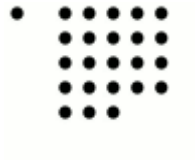

Kapitel 22 - Kommunikation / Kapitel 23 - Probabilistische Sprachverarbeitung

Autoren:
Martin Rau Matrikel-Nummer: 11045620
Christian Witte Matrikel-Nummer: 11048070

Inhaltsverzeichnis

22 Kapitel 22 – Kommunikation

Unter **Kommunikation** wird ganz allgemein, der bewusste Austausch von **Informationen** verstanden. Dieser findet durch das Senden und Wahrnehmen von **Signalen** statt. In der Tierwelt wird von den meisten Tieren ein gemeinsamer „**Signalpool**“ verwendet um die nötigsten Informationen (Wo gibt es Nahrung, wo lauert Gefahr, Rückzug, etc.) auszutauschen. Im Gegensatz dazu ist die Kommunikation zwischen Menschen oder auch Agenten wesentlich komplizierter, da es nicht nur einfache Signale sondern verschiedene Intentionen, Mehrdeutigkeiten, Grammatiken und Gesprächsverläufe gibt.
In diesem Kontext stellen sich zwei wichtige Fragen, bezüglich der Kommunikation zwischen Agenten:

- Warum tauschen Agenten informationshaltige Nachrichten aus?
- Wie tauschen Agenten informationshaltige Nachrichten aus?

Der Hauptvorteil im Nachrichtenaustausch zwischen Agenten besteht darin, dass in einer partiell beobachtbaren Welt die Kommunikation dem Agenten helfen kann richtige Entscheidungen zu treffen. Er hat so die Möglichkeit, durch Kommunikation, sein Wissen mit dem Wissen anderer Agenten zu teilen. Eine Anwendung dafür wäre zum Beispiel, das Agieren in Multi-Agenten Umgebungen um gemeinsame Pläne durchzuführen (siehe Kapitel 12).

22.1 Kommunikation als Aktion

Eine mögliche Aktion die einem Agenten zur Kommunikation zur Verfügung steht ist der Sprachakt, der zur Erzeugung von Sprache dient. Wobei Sprache in diesem Fall allgemein gehalten wird („freie Sprache“), hierzu zählen auch E-Mails, Reden, Luftschreiber, etc. Während ein Sprechakt sich nur auf das „Reden“ bezieht, bei dem durch Mund-bewegungen Luftmoleküle zum vibrieren gebracht werden.

Wichtige Grundbegriffe zum Verstehen des Kommunikationsmodells sind:

- Sprecher (Sender), Zuhörer (Empfänger), Äußerung (Nachricht), Wort

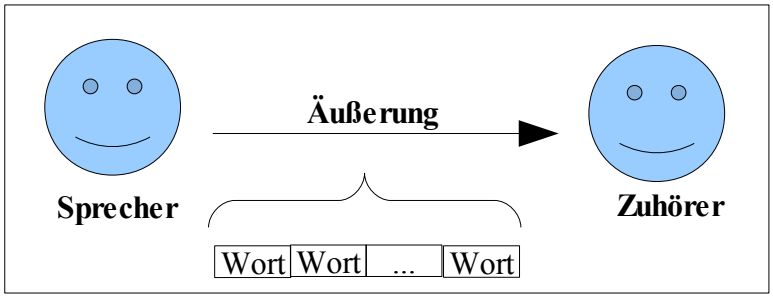

Abbildung 22.1: Vereinfachtes Kommunikationsmodell

In dem stark vereinfachten Kommunikationsmodell wird die Rollenverteilung verdeutlicht. Der Sprecher tätigt eine Äußerung (besteht aus mehreren Wörtern), um diese an den Zuhörer zu schicken, der diese daraufhin wahrnimmt. Was genau bei Sprecher und Zuhörer dafür nötig ist, bzw. was passieren muss, damit dieser simple Kommunikationsvorgang klappt wird in diesem Kapitel erläutert.
Der Sprachakt ist nur eine von vielen möglichen Aktionen, die ein Agent zum Übermittlungsvorgang nutzen kann. Weitere wichtige Aktionen sind:

- andere Agenten über die Welt **abfragen**
  - Beispiel „Hast du irgendwo das Wumpus gerochen?“
- einander über die Welt **informieren**
  - Beispiel „Auf Quadrat 3,4 ist es windig.“ oder als Antwort einer Frage
- sich gegenseitig zu Aktionen **auffordern** (Anweisungen, Direktiven)
- Anforderungen **bestätigen**
  - Beispiel „OK“
- sich gegenseitig etwas **versprechen** oder einem Plan **zustimmen**
  - Beispiel „Ich erschieße das Wumpus, du holst das Gold“

Bei der Aktion „Auffordern“ muss man noch die unterschiedlichen Stärken einer Aufforderung beachten

- freundlicher Ton: durch Formulierung der Aufforderung als Aussage oder Frage (**indirekter Sprachakt**)
  - Beispiel „Ich könnte jemanden gebrauchen, der mir hilft das Gold zu tragen“
- autoritärer Ton: Befehle geben
  - Beispiel „Helft mir beim Gold tragen“
- starker Ton: Aufforderung als Drohung mitteilen
  - Beispiel: „Helft mit beim Gold tragen, sonst...“

Sprachakte können je nach Inhalt und gewählte Aktion verschiedene Auswirkungen auf die Umgebung oder auf den Agenten haben:

- können den mentalen Zustand des Agenten beeinflussen
- die zukünftigen Aktionen eines Agenten lenken oder abändern
- Entscheidungen treffen
- Aktionen ausführen
- direkten Einfluss auf die Welt (deklarative Sprachakt) haben

Der deklarative Sprachakt tritt meistens nur auf, wenn bestimmten äußere Zustände und Konventionen dazu führen, dass sich der entsprechende Teil der Welt ändert. So würde der Satz „*Ich erkläre Sie zu Mann und Frau*“ in unserer Kultur dazu führen ein Paar zu verheiraten.
Das größte Problem eines Sprachakts liegt im verstehen und interpretieren der Nachricht. Dies ist zu vergleichen mit anderen Verständnisproblemen, zum Beispiel das Interpretieren eines Bildes „*Was will der Künstler uns mit diesem Bild sagen?*“. Die übermittelte Nachricht ist in fast allen Fällen mehrdeutig. Um diese Mehrdeutigkeit zu lösen, muss man wissen welche Hintergrundinformationen dazu geführt haben, und muss diese zusätzlich mit dem vorhandenen Wissen kombinieren. Es muss also ein Plan erstellt werden, um die Sprachakte zu verstehen.

22.1.1 Grundlagen der Sprache

Es gilt zunächst mal die zwei Hauptsprachtypen voneinander zu unterschieden. Zum einen wären da die **formalen Sprachen** (wie zum Beispiel die FOL oder Java). Sie sind streng formal definiert anhand von mathematischen Definitionen. Eine formale Sprache umfasst immer eine bestimmte Menge von **Zeichenketten**. Jede Zeichenkette ist dabei eine Verknüpfung von **Terminalsymbolen** (Wörter). Am Beispiel der Aussagelogik soll dieser Zusammenhang nochmals kurz erläutert werden:

Zu den Terminalsymbolen gehören : P, Q, $\wedge$, aus diesen lässt sich dann eine Zeichenkette: $P \wedge Q$ bilden. Die Zeichenkette $PQ\wedge$ gehört nicht zur Sprache, da sie nicht mit der formal definierten Definition übereinstimmt.

Des weiteren gibt es noch den Sprachtyp der **natürliche Sprachen**, dazu gehören zum Beispiel Deutsch oder Spanisch. Diese haben keine strengen Definitionen, obwohl Linguisten versuchen die Grammatik so zu codieren wie sie ist. Dieses Vorhaben ist aber alleine schon Aufgrund des Sprachumfangs sehr schwer und umständlich. Im Gegensatz dazu gibt es die Grammatiker, die vorschreiben wie eine Sprache sein soll („trenne *nie s-t*"), um so eine Struktur in die Sprache zu bringen.

Jede Sprache hat eine eigene **Grammatik**, die eine endliche Menge von Regeln zur Sprachspezifikation angibt. Formale Sprachen verfügen über eine offizielle Grammatik, die unbedingt eingehalten werden muss, während die Grammatiken für natürliche Sprachen nicht so streng sind. So kann man einen Satz wie „Haus groß sein" ohne weiteres verstehen, während ein umgedrehter mathematischer Ausdruck in einer formalen Sprache zu Fehlern führen würde.

Jede Zeichenkette verfügt zudem über eine **Semantik**, die der Zeichenkette eine sinngemäße Bedeutung hinzufügt. In der Sprache der Arithmetik hat der Ausdruck „$x + 4$" die Bedeutung der Summe von x und 4.

Bei den natürlichen Sprachen kommt zusätzlich zur Semantik noch die **Pragmatik** hinzu die jede Zeichenkette besitzt. Sie gibt die tatsächliche Bedeutung einer Zeichenkette im Zusammenhang mit einer bestimmten Situation an.

