithilfe eines Systemtests mit verschiedenen Benutzern evaluiert.

1.4.1 Systemarchitektur

Beim Systementwurf wurde besonderen Wert auf die Wiederverwendbarkeit und Skalierbarkeit der Softwarekomponenten geachtet. Im *Generic Agent* sind allgemeine agentenspezifische Aufgaben, welche speziell für verteilte Agenten zu erledigen sind, implementiert. Dies sind z. B. die Initialisierung und der Registrierung des Agenten im Java Agent DEvelopment Framework (JADE)-Agentenframework, speziell beim dafür vorgesehenen *Yellow Page* Agenten.

Das Dialogmanagement erfolgt hier mithilfe zweier konkurrierender Agenten, dem *Jess Agent* und dem *POMDP Agent*.

- Im *Jess-Agent* wird über Expertenwissen der Dialog in Form von Regeln und Fakten gesteuert. Damit kann der Agent über die *Jess Rule Engine* mit einer internen Wissensbasis (Knowledge Base) (KB) kommunizieren und aus den Regeln und Fakten Schlüsse ziehen, um den Dialog zu steuern.

- Im *POMDP-Agent* wird mithilfe des annotierten Trainingskorpus eine Dialogvorschrift berechnet, wie sie in Kapitel 5 vorgestellt wurde. Die Realsierung erfolgte hier mit dem POMDP-Toolkit von Taha [Tah07].

Die prinzipielle Architektur, mit den zwei konkurrienden Agenten der Dialogsteuerung, ist in Abbildung 1.2 abgebildet. Im Sinne der Wiederverwendbarkeit wurden alle Agenten vom *Generic Agent* abgeleitet. Über einen *Server Agent* verfügen die Ein- und Ausgabeagenten über die Möglichkeit per Transmission Control Protocol (TCP), Nachrichten von der „Außenwelt“ in das Agentenframework aufzunehmen bzw. abzusetzen. Alle Agenten melden ihre Dienste und Verfügbarkeit beim Hauptagenten, den *Yellow Pages* an, deren Kommunikationsabläufe im Folgenden beschrieben werden.

1.4.2 Kommunikation und Interaktion der Agenten

Das Unified Modeling Language (UML)-Sequenzdiagramm in Abbildung 1.3 zeigt die prinzipielle Interaktion der Agenten. Zuerst melden alle Agenten ihren Service und ihre Verfügbarkeit bei den *Yellow Pages* des JADE Frameworks an. Der *Yellow Page*-Agent agiert als Informationshändler, der weiß, welche Agenten gerade verfügbar sind.

Nach der Registrierung wird der *Jess Agent* initialisiert, indem Regeln und Fakten aus einer Datei gelesen werden. Wenn der *Jess Agent* mit den semantischen Slots

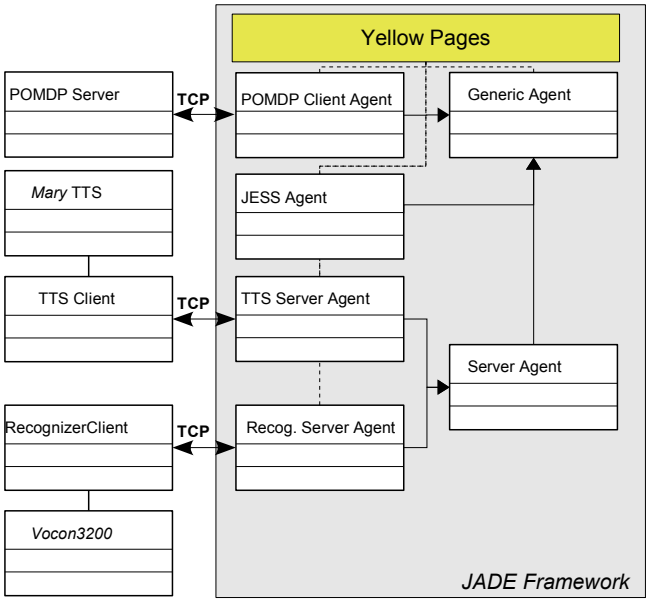


Abbildung 1.2: Hybrides Dialogmanagement, realisiert in einem multiagenten Framwork JADE.

keine Schlussfolgerung ziehen kann (siehe Abbildung 1.3), wartet das System die Antwort das *POMDP Agenten* ab, bevor es mithilfe der Sprachsynthese (*text-to-speech*) (TTS) eine Aktion auslöst und zum nächsten Dialogschritt wechselt.

Nach eingehenden Untersuchungen wird hier den regelbasierten Verfahren mithilfe Java Expert System Shell (JESS) gegenüber stochastischen Modellen der Vorrang gegeben. Nach Untersuchungen in Kapitel 4 wird zuerst nach der sicheren und einfacheren Regel gesucht (Konfidenz = 100%), bevor es zu einem stochastischen und fehlerbehafteten Modell den nächsten Dialogschritt abschätzt.