Bei der Erstellung einer Grammatik hilft das **Konzept der Phrasenstruktur**. Die

Zeichenketten werden dabei in Unterzeichenketten (**Phrasen**) zerlegt, die sich dann in verschiedene Kategorien einordnen lassen. „*Das Wumpus*“ oder „Der Agent“ werden dabei als **Substantiv-Phrase** (Noun Phrase, NP) eingestuft. Die Substantiv-Phrase kann dann mit einer **Verb-Phrase** (VP) kombiniert werden, um eine Phrase der Kategorie **Satz** (S) zu erzeugen. Dieses Konzept bringt zwei Vorteile mit sich: Zum Einen entsprechen die Phrasen ihren natürlichen semantischen Elementen (z.B. NP = Objekte) wodurch die Bedeutung einer Äußerung konstruiert werden kann. Zum Anderen helfen die Phrasenkategorien eine Sprache zu beschreiben.

Für die Umschreibungsregeln der Grammatik, werden zusätzlich zu den Terminalsymbolen noch **Non-Terminalsymbole** verwendet. Ein simples Beispiel für Non-Terminalsymbole wären die Phrasen NP, VP und S. Diese werden dann in Regeln verwendet, die in der Backus-Nauer-Form (BNF) angegeben sind:

„S → NP VP“ besagt zum Beispiel, dass man ein S Nonterminal durch ein NP und VP Nonterminal ersetzen kann. Diese können dann ebenfalls ersetzt werden, allerdings in diesem Fall durch Terminalsymbole, zum Beispiel „*Das Wumpus (NP) lebt (VP)*“.

Jede Grammatik bekommt eine bestimmte **generative Kapazität** zugeordnet, die Auskunft darüber gibt welche Sprachmenge die Grammatik repräsentieren kann. Chomsky[1] erstellte vier Klassen für diese Unterteilung, die sich lediglich im Aufbau der Umschreibungsregeln unterscheiden. Diese Klassenanordnung ist streng hierarchisch, wobei jede höher angeordnete Klasse die volle Leistungsfähigkeit der niedrigeren Klassen umfasst.

- rekursiv aufzählbare Grammatik (ganz oben in der Hierarchie)
  - keine Regelbeschränkungen
  - beide Seiten der Regeln können beliebig viele Nonterminale und Terminalsymbole enthalten
  - äquivalent zu Turing Automaten
- kontextsensitive Grammatiken
  - einzige Beschränkung ist, dass auf der rechten Seite immer mindestens so viele Symbole wie auf der linken Seite stehen müssen (keine Löschregeln möglich)
  - kontextsensitiv lässt sich anhand der Regel ASB → AXB erklären (S im Kontext

1 Noam Chomsky (* 1928), US-amerikanischer Sprachwissenschaftler

eines vorausgehenden A und nachfolgenden B wird als X umschrieben)

- kontextfreie Grammatiken (CFGs)
  - linke Seite besteht immer aus einem einzigen Nonterminalzeichen
  - jede Regel erlaubt die Umschreibung der rechten Seite in jedem beliebigen Kontext
  - die meisten Programmiersprachen benutzen CFGs
- reguläre Grammatiken
  - jede Regel hat ein einziges Nonterminal-Symbol auf der linken Seite und ein Terminal-Symbol optional gefolgt von einem Nonterminal-Symbol auf der rechten Seite
  - äquivalent mit endlichen Automaten

22.1.2 Einzelkomponenten der Kommunikation

Ein typischer Kommunikationsvorgang entsteht, wenn Sprecher S dem Hörer H einen Sachverhalt P unter Verwendung von Wörtern W mitteilen. Dieser in der unteren, vereinfachten Skizze erwähnte Vorgang setzt sich aus insgesamt sieben Prozessen zusammen.

1. Absicht (Intention)
   Irgendwie beschließt der Sprecher, dass ein Sachverhalt einem Hörer mitgeteilt werden sollte. Zum Beispiel könnte ein Agent davon ausgehen, dass es andere Agenten interessiert, wenn ein Wumpus tot ist.
2. Erzeugung (Generation)
   Der Sprecher erstellt einen Plan, wie er den Sachverhalt in eine Äußerung umwandeln kann. Dieser Vorgang scheint simpel, ist er aber überhaupt nicht, da die Generation so geschehen muss, dass der Hörer die Äußerung möglichst gut ableiten kann. Er darf sie auf keinen Fall falsch verstehen, dies würde zu Missverständnissen oder dem erneuten Senden einer ähnlichen Nachricht führen.
3. Synthese

Beschäftigt sich mit der physikalischen Darstellung der Wörter W, diese werden durch die physische Realisierung als W' dargestellt. Dieser Prozess kann über verschiedene Medien erfolgen (Luft, Papier, Schall, Steine, etc.).

4. Wahrnehmung

   Der Hörer H nimmt die physische Realisierung von W' als W_2' wahr und codiert diese dann als W_2.

5. Analyse

   Der Hörer leitet ab, dass W_2 die möglichen Bedeutungen $P_1,...,P_n$ haben kann. Die Analyse besteht aus drei weiteren Prozessen:

   I. Parsing

      Für eine Eingabekette wird ein Parse-Baum erzeugt, dessen innere Knoten Phrasen und dessen Blätter Wörter sind.

   II. Semantische Interpretation

      Die Bedeutung der Äußerung wird als Ausdruck einer Repräsentationssprache extrahiert. Hierbei werden Äußerungen mit mehreren Interpretationsmöglichkeiten als **mehrdeutig** bezeichnet.

   III. Pragmatische Interpretation

      Hier wird ein besonderer Fall der Mehrdeutigkeit berücksichtigt: dieselben Wörter können in unterschiedlichen Situationen andere Bedeutungen haben. Hier muss der Kontext und die Situation in der die Äußerung getätigt wurde betrachtet werden. So könnte zum Beispiel das Wort *„Wumpus*“ durch einen speziellen Wumpus ($Wumpus_1$ das tot ist) ersetzt werden. Die Konstante „*Jetzt*“ könnte durch die aktuelle Situation S_3 ersetzt werden.

6. Sicherstellung der Eindeutigkeit

   Im optimalen Fall gilt, dass Pi = P ist, dies ist allerdings nicht immer der Fall, da es für fast jede Äußerung mehrere Interpretationsmöglichkeiten gibt. Die Kommunikation funktioniert nur, weil sich der Hörer die Arbeit macht herauszufinden, welche Interpretation der Sprecher wahrscheinlich vermitteln wollte. Gibt es mehrere Interpretationsmöglichkeiten, so ist es Aufgabe der Sicherstellung der Eindeutigkeit, die wahrscheinlichste unter ihnen zu finden.

7. Aufnahme

Befasst sich mit dem Thema, wie der Hörer mit der Nachricht umgeht, er kann sie zum Beispiel glauben oder anzweifeln. Ein naiver Agent könnte alles glauben, es könnte aber auch möglich sein, das der Agent eine Rückfrage stellt um seine Entscheidung zu optimieren.

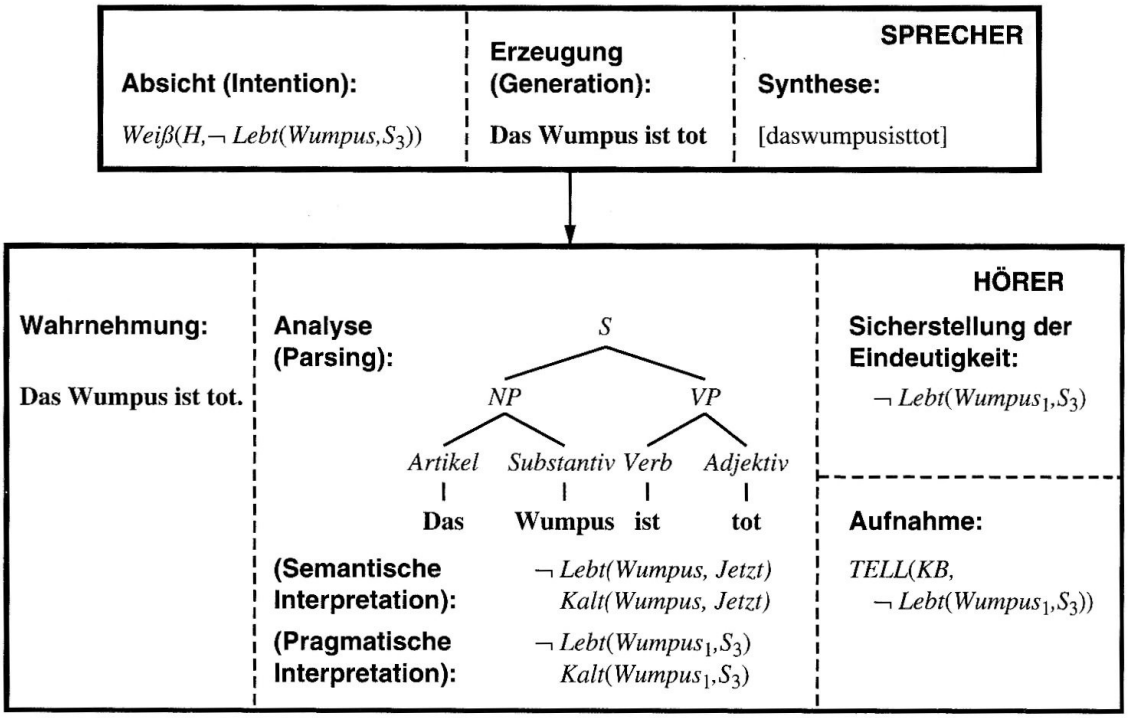

Abbildung 22.2: Sieben Kommunikationsprozesse am Satz "Das Wumpus ist tot"

22.2 Eine formale Grammatik für ein Fragment von Englisch

Als Beispiel zum Thema Grammatik soll nachfolgende eine formale Grammatik Englischε_0 beschrieben werden, mit der es möglich sein soll Aussagen über die Wumpuswelt zu treffen. Eine vollständige Grammatik Englisch abzuleiten ist im Grunde nicht möglich, da sie zum einen viel zu umfangreich wäre und es zudem noch unterschiedliche Meinungen gibt, was gültiges Englisch ist.

22.2.1 Das Lexikon ε_0

Das Lexikon gibt eine Liste von erlaubten Worten an, die in der Sprache gültig sind, diese sind nach Kategorien und Teilen der Sprache sortiert:

- Substantive, Pronomen und Namen bezeichnen Dinge
- Verben bezeichnen Ereignisse
- Adjektive modifizieren Dinge
- Adverbien modifizieren Ereignisse
- Artikel (z.B. der, die, das)
- Präpositionen (z.B. in)
- Konjunktionen (z.B. und)

Es gibt zwei Arten von Klassen in die man die Wörter unterteilen kann. Offene Klassen sind Substantive, Verben und ihre Modifikatoren. Diese haben sehr viele Mitglieder in den Klassen, es ist unmöglich alle aufzulisten, zudem kommen ständig neue hinzu (z.B: MP3). Geschlossene Klassen umfassen hingegen nur eine kleine Anzahl an Wörtern und könnten theoretisch komplett aufgelistet werden. Dieses liegt auch daran, dass sie sich nur sehr selten ändern, höchstens alle 100 Jahre. Zu den geschlossen Klassen zählen Pronomen, Artikel, Präpositionen und Konjunktionen.

Substantiv	→ stench | breeze | glitter | nothing | agent | wumpus | pit | pits | gold | east | ...
Verb	→ is | see | smell | shoot | feel | stinks | go | grab | carry | kill | turn | ...
Adjektiv	→ right | left | east | dead | back | smelly | ...
Adverb	→ here | there | nearby | ahead | right | left | east | south | back | ...
Pronomen	→ me | you | I | it | ...
Name	→ John | Mary | Boston | Aristotle | ...
Artikel	→ the | a | an | ...
Präpositionen	→ to | in | on | near | ...
Konjunktion	→ and | or | but | ...
Ziffer	→ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Abbildung 22.3: Beispiellexikon für die Sprache ε_0

Um eine Grammatik für die Sprache ε_0 zu erstellen müssen zunächst die Wörter des Beispiellexikons zu Phrasen kombiniert werden. Dazu werden fünf Nonterminal-Symbole

eingeführt:

- Satz (S)
- Substantivphrase (NP)
- Verbphrase (VP)
- Präpositionalphrase (PP)
- relative Klausel (RelClause, folgt einer NP und modifiziert diese)

S	→	NP VP	I + feel a breeze
	|	S Konjunktion S	I feel a breeze + and + I smell a wumpus
NP	→	Pronomen	I
	|	Name	John
	|	Substantiv	Pits
	|	Artikel Substantiv	the + wumpus
	|	Ziffer Ziffer	3 + 4
	|	NP PP	the wumpus + to the east
	|	NP RelClause	the wumpus + that is smelly
VP	→	Verb	stinks
	|	VP NP	feel + a breeze
	|	VP Adjektiv	is + smelly
	|	VP PP	turn + to the east
	|	VP Adverb	go + ahead
PP	→	Präposition NP	to + the east
RelClause	→	**that** VP	that + is smelly

Abbildung 22.4: Grammatik der Sprache ε_0

Diese Beispielgrammatik funktioniert allerdings nicht problemlos, da die Grammatik zu viele Sätze erzeugt, die grammatikalisch nicht korrekt sind (z.B: „*Me go Boston*“). Andererseits erzeugt sie auch zu wenig Sätze, wie zum Beispiel den Satz „*I think the Wumpus is smelly.*“, der nicht abgeleitet werden kann. Zudem gibt es noch einige Kleinigkeiten, wie das Problem der Großschreibung beim ersten Wort im Satz oder der

fehlende Punkt am Ende. Das letzte Problem liegt darin begründet, dass die Grammatik nicht für das geschriebene Wort entwickelt worden ist.

22.3 Syntaktische Analyse (Parsing)

Der Prozess des Parsings wurde bereits als Funktion beschrieben die einen Parse-Baum zu einer bestimmten Eingabekette findet. Die Funktion wird mit `PARSE(„the wumpus is dead“, ε0, S)` aufgerufen und liefert einen Parse Baum mit der Wurzel S zurück. Die Blätter der Parse-Baums sind „*the*“, „*wumpus*“, „*is*“ und „*dead*“, die inneren Knoten sind Nonterminal-Symbol der Grammatik ε_0.

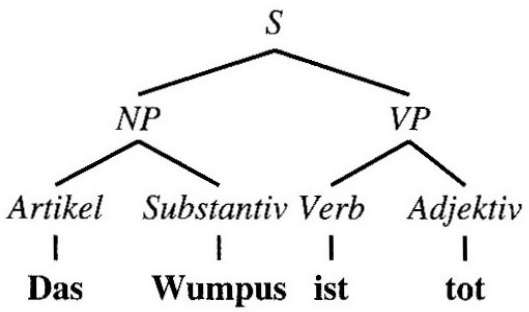

Abbildung 22.5: Beispiel Parse-Baum

Um den Suchraum zu erzeugen, gibt es zunächst zwei simple Möglichkeiten – **Top-Down-Parsing** und **Bottom-Up-Parsing**. Beim Top-Down-Parsing beginnt man mit dem Symbol S und sucht einen Baum der die Eingabewörter als Blätter hat. Beim Bottom-Up-Parsing ist die Vorgehensweise genau andersherum – man beginnt mit den Wörtern und sucht nach einem Baum mit der Wurzel S.

Top-Down-Parsing als Suchproblem:

Ausgangszustand:

Parse-Baum der nur aus der Wurzel S besteht und unbekannte Kinder hat. Jeder Zustand im Suchraum ist ein Parse-Baum.

Nachfolgefunktion:

Wählt den am weitesten links liegenden Knoten mit unbekannten Kindern im Suchraum aus. Dann wird in der Grammatik nach Regeln gesucht, die das Symbol des Suchknotens auf der linken Seite der Regel stehen haben. Diese Regel wird dann auf den Knoten angewendet und die unbekannten Kindern, werden durch eine Liste ersetzt, die rechts vom Pfeil in der entsprechenden Regel stehen.

Zieltest:

Hier wird nur überprüft, ob die Blätter des Parse-Baumes genau der Eingabezeichenkette entsprechen. Es darf keine unbekannten Kinder mehr geben.

Bottom-Up-Parsing als Suchproblem:

Ausgangszustand:

Die Liste von Wörtern aus der Eingabezeichenkette wird jeweils im Parse-Baum betrachtet, der nur aus einem Blatt Knoten besteht. Auch hier gilt, dass jeder Zustand im Suchraum ein Parse-Baum ist.

Nachfolgefunktion:

Betrachtet an jeder Position i die Liste der Bäume und an jeder rechten Seite des Baumes eine Grammatikregel. Stimmt die Folge der Liste von Bäumen beginnend mit i mit der rechten Seite einer Regel überein, so wird die Folge durch einen neuen Baum ersetzt. Eine Übereinstimmung gibt es, wenn die Kategorie des Knotens dieselbe ist wie das Element auf der rechten Seite, wobei die Kategorie die linke Seite der Regel ist und dessen Kinder die Nachfolger.

Zieltest:

Der Zieltest überprüft nur den Zustand, der aus einem einzigen Baum mit der Wurzel S besteht.

Es gibt noch weitere schnellere Parsing-Algorithmen die aber hier nicht näher erläutert werden sollen, da sie nicht zum direkten Umfang des Fachs „Künstliche Intelligenz“ gehören.

Artikel	**Knotenliste**	**Folge**	**Regel**
ANFANG	**the wumpus is dead**	**the**	Artikel → the
2	Artikel **wumpus is dead**	**wumpus**	Substantiv → wumpus
3	Artikel Substantiv **is dead**	Artikel Substantiv	NP → Artikel Substantiv
4	NP **is dead**	**is**	Verb → is
5	NP Verb **dead**	**dead**	Adjektiv → dead
6	NP Verb Adjektiv	Verb	VP → Verb
7	NP Verb Adjektiv	VP Adjektiv	VP → VP Adjektiv
8	NP VP	NP VP	S → NP VP
Ziel	S		

Abbildung 22.6: Bottom-Up Parsing für den Satz „the wumpus is dead“

22.4 Erweiterte Grammatiken

Das Problem der Beispielgrammatik ε_0 ist, dass viele Nicht-Sätze erzeugt werden, die zwar zur Grammatik gehören, aber im Grunde dennoch falsch sind. Ein Beispiel für einen solchen Satz wäre „*Me smell a Stench*“. In diesem Fall kommt der Fehler zustande, da die Grammatik nicht erkennt, dass das Wort „`me`“ als Subjekt im Satz kein gültiges NP ist. Die Linguisten verwenden „*I*“ als **Subjektfall** und „*me*“ als **Objektfall**. Dieser Fall zeigt, dass die Grammatik nicht kontextfrei ist. Das Problem kann nur durch die Einführung neuer Kategorien behoben werden – NP_s (Subjektfall), NP_o Objektfall, $Pronomen_S$ und $Pronomen_O$. Die geänderte Grammatik wird als ε_1 bezeichnet. Die Einführung der neuen Regeln haben zur Folge, dass alle NP-Regeln dupliziert werden müssen.