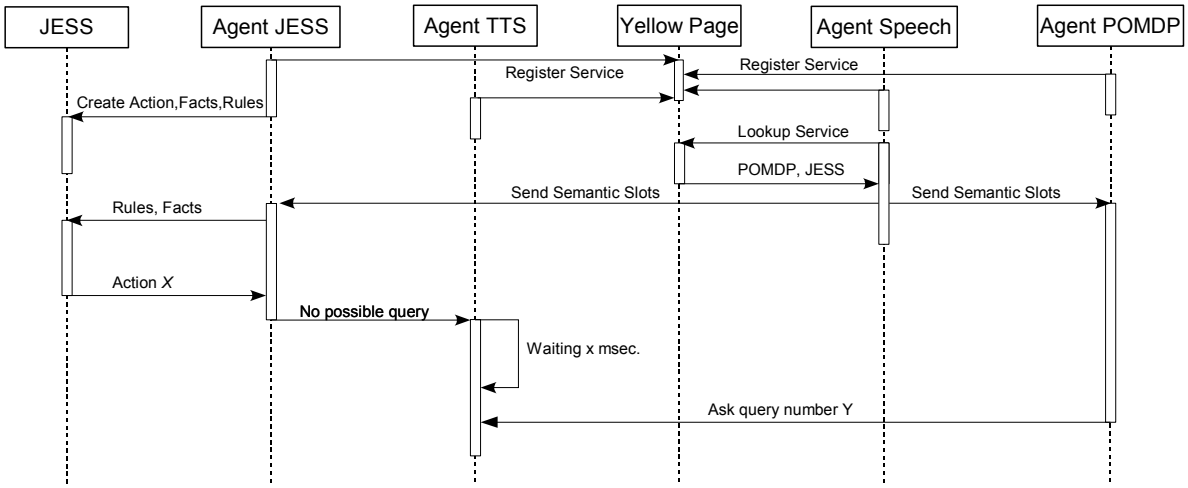


Abbildung 1.3: UML Sequenz Diagramm des multiagenten Dialogmangementsystems.

Abkürzungsverzeichnis

ADC	Absolute Discounting
ATIS	„Air Travel Information System“
ATT	American Telephone & Telegraph Corporation
BBN	Bolt Beranek and Newman Technologies
BN	Bayes’sche Netze
CFG	kontextfreien Grammatiken (*context-free grammars*)
CPT	bedingten Elternknoten (*conditional parents*)
DBN	dynamischen Bayes’schen Netzwerkes
ECFG	erweiterten kontextfreien Grammatiken (*extended context-free grammars*)
EM	Expectation-Maximization
FST	Transduktoren (*finite-state transducer*)
GM	Graphische Modelle
GMTK	GM-Software (*Graphical Model Toolkit*)
HMD	Hidden-Markov-Modell für das Dialogmanagement
HMM	Hidden-Markov-Modelle
HVS	Hidden-Vector-State
JADE	Java Agent DEvelopment Framework
JESS	Java Expert System Shell
KB	Wissensbasis (Knowledge Base)
KI	Künstlichen Intelligenz
MDP	Markoffschen Entscheidungsprozesse (*Markov Decision Processes*)
MIT	Massachusetts Institute of Technology
ML	Maximum-Likelihood Methode
NADIA	„Natürlichsprachigen Dialogführung im Auto“

NDA nicht-deterministischen endlichen Automaten
NPV negativer Vorhersagewert (*negative predictive value*)

OAA offenen Agentenarchitektur
OOV Out-Of-Vocabulary

POMDP „teilweise beobachtbaren markoffschen Entscheidungsprozess“

RE Reguläre Ausdrücke (*regular expressions*)
ROC „Empfänger Operationscharakteristik“ (*Receiver Operating Characteristic*)

SARSOP „Successive Approximations of the Reachable Space under Optimal Policies“
SPT schaltende Elternknoten (*switching parents*)
SQL Structured Query Language
SRI Stanford Research Institute

TCP Transmission Control Protocol
TTS Sprachsynthese (*text-to-speech*)

UML Unified Modeling Language

VB Verbund Baum
VQ Vektor-Quantisierer

WDF Wahrscheinlichkeitsdichtefunktion
WOZ Wizard-of-Oz

XML Extensible Markup Language

Symbolverzeichnis

$a \in \mathcal{A}$	Aktion
$\beta_t(s_i)$	Funktion des Rückwärtsalgorithmus: Knoten s_i zur Zeit t
$\alpha_t(s_i)$	Funktion des Vorwärtsalgorithmus: Knoten s_i zur Zeit t
$\gamma_t(s_i)$	a-posteriori Wahrscheinlichkeit im Zustand s_i
M_A	(nicht-deterministischer) endlicher Automat
L_A	Sprache von M_A
s_A	Startzustand $s_A \in Q_A$
δ_A	Transitionsmenge, endlicher Automat
Q_A	Zustandsmenge $Q_A = \Sigma \cup N$, endlicher Automat
$\mathcal{O}$	vollständige Beobachtungen
$\mathcal{L}$	Funktion der Datenwahrscheinlichkeit
$\delta_t(s_i)$	partiell optimaler Pfad
N	nicht-terminales Eingabealphabet
Σ	terminales Eingabealphabet
π	Vektor der Einsprungswahrscheinlichkeiten
$\mathcal{F}(Q, \lambda)$	Hilfsfunktion des EM-Algorithmus
$Q(\cdot)$	Verteilungsfunktion des EM-Algorithmus
$b_i(\cdot)$	Emmisionsfunktion zum Zeitpunkt i
$\mathbf{B}$	Matrix der Emissionswahrscheinlichkeiten
$\mathbf{G}$	Grammatik, z. B. erweiterte kontextfreie Grammatik
$\mathcal{G}$	Grahphen
$\mathcal{G}^D$	gerichteter, zyklusfreier Grahph
$\mathcal{G}'$	Teilgrahph
$\mathcal{G}_\setminus$	vollständiger Grahph
c	semantisches Konzept
$\mathbf{c}$	Folge von Konzepten $\mathbf{c} = c_1, c_2, \ldots, c_n$
$\mathcal{C}$	Konzeptmenge $\mathcal{C}$
Σ_{q_t}	Kovarianzmatrix im Zustand q_t
$\Sigma^k_{q_t}$	Kovarianzmatrix der Mixtur k im Zustand q_t
$\mathbf{x}$	(akustische) Merkmale
$\mathcal{X}$	(akustischer) Merkmalsatz $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
$\mathcal{Y}$	unvollständiger (verborgener) Merkmalsatz $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$
μ_{q_t}	Mittelwert im Zustand q_t
$\mu^k_{q_t}$	Mittelwert der Mixtur k im Zustand q_t
k	Mixtur $kinK_{q_t}$ in Zustand q_t