Diese geänderte Grammatik erzeugt allerdings immer noch zu viele Sätze. So ist zum Beispiel des Satz „*I smells a stench*“ generierbar. Bei natürlichen Sprachen ist es allerdings wichtig, dass es eine Übereinstimmung zwischen dem Subjekt und dem Hauptverb des Satzes gibt, es muss also auch hier der Grammatik der richtige Fall entnommen werden. So würde der Beispielsatz „*I smell a stench*“ heißen müssen. Im Englischen ist das Problem nicht so groß, da es nur zwei Formen gibt (dritte Person Singular mit dem „*s*“-Suffix und eine Form für alles andere). Hier gibt es allerdings auch noch ein paar unregelmäßige

Verben, auf die das nicht zutrifft, zum Beispiel das Verb „*be*“ (*i am, you are, he is...*). Solche Sonderfälle sollen der Einfachheit halber nicht weiter betrachtet werden. Alleine mit den bisher neu eingeführten Änderungen in diesem Kapitel entstehen sechs Formen von NP Regeln. Da diese Regeln aber noch lange nicht alle vollständig sind, würde auf diese Art und Weise eine exponentielle Anzahl an Regeln entstehen.

Es muss also eine alternative Möglichkeit gefunden werden die Grammatik zu **erweitern**, statt neue Regeln einzuführen. Diese Erweiterung sieht formal wie folgt aus:

S	→	NP(Subjekt) VP | ...
NP(Fall)	→	Pronomen(Fall) | Name | Substantiv | ...
VP	→	VP NP (Objekt) | ...
PP	→	Präposition NP (Objekt)
Pronomen(Subjekt)	→	I | you | he | she | it | ...
Pronomen(Objekt)	→	me | you | him | her | it | ...

Abbildung 22.7: Erweiterte Grammatik ε_1

Die erweiterten Regeln unterstützen Parameter für nonterminale Kategorien. Die Kategorien NP und Pronomen wurden um den Parameter „*Fall*“ erweitertet. Für die Regel S muss das NP im Subjektfall stehen, in den Regeln für VP und PP muss es im Objektfall stehen. Die Regel NP nimmt eine Variable „*Fall*“ auf, so ist es möglich, dass das NP einen beliebigen Fall haben kann. Sobald das NP als Pronomen umschrieben wird, muss der entsprechende Fall angewendet werden.

Diese Vorgehensweise wird auch als definite Klauselgrammatik (DCG, Definite Clause Grammar) bezeichnet. Jeder Grammatikregel kann als definite Klausel in der Horn-Logik interpretiert werden. So kann die Regel $S \rightarrow NP\ VP$ geschrieben werden als $NP(s_1) \wedge VP(s_2) \Rightarrow S(s_1 + s_2)$, wobei $s_1 + s_2$ die Konkatenation zweier Zeichenketten bedeutet. Falls es sich bei der obigen Zeichenkette s_1 um ein NP handelt und bei der Zeichenkette s_2 um ein VP, ergibt ihre Konkatenation ein S (was genau der Interpretation der CFG Regel entspricht). Durch diesen Formalismus ist es möglich über das Parsing als logische Referenz zu sprechen, so dass über Sprachen und Zeichenketten auf viele unterschiedliche Arten geschlossen werden kann. Das Bottom-Up-Parsing könnte unter

Verwendung der Vorwärtsverkettung und das Top-Down-Parsing unter Verwendung der Rückwärtsverkettung durchgeführt werden. Der Hauptvorteil wurde allerdings bereits bei der Grammatik ε_1 angewendet – das Erweitern der Kategorien um zusätzliche Argumente.
Wendet man diese Erkenntnis auf die Regel $NP(Fall) \rightarrow Pronomen(Fall)$ an, die eine Abkürzung für eine definite Klausel $Pronomen(Fall, s_1) \Rightarrow NP(Fall, s_1)$ ist, so bedeutet das, dass die Zeichenkette s_1 ein Pronomen ist, dessen Fall durch die Variable „*Fall*" vorgegeben wird. s_1 ist dann auch ein NP mit demselben Fall.
Die Kategoriesymbole können dabei mit beliebig vielen Argumenten erweitert werden, die Argumente sind dabei Parameter, die einer Unifizierung unterliegen, wie üblich bei Horn-Klauseln. Der Nachteil dieser bequemen Methode eine Grammatik zu parametrisieren ist, dass die Grammatik nicht mehr zwingend in $O(n^3)$ syntaktisch geparsed werden kann. Sie kann sogar NP-vollständig oder unentscheidbar sein, je nach Änderungen.
Damit die DCG-Regeln vollständig funktionieren muss noch ein Formalismus eingeführt werden um die Terminalsymbole zu spezifizieren.

- Notation X → Y Z ... wird als $Y(s_1) \wedge Z(s_2) \wedge ... \Rightarrow X(s_1 + s_2 + ...)$ übersetzt
- Notation X → Y | Z | ... wird als $Y(s) \vee Z(s) \vee ... \Rightarrow X(s)$ übersetzt

In beiden Regeln kann jedes Nonterminal-Symbol durch ein oder mehrere Argumente erweitert werden, jedes Argument kann eine Variable, eine Konstante oder eine Funktion von Argumenten sein. Bei der Übersetzung gehen die Argumente der Zeichenkette voraus. So wird beispielsweise das NP(Fall) als „*NP(Fall,s1)*" übersetzt. Die Notation X → Wort wird in X([Wort]) übersetzt.

22.4.1 Verbsubkategorisierung

ε_1 ist zwar schon eine Verbesserung gegenüber ε_0, kann aber immer noch viele unsinnige Sätze erzeugen. Sätze wie „*Give me the gold*" oder „*Go to 1,2*" sind korrekt, allerdings können so auch Sätze wie „*Go me the gold*" oder „*Give to 1,2*" erstellt werden, die nicht

zur Sprache gehören sollen. Die Sprache ε_2 legt explizit fest, welche Phrasen welche Verben folgen dürfen. Es wird eine Liste für das Verb erstellt, die Subkategorisierungsliste. Dazu ist es nötig die Kategorie Verb in Subkategorien zu zerlegen. Eine Möglichkeit wäre es nach Verben mit und ohne Objekt oder Ähnliches zu trennen.

Um diese Idee umzusetzen, teilt man jedem Verb eine Subkategorisierungsliste zu, die alle Komplemente des Verbs auflistet. Ein Komplement ist dabei eine obligatorische Phrase, die dem Verb innerhalb einer Verbphrase folgt. In dem Beispielsatz „*Give the gold to me*" sind das NP „*the gold*" und das PP „*to me*" Komplemente von „*give*". Als formale Schreibweise wird folgende Notation eingeführt: „*Verb([NP,PP]) → give | hand | ...*".

Verb	Subkategorien	Beispiel Verbphrase
give	[NP,PP]	give the gold in 3 3 to me
	[NP,NP]	give me the gold
smell	[NP]	smell a wumpus
	[Adjektiv]	smell awful
	[PP]	smell like a wumpus
is	[Adjektiv]	is smelly
	[PP]	is in 2,2
	[NP]	is a pit
died	[]	died
believe	[S]	believe the wumpus is dead

Abbildung 22.8: Subkategorien verschiedener Verben

Um eine Verb-Subkategorisierung in eine Grammatik einzubinden sind drei Schritte nötig

1. Kategorie erweitern, so dass sie ein Subkategorisierungsargument entgegen nimmt
   Das Argument enthält eine Liste mit Komplementen die für ein vollständiges VP erforderlich sind „*VP(subcat)*". Das Verb „*give*" kann zu einem vollständigen VP gemacht werden, indem man [NP,PP] hinzufügt. „*give the gold*" kann vervollständigt werden, indem man ein [PP] hinzufügt. „*give the gold to me*" ist bereits ein vollständiges VP, in diesem Fall ist die Subkategorisierungsliste die leere

Liste [ ].

VP(subcat)	→	Verb(subcat)
	|	VP(subcat + [NP]) NP (Objekt)
	|	VP(subcat + [Adjektiv]) Adjektiv
	|	VP(subcat + [PP]) PP

Abbildung 22.9: Erweiterung der VP Kategorie

2. Die letzte Zeile der Grammatik bedeutet, dass ein VP mit einer bekannten Subkategorisierungsliste „*subcat*“ durch ein eingebettetes VP gefolgt von einem PP gebildet werden kann. Man muss nur beachten, dass das eingebettete VP eine Subkategorisierungsliste hat, die mit den Elementen der Liste „*subcat*“ beginnt und mit dem Symbol VP aufhört. Die erste Zeile drückt aus, dass ein VP mit der Subkategorisierungsliste „*subcat*“ durch ein Verb mit derselben Subkategorisierungsliste gebildet werden kann. Ein Beispiel hierfür wäre das Verb „grab“, dass nach Anwendung der Regel und ersetzen mit dem NP „*the gold*“ „*grab the gold*“ ergibt.
3. Regel für S anpassen

   Die Startregel S muss ein VP fordern, dass alle Komplemente von S enthält, also in diesem Fall die Subkategorisierungsliste. $S \rightarrow NP(Subjekt)\ VP([\,])$ heißt, dass ein Satz aus einem NP im Subjektfall und einem VP mit leerer Subkategorisierungsliste bestehen kann.

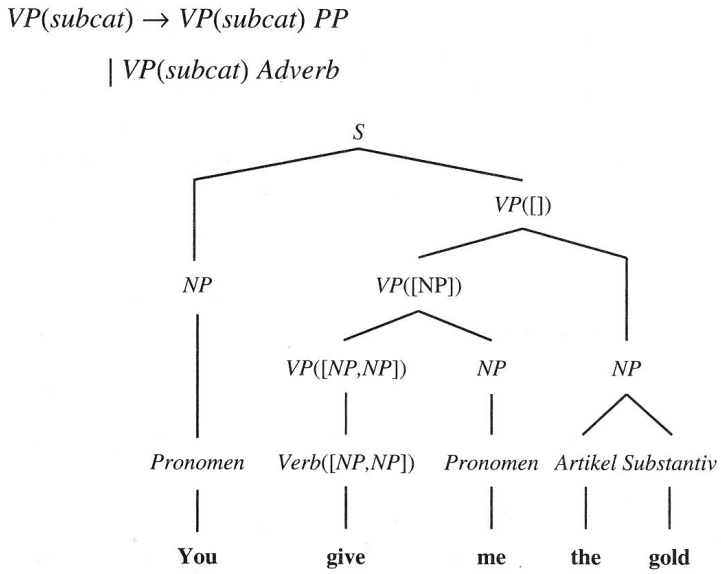

Abbildung 22.10: Parse-Baum für „You give me the gold“

4. Erweiterungen der Verbphrasen und Komplemente um **Beifügungen**
   Beifügungen können quasi in jeder Verbphrase erscheinen. Ein gutes Beispiel für diese Erweiterung sind Phrasen, die Auskunft über einen Zeitpunkt oder einen Ort machen. Als Adverb kann dies „*I smell a wumpus now*“ oder als PP „*on Tuesday*“ sein.

VP(subcat)	→	VP(subcat) PP
	|	VP(subcat) Adverb

Abbildung 22.11: Erweiterung um Beifügungen

22.5 Semantische Interpretation

Um die Bedeutung einer Zeichenkette extrahieren zu können wird die First-Order-Logic verwendet. Als Beispiel soll eine Rasterposition genommen werden die aus zwei Ziffern besteht und als Angabe einer Position dient. Die Regel „*NP → Ziffer Ziffer*“ wird so erweitert, dass jedem Bestandteil ein Argument für die Semantik hinzugefügt wird - „*NP([x,y]) → Ziffer(x) Ziffer(y)*“. Dieser neue Ausdruck bedeutet, dass eine Ziffer mit der Semantik x gefolgt von einer Ziffer mit der Semantik y ein NP mit der Semantik [x,y]

ergibt, also die Rasterposition. Als einführendes Beispiel soll die Grammatik für arithmetische Ausdrücke näher untersucht werden.

$\text{Exp}(x) \rightarrow \text{Exp}(x_1)\ \text{Operator}(op)\ \text{Exp}(x_2)\ \{x = \text{Apply}(op, x_1, x_2)\}$
$\text{Exp}(x) \rightarrow (\ \text{Exp}(x)\ )$
$\text{Exp}(x) \rightarrow \text{Number}(x)$
$\text{Number}(x) \rightarrow \text{Digit}(x)$
$\text{Number}(x) \rightarrow \text{Number}(x_1)\ \text{Digit}(x_2)\ \{x_2 = 10 \times x_1 + x_2\}$
$\text{Digit}(x) \rightarrow x\ \{0 \leq x \leq 9\}$
$\text{Operator}(x) \rightarrow x\ \{x\ \varepsilon\ \{+,-,*,\div,\times\}\}$

Abbildung 22.12: Erweiterung um Beifügungen

Jeder Variable x_i stellt die entsprechende Semantik dar. Daraus lässt sich zum Beispiel folgender Semantik Parse-Baum erzeugen.

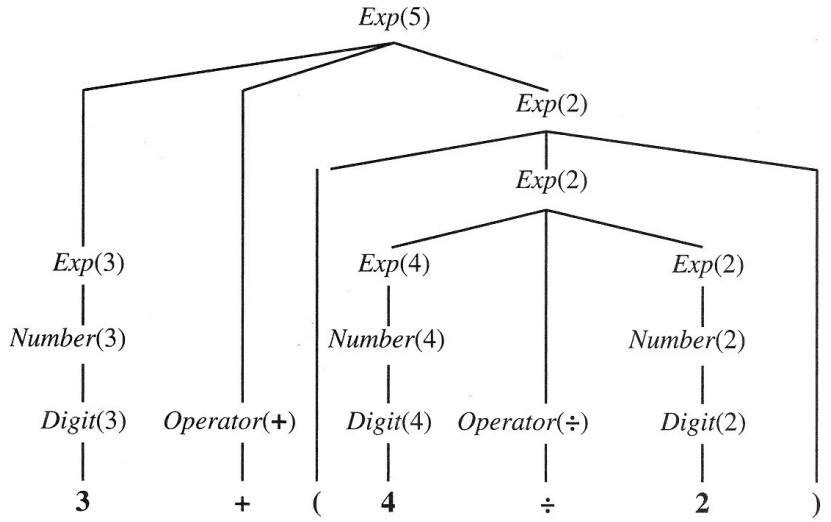

Abbildung 22.13: Parse-Baum für die Zeichenkette: „3+(4÷2)“

22.5.1 Die Semantik eines Fragments von Englisch

Zunächst soll festgelegt werden welche semantische Repräsentation welchen Phrasen

zugeordnet wird. Dazu wird der Beispielsatz „*John loves Mary*“ betrachtet. Das NP John sollte als semantische Interpretation den logischen Term John liefern und der Satz als Ganzes sollte „*Loves(John,Mary)*“ liefern. Das Problem liegt in der semantischen Interpretation von „*loves Mary*“. Dieses Fragment ist weder ein logischer Term (ein Argument fehlt) noch ein vollständiger Satz. Intuitiv könnte man sagen, dass „*loves Mary*“ auf eine bestimmte Person bezogen ist. Es handelt sich also um ein **Prädikat**, dass bei Kombination mit einem Term, der eine Person darstellt, einen vollständigen logischen Satz erzeugt. Daraus ergibt sich „*λ x Loves(x,Mary)*“[2] woraus die Regel abgeleitet werden kann, dass ein NP mit Semantik „obj“ gefolgt von einem VP mit Semantik „rel“ einen Satz mit der Semantik der Anwendung von „rel“ auf „obj“ darstellt ($S(rel(obj)) \rightarrow NP(obj)\ VP(rel)$). Die semantische Interpretation von „John loves Mary“ lautet also: „*(λ x Loves(x,Mary))(John)*“, was äquivalent zu „*Loves(John, Mary)*“ ist. Da VPs als Prädikate dargestellt werden, sollte Verben ebenfalls als solche dargestellt werden. Das Verb „*loves*“ wird demnach als „*λx λy Loves(x,y)*“ dargestellt.

S(rel(obj))	→ NP(obj)VP(rel)
VP(rel(obj))	→ Verb(rel) NP(obj)
NP(obj)	→ Name(obj)
Name(John)	→ John
Name(Mary)	→ Mary
Verb(λx λy Loves(x,y))	→ loves

Abbildung 22.14: Ausschnitt der Grammatik mit Semantik

2 λ-Notation: Hilfe um neue Funktionssymbole dynamisch zu erzeugen. Beispiel: Funktion die ihr Argument quadriert: λx x × x

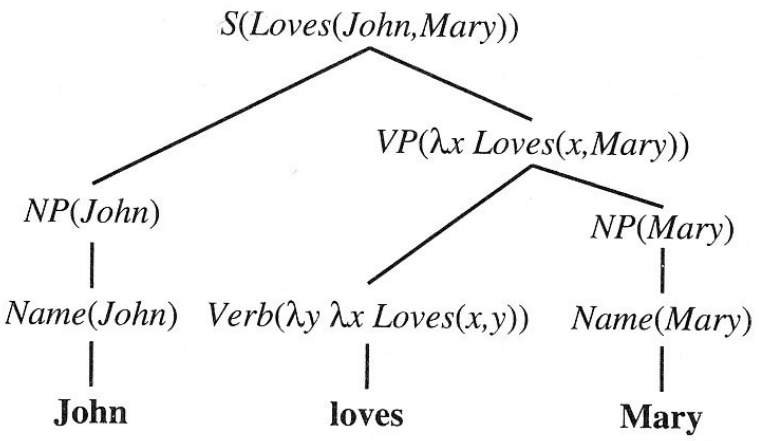

Abbildung 22.15: Parse-Baum für den String „John loves Mary“

22.5.2 Zeit und Zeitform

Ein weiteres Problem, welches die englische Grammatik mit sich bringt, ist die Benutzung von unterschiedlichen Zeiten („*past*“, „*present*“ und „*future*“). Um die Semantik diesbezüglich zu erweitern wird das Ereigniskalkül verwendet. Aus den Beispielen $e \in Loves(John, Mary) \wedge Während(Jetzt, e)$ und $e \in Loves(John, Mary) \wedge Nach(Jetzt, e)$ ergibt sich dann:

$Verb(\lambda x \lambda y e \in Loves(John, Mary) \wedge Während(Jetzt, e)) \rightarrow loves$
$Verb(\lambda x \lambda y e \in Loves(John, Mary) \wedge Nach(Jetzt, e)) \rightarrow loved$

Bei dem Zeitform Problem kann die Grammatik also ohne viel weiteren Aufwand verändert werden.