λ	Parameter eines (HMM-)Modell
E	Kanten (*nodes*)
N_{phon}	Anzahl der Phoneme
$\mathbf{p}$	Phonemekette
P	(Menge an) Produktionen
$\mathbf{q}$	Zufstandsfolge
$\hat{\mathbf{q}}$	wahrscheinlichste Zufstandsfolge
$\mathcal{Q}$	Menge aller Zustandsfolgen
q_t	Zustand zum Zeitpunkt t
$y_q^k t$	relative Häufigkeit k im Zustand q_t in Zustand q_t
$\psi_t(s_i)$	Rückschlusswahrscheinlichkeit im Zustand s_i
$\mathcal{S}$	Menge aller verborgenen Zustände im HMM
s_i	verborgener Zustand i
$\mathbf{x}$	akustisches Signal
$p(\mathbf{w})$	Sprachmodell
$S \in N$	Startsymbol des endlichen Automaten
$\mathbf{A}$	Matrix der Übergangswahrscheinlichkeiten
$a_{q_{t-1}\, q_t}$	Übergangswahrscheinlichkeit von Zustand q_{t-1} in den Zustand q_t
h	Vergangenheit (*history*), z. B. Dialogvergangenheit
V	Knoten (*vertices*)
$\mathbf{w}$	Wörter w
$\hat{\mathbf{w}}$	beste Wortsequenz
$\mathbf{w}$	Wortfolge $w_1, \ldots, w_i, \ldots, w_N$ mit $w_i \in \mathcal{W}$
l	(allg.) semantische Ebene, meist Wortklasse(label) l
$\mathbf{l}$	Folge von Wortklassen $\mathbf{l} = l_1, l_2, \ldots, l_n$
$\mathcal{L}$	(allg.) Annotationsmenge, meist Wortklassenmenge $\mathcal{L}$
w^p	Wortposition des Phonems p_t
w^{tr}	Wortübergang
w_t	Wort zum Zeitpunkt t
$s \in \mathcal{S}$	Zustand
$\mathcal{B}$	Menge aller semantischen Informationsfolgen $s_{i-1}\, s_i$, die zur Schätzung der Bigramm-Wahrscheinlichkeit des Dialogmodells verwendet werden
$\mathbf{O}$	Beobachtungsmatrix
$\mathbf{R}$	(Reward-)Belohungsmatrix
$t_{i\,j}$	Transition vom Zustand i in den Zustand j
$\mathbf{T}$	Transitionsmatrix
γ	backing-off Faktor γ
β	Verringerungsfaktor β
$N(d)$	Anzahl der Dialoge
$N(s)$	Anzahl möglicher Dialogübergänge
$N(d_s)$	Anzahl der (theoretisch) möglichen Dialogschritte
$N(T)$	Anzahl möglicher Transtionen in der Transitionsmatrix
$N(u)$	Anzahl der Benutzeräusserungen
$N(w)$	Anzahl der möglichen Wörter
d	Dialog
d_s	Dialogschritt
$\mathbf{d}_s$	Speichervektor eines Dialogschrittes
s	dezimaler Dialogzustand
g	Benutzerziel $g \in \mathcal{G} \subseteq \mathcal{S}$
u	Benuterzäusserung