22.5.3 Quantifizierung

Untersucht man den Satz „Jeder Agent riecht ein Wumpus“, so könnte man diesen in der FOL darstellen als:

Jeder Agent $NP(\forall\ a\ \ a \in\ Agenten \Rightarrow\ P)$
riecht ein Wumpus $VP(\exists\ w\ \ w \in\ Wumpi \wedge \exists\ e(e \in\ Riecht(a,w) \wedge Während(Jetzt,e))$

Die Variable a müssten man als Argument der Relation „*riecht*“ erhalten. Die Semantik wird gebildet, indem die Semantik des VP an den richtigen Argumentplatz des NP eingefügt wird. Allerdings wird die Variable a gleichzeitig aus dem NP an den richtigen Argumentplatz der Semantik VP eingefügt. Das Problem liegt darin, dass sich die semantische Struktur wesentlich von der syntaktischen Struktur unterscheidet.

Lösung für dieses Problem liefert eine **Zwischenform (quasi logische Form = QLF)**. Diese ist von der Struktur her, der Syntax des Satzes ähnlich und kann mit Hilfe von kompositioneller Mittel leicht erzeugt werden. Die QLF enthält alle nötigen Informationen um in die FOL übersetzt zu werden und wird um den λ-Ausdruck und um quantifizierte Terme erweitert.

Der quantifizierte Term $[\forall\ a\ a \in\ Agenten]$ hat die Bedeutung „*jeder Agent*“. Er sieht aus wie ein logischer Satz wird aber wie ein logische Term verwendet. Der Satz „*Jeder Agent riecht ein Wumpus*“ würde in der QLF als

$$\exists\ e\left(e \in\ Riecht\left([\forall\ a\ a \in\ Agent],[\exists\ w\ w \in\ Wumpi]\right) \wedge Während(Jetzt,e)\right)$$

dargestellt werden. Ein Großteil der bisher aufgestellten Regeln bleiben erhalten, allerdings müssen ein paar Regeln noch angepasst werden, zum Beispiel die Regel für „*ein*“ - $Artikel(\exists) \rightarrow\ ein$ und die Kombination eines Artikels mit einem Substantiv - $NP([q\,x\,sem(x)]) \rightarrow\ Artikel(q)Substantiv(sem)$. Die Semantik von NP ist also ein quantifizierter Term, wobei der Quantifizierer durch den Artikel bestimmt wird.

Um die QLF in die FOL umzuwandeln müssen die quantifizierten Terme in reale Terme umgewandelt werden. Für jeden quantifizierten Term „*[q x P(x)]*“ wird der quantifizierte Term durch „*x*“ und die QLF durch „*q x P(x)* ⇒ *QLF*“ ersetzt, wenn „*q*“ gleich ∀ ist, und ∧, wenn „*q*“ gleich ∃. Der Satz „Every dog has a day“ soll dieses noch einmal verbildlichen. $\exists\ e\left(e \in\ Has([\forall\ d\ \ d \in\ Dogs],[\exists\ a\ \ a \in\ Days],Jetzt)\right)$ ist der Ausdruck um diesen Satz in der QLF darzustellen. Es wird allerdings nicht angegeben welcher der quantifizieren Terme zuerst ausgewählt wird, so dass es zwei Lösungsmöglichkeiten gibt:

$\forall\ d\ \ d \in\ Dogs \Rightarrow\ \exists\ a\ \ a \in\ Days \wedge \exists\ e\ e \in\ Has(d,a,Jetzt)$
$\exists\ a\ \ a \in\ Days \wedge \forall\ d\ \ d \in\ Dogs \Rightarrow\ \exists\ e\ \ e \in\ Has(d,a,Jetzt)$

Die erste Möglichkeit besagt, dass jeder Hund einen speziellen Tag hat. Bei der zweiten, dass es einen speziellen Tag gibt, den alle Hunde gemeinsam haben. Um zu erfahren welche Lösung die richtige ist, muss die Mehrdeutigkeit aufgelöst werden.

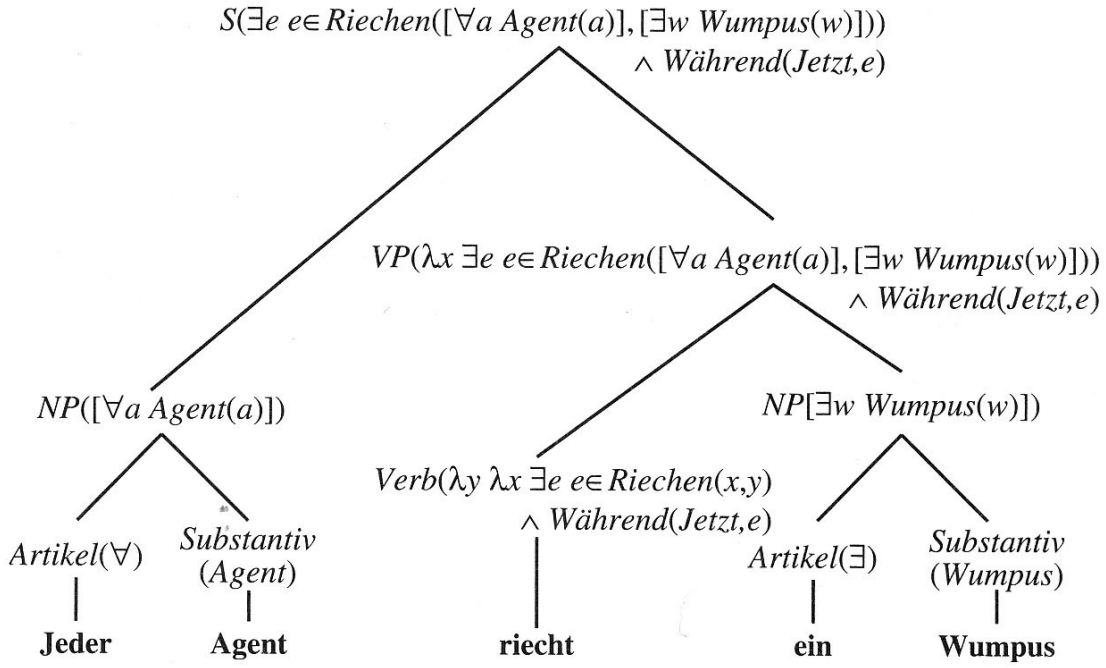

Abbildung 22.16: Parse-Baum für den Satz „Jeder Agent riecht ein Wumpus“ mit Semantik

Der Vorteil der QLF ist, dass alle Möglichkeiten in kurzer Form dargestellt werden, allerdings ist es so nicht möglich aus mehreren Möglichkeiten die richtige auszuwählen.

$$S(rel(obj)) \rightarrow NP(obj)\,VP(rel)$$
$$S(conj(sem_1, sem_2)) \rightarrow S(sem_1)\,Konjunktion(conj)\,S(sem_2)$$

$$NP(sem) \rightarrow Pronomen(sem)$$
$$NP(sem) \rightarrow Name(sem)$$
$$NP([q\,x\,sem(x)]) \rightarrow Artikel(q)\,Substantiv(sem)$$
$$NP([q\,x\,obj \wedge rel(x)]) \rightarrow NP([q\,x\,obj])\,PP(rel)$$
$$NP([q\,x\,obj \wedge rel(x)]) \rightarrow NP([q\,x\,obj])\,RelClause(rel)$$
$$NP([sem_1, sem_2]) \rightarrow Ziffer(sem_1)\,Ziffer(sem_2)$$

$$VP(sem) \rightarrow Verb(sem)$$
$$VP(rel(obj)) \rightarrow VP(rel)\,NP(obj)$$
$$VP(sem_1(sem_2)) \rightarrow VP(sem_1)\,Adjektiv(sem_2)$$
$$VP(sem_1(sem_2)) \rightarrow VP(sem_1)\,PP(sem_2)$$

$$RelClause(sem) \rightarrow that\,VP(sem)$$

$$PP(\lambda\,x\,rel(x, obj)) \rightarrow Präposition(rel)\,NP(obj)$$

Abbildung 22.17: Fragment von Englisch in QLF

22.5.4 Pragmatische Interpretation

Bisher ist es möglich, dass der Agent aufeinander folgende Wörter so ableiten kann, dass er mögliche semantische Interpretationen erhält. Zusätzlich dazu soll die Pragmatik die aktuelle Situation mit einbeziehen, dazu werden **Indexicals** verwendet. Dies sind Phrasen die sich direkt auf die aktuelle Situation beziehen. Wenn man den Satz „*Ich bin heute leider nicht in Gummersbach*“ betrachtet, so stellt man fest, dass die Interpretation davon abhängt, wer diesen Satz gesprochen hat. Der Hörer der den Sprechakt wahrnimmt muss also wissen, wer der Sprecher ist. Die Indexicals repräsentieren Konstanten (z.B. den Sprecher) und werden durch Fluents dargestellt, um sie situationsabhängig zu machen. Um das Beispiel Indexical aufzulösen müsste der Agent zum Beispiel wissen, dass der Fluent T((Sprecher = AgentB),Jetzt) ist.

Bei Befehlen kommt es häufig zu impliziten Sätzen, wie zum Beispiel „*Gehe nach 2,2*“, die sich implizit auf den Hörer beziehen. Bisher war es nur möglich, dass die Grammatik deklarative Sätze abdeckt, und soll nun so erweitert werden um Befehle zu zulassen. In diesem Fall ist das Subjekt implizit der Hörer, so dass man es auf ein VP abbilden kann. Da Befehle von Aussagen unterschieden werden müssen, muss dies auf der obersten Regel (für den Satz) geschehen.

$S(Aussage(Sprecher, rel(obj))) \rightarrow NP(obj)VP(rel)$
$S(Befehl(Sprecher, rel(Hörer))) \rightarrow VP(rel)$

Daraus ergibt sich in der QLF die Form $Befehl(\exists\, e\, e \in Gehe(Hörer,[2,2]))$ für obigen Beispielsatz.

22.6 Mehrdeutigkeit und Auflösung der Mehrdeutigkeit

Es gibt Äußerungen, da ist die Mehrdeutigkeit klar zu erkennen, z.B: „*Junge traf Nachbarn mit Gewehr*“ oder „*Ehefrau fuhr Schauspieler betrunken nach Hause*“. Wenn man die Sprache hört oder liest, so scheint es vom menschlichen Empfinden her immer so zu sein, dass die Aussagen eindeutig sind. Als man in den 60er Jahren den Computer einsetze um die Sprache zu analysieren, fand man heraus, das genau das Gegenteil der Fall ist. Fast jeder Satz der Muttersprache birgt mehrere alternative Interpretationsmöglichkeiten. Der Mensch hat von Natur aus die Fähigkeit einen Satz richtig zu interpretieren, ein Agent hat dieses nicht, was zu großen Problemen führen kann. Bei Sprachen mit großen Grammatiken und mit großem Lexikon ergeben sich tausende von Interpretationsmöglichkeiten für einen prinzipiell eindeutigen Satz. Dieses soll der Satz „*Der Schläger schlägt den Ball“* zeigen. Die eigentliche Interpretation ist ein Baseballspieler, der einen Ball schlägt. Ist der vorhergehende Satz allerdings „*Der verrückte Boxer schlug haltlos auf die Menschen im Ball ein*“ erhält man eine vollkommen andere Interpretation.

22.6.1 Arten von Mehrdeutigkeiten

Falls ein Wort mehrere Bedeutungen hat, spricht man von der **lexikalischen Mehrdeutigkeit**. Diese tritt sehr oft auf, da es Wörter gibt die Adjektiv oder Substantiv sind oder mehrere Bedeutungen haben. Ein Beispiel hierfür wäre das Wort „Bock“, dass sowohl ein Turngerät, ein Skatbegriff, sowie ein Tier sein kann. Die **syntaktische** oder auch **strukturelle Mehrdeutigkeit** entsteht durch unterschiedliche Parsing Möglichkeiten. Der Satz „*Ich rieche ein Wumpus in 2,3*“ hat zwei mögliche Bedeutungen.

1. „in 2,3“ modifiziert das Substantiv → Das Wumpus ist in Feld 2,3
2. „in 2,3“ modifiziert das Verb → in Feld 2,3 stinkt es

Die syntaktische Mehrdeutigkeit würde in diesem Fall zu einer **semantischen Mehrdeutigkeit** führen. Diese kann allerdings auch in Sätzen ohne lexikalische oder syntaktische Mehrdeutigkeit auftreten. Zum Beispiel hat der Satz „*Ein Junggeselle ist ein Mann, dem zum Glück noch die Frau fehlt.*“ zwei Interpretationsmöglichkeiten

1. Er wäre glücklich, wenn er eine Frau hätte
2. Er kann sich freuen, dass er noch keine Frau hat

Falls eine Äußerung syntaktisch und semantisch eindeutig ist kann dennoch die **pragmatische Mehrdeutigkeit** auftreten, zum Beispiel in dem Satz „*Das ist aber kalt hier*“.

1. Feststellung: Sprecher sagt etwas über die Temperatur in einem Raum aus
2. Aufforderung: Sprecher will das jemand die Heizung anmacht.
3. Klage: Zustand wird negativ empfunden, kann aber nicht geändert werden

Die **phonologische Mehrdeutigkeit** kann nur in der gesprochenen Sprache auftreten, wenn also eine Übertragung mittels Schall stattfindet. Sie tritt in Erscheinung, wenn zwei Worte gleich klingen, wie zum Beispiel „*Bären*“ / „*Beeren*“.

22.6.2 Sprachfiguren

Zudem gibt es noch Mehrdeutigkeiten zwischen **wörtlichen** und **figürlichen Bedeutungen**. Sprachfiguren werden in der Poesie bewusst eingesetzt, allerdings verwenden wir sie auch im alltäglichen Gebrauch sehr oft, werden dann allerdings unbewusst interpretiert (z.B.: „*Die Sonne geht auf...*“).

Ein **Metonym** ist eine Sprachfigur, die ein Objekt verwendet anstelle eines anderen Objektes. So sollte der Satz „*Ford kündigt ein neues Modell an*“ nicht so interpretiert werden, dass das Unternehmen selber sprechen könnte. Man geht stattdessen davon aus, das ein Pressesprecher die Mitteilung in die Welt setzt. Auch hier stellt man fest, dass der Mensch dieses vollkommen unbewusst interpretiert. Eine Grammatik ist dazu allerdings nicht in der Lage, man braucht also einen Mechanismus der eine neue Ebene der Mehrdeutigkeit erkennt. Dazu werden zwei Objekte für die semantische Interpretation jeder Phrase im Satz zur Verfügung gestellt, eine für das wörtliche Objekt (Ford) und eine für den metonymischen Verweis auf den Pressesprecher. Diese Objekte müssen dann noch in eine Relation gestellt werden, so dass der ursprüngliche Ausdruck $\exists x,e\ x = Ford \wedge e \in Ank\ddot{u}ndigung(x) \wedge Nach(Jetzt,e)$ mit Hilfe der neuen Ebene in den Ausdruck $\exists m,x,e\ x = Ford \wedge e \in Ank\ddot{u}ndigung(m) \wedge Nach(Jetzt,e) \wedge Metonym(m,x)$ umgewandelt wird. Dieser drückt aus, dass sich „*m*“ und „*x*“ in metonymischer Relation zueinander befinden. Es gibt verschiedenste Arten von Metonymie, einige sollen kurz erläutert werden.

- keine Metonymie (m = x) , trivial
- Unternehmen, Pressesprecher
  $\forall m,x\ x \in Unternehmen \wedge Pressesprecher(m,x) \Rightarrow Metonym(m,x)$
- Autoren von Büchern „*Ich lese Shakespeare*“
- Produzent eines Produktes „*Ich fahre einen Ford*“
- auf bestimmte Situation interpretiert („*Das Schinkensandwich an Tisch 4 will noch ein Bier*“) - soll heißen, dass der Mann an Tisch 4, der ein Schinkensandwich isst noch ein Bier bestellt hat

Anhand dieser Beispiele zeigt sich, dass der Mensch die Metonyme ohne weiteres in Relation mit den eigentlichen Begriffen stellen kann. Ein Agent könnte dieses nicht zwingend über logisches Schließen herausfinden, sondern müsste stattdessen probabilistisches oder nicht monotones Schließen anwenden.

Metaphern sind Sprachfiguren, in denen eine Phrase mit einer bestimmten wörtlichen Bedeutung dazu verwendet wird, um mit Hilfe einer Analogie, eine andere Bedeutung auszudrücken. Auch hier ist es wieder wichtig zu wissen, dass sie nicht eine reine Erfindung von Dichtern sind, sondern sehr häufig im Sprachgebrauch verwendet wird, wenn auch unbewusst. Ein gutes Beispiel bietet das so genannte „*hoch*"-Konzept. Mit der Metapher „*die Preise sind hoch*" wird ausgedrückt, dass die Preise gestiegen sind. Es gibt zwei Möglichkeiten eine solche Metapher zu verarbeiten.