w	ein Wort $w \in \mathcal{W}$
a	(Dialog-)Aktion a
$\mathbf{a}$	Folge von Aktionen $\mathbf{a} = a_1, a_2, \ldots, a_n$
$\mathcal{A}$	Folge von Aktionen $\mathbf{a} = a_1, a_2, \ldots, a_n$
$N(s)$	Anzahl der Zustände
Σ	(terminales) Eingabealphabet (allg.), z. B. Wortfolge $\mathbf{w}$
$AR_{\textbf{Satz}}$	semantische Satzerkennungsrate
$AR_{\textbf{Wort}}$	semantische Wortkennungsrate
$G(\cdot)$	Gewichtungsfunktion, z. B. über alles Transitionen
$\mathbf{H}$	Hierarchische Strukturliste
D_t	Kellerspeicherelement $D_t = D_{t-1} + 1 - n_t$ im HVS-Modell
$N = (n_1, n_2, \ldots, n_t)$	Folge von Kellerspeicheroperationen im HSV-Modell bis zum Zeitpunkt t
ST	Startknoten (Wurzel), z. B. im HVS Modell
$\mathbf{c}_t$	Zustandsvektors von Konzepten $(c_t[1], c_t[2], \ldots, c_t[D_t])$, z. B. im HSV Modell zum Zeitpunkt t
Γ	Kelleralpabet
$\#$	Anfangssymbol im Keller $\# \in \Gamma$
$\mathcal{M}$	Kellerautomat $\mathbf{M}$
$\mathcal{F}$	Menge an Endzuständen $\mathcal{F} \in \mathcal{S}$
c	Konzept c
$\mathbf{c}$	Folge von Konzepten $\mathbf{c} = c_1, c_2, \ldots, c_n$
$\mathcal{C}$	Konzeptmenge $\mathcal{C}$
$P(\mathbf{a})$	Dialogmodell
$\mathbf{N}$	Netzwerkwerk, bestehend aus Knoten und Kanten
$\mathcal{S}_{\textbf{Test}}$	Testset
$\mathcal{S}_i$	i-tes Testset
$\mathcal{S}_{\textbf{Train}}$	Trainingsset
$\mathbf{T}$	Transduktor $\mathbf{T}$
g	Benutzerziel (User Goal)
l	(allg.) semantische Ebene, meist Wortklasse$l$
$\mathbf{l}$	Folge von Wortklassen $\mathbf{l} = l_1, l_2, \ldots, l_n$
$\mathcal{L}$	(allg.) Annotationsmenge, meist Wortklassenmenge $\mathcal{L}$
$\mathcal{W}$	Wörter $w \in \mathcal{W}$ in einem Lexikon
$\mathcal{S}$	Zustandsmenge (allg.)
$N(\mathcal{O})$	Anzahl der möglichen Beobachtungen
$N(\mathcal{S})$	Anzahl der möglichen Zustände
$\mathcal{A}$	Aktionsmenge
$N(d)$	Anzahl der Dialoge
Ns	Anzahl der semantischen Slots
$\mathbf{A}$	Aktionsvektor abgeleitet aus $\mathcal{A}$
$\mathbf{b}$	Belief Vektor
$\mathcal{O}$	Beobachtungsmenge
$\mathbf{B}(s, o)$	Beobachtungswahrscheinlichkeitsmatrix
Δt	Zeitdifferenz $t_2 - t_1$
$\mathcal{A}$	Menge möglicher Dialogaktionen
$\mathbf{R}(s, s')$	Beschreibt die direkte Belohnung eines Dialogschrittes.
$\mathbf{T}(s, a, s')$	Übergangswahrscheinlichkeit, eines Zustandes $s_i \mapsto s_j$
$\mathcal{S}$	Menge der Zustände in einem für den Dialog definierten Zustandsraum

λ	Discountfaktor
$E_{\textbf{Sim}}$	simulierter Fehler zum Test
$E_{\textbf{Sys}}$	Fehler des Systems
Γ	Kelleralpabet
$\#$	Anfangssymbol im Keller $\# \in \Gamma$
$\mathbf{M}$	Kellerautomat $\mathbf{M}$
$\mathcal{F}$	Menge an Endzuständen $\mathcal{F} \in \mathcal{S}$
$\mathcal{M}$	Modell
k	Normierungsfaktor
o	Beobachtungen
$\mathbf{O}$	Beobachtungsmatrix
π	Berechnungsvorschrift (*Policy*)
P	POMDP Modell
$V(s)$	Wertfunktion der Rewards
$\mathbf{R}(s, a, s')$	Belohnungsmatrix
$\mathcal{S}_{\textbf{train}}$	Trainingsdatensatz
$\boldsymbol{t}$	Zeitliche Variable t
T	Zeitpunkt T des Satzendes
t	Theshold
$\mathbf{T}$	Zustandsübergangsmatrix
$\mathbf{T}(s, a, s')$	Zustandsübergangswahrscheinlichkeitsmatrix
$\mathcal{S}$	Zustandsmenge
s	aktueller Zustand
s'	neuer Zustand

Index

Literaturverzeichnis

[Ada79] **Adams**, D.: *The Hitchhiker's Guide to the Galaxy*. MacMillan, 1979

[AH08] **Al-Hames**, M.: *Graphische Modelle in der Mustererkennung*. Technische Universität München, 2008 (Dissertation)

[All95] **Allen**, J.: *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., 1995

[AW04] **Acero**, A. ; **Wang**, Y.: A Semantically Structured Language Model. In: *Special Workshop in Maui (SWIM)*, 2004

[BB05] **Bilmes**, J.A. ; **Bartels**, C.: *Graphical Model Architectures for Speech Recognition*. IEEE Signal Processing Society, 2005. – 89–100 S.

[BBW07] **Buss**, M. ; **Beetz**, M. ; **Wollherr**, D.: CoTeSys — Cognition for Technical Systems. In: *International Journal of Assistive Robotics and Mechatronics* 8 (2007), Nr. 4, S. 25–36

[BCPR03] **Bellifemine**, F. ; **Caire**, G. ; **Poggi**, A. ; **Rimassa**, G.: JADE A White Paper. In: *In search of innovation* 3 (2003), September, Nr. 3

[Bel57] **Bellmann**, R.E.: *Dynamic programming*. Princeton Univ. Press, 1957. – 342 S.

[BGWL01] **Barras**, C. ; **Geoffrois**, E. ; **Wu**, Z. ; **Liberman**, M.: Transcriber: development and use of a tool for assisting speech corpora production. In: *Speech Communication* 33 (2001), Nr. 1–2

[Bila] **Bilmes**, J.A.: *Graphical Models*. – Umdruck zu Vorlesung EE512 Sommer 2006, University of Washington, Department of Electrical Engineering

[Bilb] **Bilmes**, J.A.: *Graphical Models in Speech and Language, and the Graphical Models Toolkit*. – Umdruck: The Center for Language and Speech Processing Workshop 2003 at Johns Hopkins University

[Bil98] **Bilmes**, J.A.: *A Gentle tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*. U.C. Berkely, TR-97-02, 1998

[Bil00] **Bilmes**, J.A. ; **UAI** (Hrsg.): *Dynamic Bayesian Multinets*. 2000

[Bil04] **Bilmes**, J.A.: Graphical Models and Automatic Speech Recognition. In: **Johnson**, M. (Hrsg.) ; **Khudanpur**, S.P. (Hrsg.) ; **Ostendorf**, M. (Hrsg.) ; **Rosenfeld**, R. (Hrsg.): *Mathematical Foundations of Speech and Language Processing*. Springer-Verlag, 2004, S. 191–246

[BK08] **Brüggemann-Klein**, A.: *Elektronsiches Publizieren*. 2008. – Umdruck zur Vorlesung , TU München, Fakultät für Informaitk

[BKW02] **Brüggemann-Klein**, A. ; **Wood**, D.: The Parsing of Extended Context-Free Grammars. In: *HKUST Theoretical Computer Science Center Research Report*, 2002

[BPSW70] **Baum**, L.E. ; **Petrie**, T. ; **Soules**, G. ; **Weiss**, N.: A Maximization Technique Occurring in The Statistical Analysis of Probabilistic Functions of Markov Chains. In: *The Annals of Mathematical Statistics* 41 (1970), Nr. 1, S. 164 – 171

[BW00] **Både**, L. ; **Westergren**, B. ; **Vachenauer**, P. (Hrsg.): *Mathematische Formeln und Tabellen*. Springer, 2000

[BZ02] **Bilmes**, J. ; **Zweig**, G.: The Graphical Model Toolkit: An Open Source Software System for Speech and Time-Series Processing. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002

[Cas05] **Cassandra**, A.: *The POMDP Page*. 2003-2005

[CM01] **Cheyer**, A. ; **Martin**, D.: The Open Agent Architecture. In: *Journal of Autonomous Agents and Multi-Agent Systems* 4 (2001), March, Nr. 1, S. 143–148. – OAA

[DG93] **Dowding**, J. ; **Gawron**, J.: Gemini: a natural language system for spoken language understanding. In: *Proceedings of the Spoken Language Systems Technology Workshop*, MIT Press, Cambridge, Massachusetts, 1993

[DLR77] **Dempster**, A.P. ; **Laird**, N.M. ; **Rubin**, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society B* 39 (1977), Nr. 1, S. 1–38

[Faw06] **Fawcett**, T.: An introduction to ROC analysis. In: *Pattern Recognition Letters*, 2006, S. 861–874

[FH03] **Friedman-Hill**, E.: *Jess in Action Java Rule-based Systems*. 2003. – 480 S. – ISBN: 1930110898

[Fin03] **Fink**, G.A.: *Mustererkennung mit Markov-Modellen*. B.G. Teubner, 2003 (Leitfäden der Informatik)

[For82] **Forgy**, C.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In: *Artificial Intelligence* Bd. 19, 1982, S. 17–37

[GH96] **Geiger**, D. ; **Heckerman**, D.: Knowledge representation and inference in similarity networks and Bayesian multinets. In: *Artificial Intelligence*, 1996, S. 45 – 74

[Gha98] **Ghahramani**, Z.: Learning Dynamic Bayesian Networks. In: **Giles**, C.L. (Hrsg.) ; **Gori**, M. (Hrsg.): *Adaptive Processing of Sequences and Data Structures*. Springer-Verlag, 1998 (Lecture Notes in Artificial Intelligence), S. 168–197

[GJ97] **Ghahramani**, Z. ; **Jordan**, M.: Factorial Hidden Markov Models. In: *Machine Learning, Special issue on learning with probabilistic representations 29*, 1997, S. 245–273

[Glu60] **Glushkov**, V. M.: On Synthesis Algorithm for Abstract Automata. In: *Ukr. Mathem. Zhurnal* 12 (1960), Nr. 147–156, S. 2

[Grü06] **Grütter**, R.: Software-Agenten im Semantic Web. In: *Informatik Spektrum* 29 (2006), Nr. 1

[Hec98] **Heckerman**, D. ; **Jordan**, M.I. (Hrsg.): *Learning in graphical models*. Kluwer, 1998

[HGD90] **Hemphill**, C. ; **Godfrey**, J. ; **Doddington**, G.: The ATIS Spoken Language Systems Pilot Corpus. In: *in Proceedings of the DARPA Speech and Natural Language Workshop*, Morgan Kaufmann, 1990, S. 96–101

[HLR08] **Hsu**, D. ; **Lee**, W.S. ; **Rong**, N.: What makes some POMDP problems easy to approximate? In: **Platt**, J.C. (Hrsg.) ; **Koller**, D. (Hrsg.) ; **Singer**, Y. (Hrsg.) ; **Roweis**, S. (Hrsg.): *Advances in Neural Information Processing Systems 20*. MIT Press, 2008, S. 689–696

[HNR68] **Hart**, P.E. ; **Nilsson**, N.J. ; **Raphael**, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In: *IEEE Transactions on Systems Science and Cybernetics SSC4 (2)*, 1968, S. 100–107

[HU00] **Hopcraft**, J.E. ; **Ullman**, J.D.: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. 4. Oldenburg-Verlag, 2000

[HY03] **He**, Y. ; **Young**, S.: A Data-Driven Languag Understanding System. In: *Proceedings ASRU*, 2003

[HY05] **He**, Y. ; **Young**, S.: Semantic Processing Using the Hidden Vector State Model. In: *Computer Speech and Language* 19 (2005), Nr. 4, S. 85–106

[JGJS01] **Jordan**, M.I. ; **Ghahramani**, Z. ; **Jaakkola**, T.S. ; **Saul**, L.K.: An Introduction to Variational Methods for Graphical Models. In: **Jordan**, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 105 – 161

[JLO90] **Jensen**, F.V. ; **Lauritzen**, S. ; **Olesen**, K.: Bayesian Updating in Causal Probabilistic Networks by Local Computations. In: *Computational Statistics Quaterly* 4 (1990), S. 269–282

[Joh] **Johnson**, S.C.: *Yacc: Yet Another Compiler-Compiler*, `http://dinosaur.compilertools.net/`

[Jor01] **Jordan**, M.I. (Hrsg.): *Learning in Graphical Models*. Second Printing. MIT Press, 2001

[JS04] **Jelinek**, L. ; **Smidl**, L.: Timetable Inquiry Processing Based on N-gram Language Models and Finite State Transducers. In: *Proceedings of the 8th world multi-conference on systemics, cybernetics and informatics*, 2004

[Kat87] **Katz**, S.M.: Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. In: *IEEE Trans. on Acoustics, Speech, and Signal Processing* 35 (1987), Nr. 3, S. 400 – 401

[KC68] **Kubrick**, S. ; **Clarke**, A.C.: *A Space Odyssey*. National Publishers, Inc., 1968

[KHL08] **Kurniawati**, H. ; **Hsu**, D. ; **Lee**, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Proceedings Robotics: Science and Systems*, 2008

[KLC95] **Kaelbling**, L.P. ; **Littman**, M.L. ; **Cassandra**, A.R.: Planning and Acting in Partially Observable Stochastic Domains. In: *Artificial Intelligence* 101 (1995), S. 99–134

[KLC98] **Kaelbling**, L.P. ; **Littman**, M.L. ; **Cassandra**, A.R.: Planning and acting in partially observable stochastic domains. In: *Artificial Intelligence* Bd. 101, 1998, S. 99–134

[Kro01] **Kromrey**, H.: Evaluation - ein vielschichtiges Konzept. Begriff und Methodik von Evaluierung und Evaluationsforschung. Empfehlungen für die Praxis. In: *Sozialwissenschaften und Berufspraxis (SUB)* Bd. 24. 2001, S. 105 – 131

[Lau96] **Lauritzen**, S.L.: *Graphical Models*. Oxford Science Publications, 1996 (Oxford Statistical Science Series)

[LBHH04] **Larsson**, S. ; **Bernman**, A. ; **Hallenborg**, J. ; **Hjelm**, D.: Specification, Interaction and Reconfiguration in Dialogue Understanding Systems. In: *Trindikit Manual*, 2004, S. 111

[LC08] **Lämmel**, U. ; **Cleve**, J.: *Künstliche Intelligenz*. Carl Hanser, 2008

[Lie03] **Lieb**, R.: Natürlichsprachliche Dialogführung im Auto / Lehrstuhl für Mensch-Maschine-Kommunikation, TU München. 2003. – Forschungsbericht

[Lie06] **Lieb**, R.: *Effiziente einstufige Verarbeitung hierarchisch gegliederter Wissensquellen in Sprachinterpretationssystemen*, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diss., 2006

[Lin86] **Linguarum**, J.: *Die Lautgestalt der Sprache*. Mouton de Gruyter, 1986 (Maior 75)

[LS] **Lesk**, M.E. ; **Schmidt**, E.: *Lex - A Lexical Analyzer Generator*, `http://dinosaur.compilertools.net/`

[MB95] **Miller**, S. ; **Bates**, M.: Recent progress in hidden understanding models. In: *Proceedings of the ARPA Spoken Language Systems Technology Workshop*, Morgan Kaufmann Publishers, Palo Altos, California, 1995, S. 22–25

[McT04] **McTear**, M.F.: *Spoken Dialogue Technology*. Springer, 2004

[MG76] **Markel**, J.D. ; **Gray**, A.H.: Linear Prediction of Speech. In: *IEEE Trans. on Acoust., Speech, and Sign. Proc* Bd. 24, 1976

[Mur01] **Murphy**, K.: *Dynamic Bayesian Netoworks: Representation, Inference and Learning*, University of California, Diss., 2001

[Nor95] **Normadin**, Y.: Optimal Splitting of HMM Gaussian Mixture Components with MMIE Training. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1995, S. 449–452

[NR94] **Neukirchen**, C. ; **Rigoll**, G.: Ein systematischer Vergleich von diskreten, kontinuierlichen und hybriden HMM-basierten Systemen zur Spracherkennung. In: *Elektronische Sprachsignalverarbeitung, Tagungsband*, 1994, S. 83–89

[Pea88] **Pearl**, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Revised second printing. Morgan Kaufmann Publishers, San Fransisco, California, 1988

[Pea01] **Pearl**, J.: *Causality: Models, Reasoning, and Inference*. Second Reprint with corrections. Cambridge University Press, 2001

[Phi95] **Philips**, C.: Right Association in Parsing and Grammar. In: **Schütze**, C. (Hrsg.) ; **Broihier**, K. (Hrsg.) ; **Ganger**, J. (Hrsg.): *Papers on Language Processing and Acquisition*, 1995 (MITWPL 26), S. 57

[PT92] **Pieraccini**, R. ; **Tzoukermann**, E.: Progress report on the chronus system: ATIS benchmark results. In: *Proceedings of the DARPA Speech and Natural Language Workshop*, Morgan Kaufmann Publishers, Los Altos, California, 1992

[PTG+92] **Pieraccini**, R. ; **Tzoukermann**, E. ; **Gorelov**, Z. ; **Levin**, E. ; **Lee**, C-H. ; **Gauvain**, J-L.: Progress report on the Chronus system: ATIS benchmark results. In: *Proceedings 5th DARPA Workshop on Speech and Nat. Lang.*, 1992

[Qui67] **Quillian**, M. R.: *Word concepts. A theory and simulation of some basic semantic capabilities*. Behavioral Science 12, 1967. – 410–430 S.

[Rab89] **Rabiner**, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* 77 (1989), Nr. 2, S. 257–285

[Rec94] **Rechenberg**, I.: *Evolutionsstrategie '94*. Frommann Holzboog, 1994. – 434 S.

[Rüg99] **Rüger**, B.: *Test- und Schätztheorie*. Oldenbourg, 1999 (Band I: Grundlagen)

[Rij79] **Rijsbergen**, C.J: *Information Retrieval*. Butterworths, 1979

[RJ93] **Rabiner**, L.R. ; **Juang**, B.H.: Theory and Implementation of Hidden Markov Models. In: *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1993, S. 321 – 389

[RN03] **Russel**, S. ; **Novig**, P. ; **Auflage**, 2. (Hrsg.): *Künstliche Intelligenz*. Pearson Education, 2003

[Rus94] **Ruske**, G.: *Automatische Spracherkennung: Methoden der Klassifikation und Merkmalsextraktion*. 2nd, extended. Oldenbourg-Verlag, 1994

[SR07] **Schenk**, J. ; **Rigoll**, G.: Vorlesungsskript Mensch-Maschine-Kommunikaiton 1 / Lehrstuhl für Mensch-Maschine-Kommunikation. 2007. – Forschungsbericht

[SSR08a] **Schenk**, J. ; **Schwärzler**, S. ; **Rigoll**, G.: Discrete Single vs. Multiple Stream HMMs: A Comparitive Evaluation of theri Use in On-LIne Handwriting Recogniton of Whiteboard Notes. In: *Int. Conf. on Frontiers In Handwriting Recognition*, 2008, S. 550–555

[SSR08b] **Schenk**, J. ; **Schwärzler**, S. ; **Rigoll**, G.: PCA in On-Line Handwriting Recognition of Whiteboard Notes: A Novel VQ Design for Use With Discrete HMMs. In: *Int. Conf. on Frontiers in Handwriting Recognition*, 2008, S. 544–549

[SSRR08] **Schenk**, J. ; **Schwärzler**, S. ; **Ruske**, G. ; **Rigoll**, G.: Novel VQ Designs for Discrete HMM ON-Line Handwritten Whiteboard Recognition. In: **Rigoll**, Gerhard (Hrsg.): *Proceedings of the 30th Symposium of DAGM*, Springer, 2008 (LNCS 5096), S. 234 – 243

[SSWR08] **Schwärzler**, S. ; **Schenk**, J. ; **Wallhoff**, F. ; **Ruske**, G.: Natural Language Understanding By Combining Statistical Methods And Extended Context-free Grammars. In: **Rigoll**, Gerhard (Hrsg.): *Proceedings of the 30th Symposium of DAGM*, Springer, 2008 (LNCS 5096), S. 254 – 263

[SGS+08] **Schwärzler**, S. ; **Geiger**, J. ; **Schenk**, J. ; **Al-Hames**, M. ; **Hörnler**, B. ; **Ruske**, G. ; **Rigoll**, G: Combining Statistical And Syntactical Systems For Spoken Language Understanding With Graphical Models. In: *Proceedings of the 9th International Speech Communication Association (Interspeech 2008)*, 2008, S. 1590–1593

[SBG+08] **Schwärzler**, S. ; **Bannat**, A. ; **Gast**, J. ; **Wallhoff**, F. ; **Mayer**, C. ; **Wimmer**, M. ; **Wendt**, C. ; **Schmidt**, S. ; **Giuliani**, M. ; **Kassecker**, M.: MuDiS - A Multimodal Dialogue System for Human-Robot Interaction. In: *In: Proceedings 1st Intern. Workshop on Cognition for Technical Systems, Cotesys 2008* Technische Universität München, 2008

[SRWR09] **Schwärzler**, S. ; **Ruske**, G. ; **Wallhoff**, F. ; **Rigoll**, G.: Using Graphical Models for an Intelligent Mixed-initiative Dialog Management System. In: *Proceedings of the 13th International Conference on Human-Computer Interaction*, Springer, 2009. – angenommen

[SSRW09] **Schwärzler**, S. ; **Schenk**, J. ; **Ruske**, G. ; **Wallhoff**, F.: A Multi-Agent Framework For A Hybrid Dialog Management System. In: *Int. Conf. on Multimedia & Expo ICME 2009* IEEE, 2009. – angenommen

[SMS+09] **Schwärzler**, S. ; **Maier**, S. ; **Schenk**, J. ; **Wallhoff**, F. ; **Rigoll**, G.: Using Graphical Models for Mixed-Initiative Dialog Management Systems with Realtime Policies. In: *Proceedings of the 10th International Speech Communication Association (Interspeech 2009)*, 2009. – angenommen

[SB05] **Sutton**, R. ; **Barto**, A.S.: *Reinforcement Learning: An Introduction*. MIT Press, 2005

[SBBW04] **Si**, J. ; **Barto**, A.G. ; **B.**, W.B. Powell W. ; **Wunsch**, D. ; **Fogel**, David B. (Hrsg.): *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press on Series on Computational Intelligence, 2004. – 644 S.

[Sch03] **Schulz**, R.H.: *Codierungstheorie. Eine Einführung*. 2. Auflage. Vieweg Verlag, 2003. – ISBN 3-528-16419-0

[Sen92] **Seneff**, S.: TINA: a natural language system for spoken language applications. In: *Computational Linguistics* 18 (1992), Nr. 1

[SS73] **Smallwood**, R. ; **Sondik**, E.: The optimal control of partially observable Markov processes over a finite horizon. In: *Operations Research* Bd. 21, 1973, S. 1071–1088

[ST95] **Schukat-Talamazzini**, E.G: *Automatische Spracherkennung*. Vieweg Verlag, 1995

[Sta73] **Stachowiak**, H.: *Allgemeine Modelltheorie*. Springer, Wien, 1973. – 494 S. – 3211811060

[Tah07] **Taha**, T.: POMDP Toolbox / Clemson University, Clemson, USA. Version: 2007. `www.tarektaha.com`. 2007. – Forschungsbericht

[Tho06] **Thomae**, M.: *Hierarchische Sprachmodelle für die 1-stufige stochastische Interpretation fließender Sprache*, TU München, Fakultät für Elektrotechnik und Informationstechnik, Diss., 2006

[Vit67] **Viterbi**, A.: Error Bounds for Convolutional Codes and an Asymptotically OptimumError Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In: *IEEE Transactions on Information Theory*, 1967 (13), S. 260–267

[WA05] **Wang**, Y. ; **Acero**, A.: SGStudio: Rapid Semantic Grammar Development for Spoken Language Understanding. In: *Proceedings Interspeech*, 2005

[WDA05] **Wang**, Y. ; **Deng**, L. ; **Acero**, A.: Spoken Language Understanding - An Introduction to the statistical framework. In: *IEEE Signal Processing Magazine* 22 (2005), S. 16–31

[Wei99] **Weiß**, Gerhard (Hrsg.): *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999

[WI94] **Ward**, W. ; **Issar**, S.: Recent improvements in the CMU spoken language understanding system. In: *Proceedings of ARPA Human Language Technology Workshop*, 1994, S. 213–216

[WY06] **Williams**, J. ; **Young**, S.: Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI). In: *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, 2006

[YEH+02] **Young**, S. ; **Evermann**, G. ; **Hain**, T. ; **Kershaw**, D. ; **Moore**, G. ; **Odell**, J. ; **Ollason**, D. ; **Povey**, D. ; **Valtchev**, V. ; **Woodland**, P.: *The HTK Book (for HTK Version 3.2.1)*. Cambridge University Engineering Department, 2002

[You02] **Young**, S.: Talking to Machines (Statistically Speaking). In: *Proceedings ICSLP*, 2002

[You06] **Young**, S.: Using POMDPs for dialog management. In: *IEEE/ACL Workshop*, 2006

[YY06] **Ye**, H. ; **Young**, S.: A Clustering Approach to Semantic Decoding. In: *Proceedings Interspeech*, 2006

[ZMK09] **Zettlemoyer**, L. ; **Milch**, B. ; **Kaelbling**, L.P.: Multi-Agent Filtering with Infinitely Nested Beliefs. In: **Koller**, D. (Hrsg.) ; **Schuurmans**, D. (Hrsg.) ; **Bengio**, Y. (Hrsg.) ; **Bottou**, L. (Hrsg.): *Advances in Neural Information Processing Systems 21*. 2009, S. 1905–1912