1. Das gesamte Wissen über eine Metapher wird mit in das Lexikon aufgenommen, sprich die neuen Bedeutungen der Wörter „*steigen*" oder „*klettern*" hinzufügen. Im Normalfall ist diese Vorgehensweise ausreichend, neue Versionen können aber nicht abgedeckt werden, da es zu Missverständnissen kommen kann. Beispiel „*Sturzflug*" oder „*über den Kopf wachsen*".
2. Das explizite Wissen allgemeiner Metaphern wird aufgenommen, so werden neue Verwendungen interpretiert sobald diese auftreten, was in der ersten Lösung nicht der Fall ist. Kennt das System zum Beispiel die Metapher „*steigen*" als einen logischen Ausdruck der sich auf einen Punkt einer vertikalen Skala bezieht, so kann man auf einer Qualitätsskala den neuen Punkt oberhalb des alten interpretieren, um so eine Steigerung anzuzeigen. Der Ausdruck „Die Verkäufe sind gestiegen" würde wörtliche interpretiert werden als „*Höhe(Verkäufe,Hoch)*" und wird umgewandelt zu „*Menge(Verkäufe, Viel)*", wenn man die Metapher „*steigen*" auflöst.

22.6.3 Auflösen der Mehrdeutigkeit

Das Auflösen der Mehrdeutigkeit ist ein Diagnoseproblem, welches beim Hörer, der die

Nachricht empfängt, auftritt. Es ist seine Aufgabe aus den Wörtern, dem Wissen über die Situation und seinem Gesamtwissen die wahrscheinlichste Intention des Hörers zu erkennen. Da die Regeln für die syntaktische und semantische Interpretation keine eindeutige Interpretation liefern können, muss der Prozess gesplittet werden. Die Syntax- und Semantikanalyse erstellt eine Menge von möglichen Interpretationen, der Prozess „Auflösung der Mehrdeutigkeit" wählt dann die Beste aus. Allerdings muss man beachten, dass man die Intention des Sprechers betrachtet, nicht die tatsächliche Aussage. Wenn ein Politiker sagt „Ich bin kein Halsabschneider", so könnte man der Aussage, dass er kein Krimineller ist eine Wahrscheinlichkeit von 50% zuordnen und der Aussage, dass er nicht mit einem Messer herum rennt und andere Leute bedroht eine Wahrscheinlichkeit von 99%. Dennoch wird der ersten Interpretationsmöglichkeit eine höhere Wahrscheinlichkeit zugeordnet, da sie eher der Intention des Politikers entspricht.

Es ist also eine Prioritätsheuristik nötig, um zu entscheiden, welche Interpretation die richtige ist. Am Beispiel des Satzes „*Ich rieche ein Wumpus in 2,3*" soll dieses nochmal gezeigt werden. Eine mögliche Heuristik könnte hingehen und im Parse-Baum die direkt vorgehende Phrase finden, in diesem Fall das NP „*Wumpus*". Man würde dann davon ausgehen, dass das Wumpus im Feld 2,3 ist. Diese Methode ist allerdings sehr unsicher, es wäre sinnvoller, wenn man wüsste wo sich der Sprecher befindet. Steht der Agent in Feld 2,3, so ist die Substantivinterpretation nicht korrekt, da er sonst auf einem Feld mit dem Wumpus wäre.

Die Auflösung der Mehrdeutigkeit ist ein sehr kompliziertes Thema in der KI und kann nur erfolgreich angewandt werden wenn alle Techniken zur Wissensrepräsentation kombiniert werden und auf das unsichere Schließen angewendet werden. Das Wissen lässt sich dabei in vier Modell zerlegen.

1. **Weltmodell**, gibt an wie hoch die Wahrscheinlichkeit ist, dass eine Behauptung zutrifft
2. das **mentale Modell** ermittelt die Höhe der Wahrscheinlichkeit, ob der Sprecher die Intention hat ein Ereignis dem Hörer mitzuteilen, sobald dieses auftritt
3. das **Sprachmodell** ermittelt die Wahrscheinlichkeit, dass eine bestimmte Wortfolge gewählt wird, wenn der Sprecher die Intention hat eine bestimmte Tatsache

mitzuteilen. Da die CFG und DCG Modelle nur eine boolesche Aussagekraft haben wird in Kapitel 23 ein probabilistisches Modell vorgestellt.

4. das **akustische Modell** gibt die Wahrscheinlichkeit an, dass eine bestimmte Klangfolge erzeugt wird, wenn der Sprecher eine bestimmte Wortfolge gewählt hat (siehe Kapitel 15.6)

22.7 Gesprächsverständnis

Ein **Gespräch** ist eine Sprachfolge, die normalerweise aus mehreren Sätzen besteht, Beispiele hierfür sind Lehrbücher, Romane, Nachrichten, Unterhaltungen, etc. Bisher wurden die Sätze meistens für sich allein betrachtet. Jetzt soll allerdings der Satz innerhalb seiner natürlichen Umgebung – im Gespräch - betrachtet werden. Dabei entstehen zwei sehr wichtige Probleme, die **Verweisauflösung** und die **Kohärenz**.

22.7.1 Verweisauflösung

Die Verweisauflösung befasst sich mit der Interpretation eines Pronomens oder einer Substantivphrase, die auf ein Objekt in der Welt verweisen. Das Ergebnis der Verweisauflösung ist abhängig von dem Wissen über die Welt und vom Wissen über den vorhergehenden Gesprächsverlauf.

Bei dem Beispielsatz „*Felix winkte dem Ober. Er bestelle ein Schinkensandwich.*“ geht die Verweisauflösung zunächst hin und extrahiert alle Objekte, die sich auf das „*er*“ beziehen können und gibt dann eine Liste mit den Objekten Felix und Ober zurück. Mit dem Wissen über die Welt, wird dann geschlossen, dass sich das „*er*“ auf „*Felix*“ bezieht, da Felix in der Kundenrolle ist, und deshalb seine Bestellung an den Ober weiterreicht. Es gibt allerdings auch Sätze wo dies nicht so einfach ist, da die Verweisauflösung nicht offensichtliche Möglichkeiten neu generieren muss, obwohl diese nicht explizit auftauchen. „*Nachdem Bodo um die Hand von Anna angehalten hatte, suchte er einen Standesbeamten und ließ sich trauen. Die Flitterwochen verbrachten sie aus Hawaii*“ ist

ein solcher Satz. Der erste Versuch würde Bodo, Anna und den Standesbeamten zurück liefern, was aber alleine von der Anzahl der Personen nicht stimmen kann. Nun muss das Weltwissen hinzugezogen werden, welches schließt, dass man die Flitterwochen immer zu zweit und immer ohne den Standesbeamten verbringt. Das Wort „*sie*“ bezieht sich also auf das getraute Paar Bodo und Anna.

Die richtige Auswahl des Verweises ist ebenfalls eine Auflösung von Mehrdeutigkeit, in diesem Fall müssen eine große Menge von syntaktischer, semantischer und pragmatischer Informationen kombiniert werden. Daraus ergibt sich ein Suchraum, der nur durch Nebenbedingungen eingegrenzt werden kann. Es gibt mehrere Nebenbedingungen, die man miteinander kombinieren kann:

- Übereinstimmung mit dem Vorgänger

  In diesem Fall müssen Pronomen in Geschlecht und Anzahl übereinstimmen. So kann sich „*er*“ nur auf Bodo beziehen, nicht aber auf Anna. „s*ie*“ kann sich hingegen auf eine Gruppe von Personen beziehen oder auf eine einzige Person.
- Regel der Reflexivität

  Die Pronomen müssen den entsprechenden syntaktischen Regeln gehorchen. „*Er sah ihn im Spiegel*“ heißt, dass sich die beiden Pronomen auf unterschiedliche Personen beziehen müssen, während „*Er sah sich selbst im Spiegel*“ dazu führt, dass sich beide Pronomen auf eine Person beziehen.
- Semantische Konsistenz

  Diese Regel betrachtet vor allem die semantische Korrektheit eines Satzes. Bei dem Satz „*Er aß es*“ muss sich das „*er*“ auf etwas beziehen was man essen kann, und „*es*“ auf etwas das gegessen werden kann (könnte also kein Auto sein).
- Prioritäten

  Falls zwei benachbarte Sätze eine ähnliche Struktur haben, so sollte die Verweisauflösung sich an dieser Struktur orientieren. „*Anna flog nach San Francisco von New York aus. John flog von Boston aus hin.*“ - in diesem Fall würde „*hin*“ als San Francisco gedeutet werden, da es dieselbe syntaktische Rolle spielt. Bei dem Satz „*Anna gab Susi die Hausaufgaben, dann ging sie weg*“ gibt es keine

ähnliche Struktur. Dies führt dazu, dass das Subjekt bevorzugt wird, da die Priorität eines Subjekts größer als die eines Objekts ist.

- Einheit die zuerst genannt wurde

  Der Satz „*Eva ließ die Tasse auf die Platte fallen. Sie zerbrach.*“ ist problematisch, da sich das „*sie*“, sowohl auf die Tasse als auch auf die Platte beziehen kann. Hier ist es einfach nicht möglich zu bestimmen, welches die richtige Lösung ist, aber es wäre eine Möglichkeit in solchen Fällen das zuerst genannte Objekt vorzuziehen.

- Mittelpunkt

  „*Eva war stolz auf die blaue Tasse, sie war ein Geschenk von einem Freund. Als sie eines Tages den Tisch deckte, passierte es: Eva ließ die Tasse auf die Platte fallen. Sie zerbrach.*“ Erweitert man den Kontext des Tassenbeispiels, so wird die Entscheidung wesentlich leichter fallen. Die Tasse steht jetzt eindeutig im Gesamtkontext im Mittelpunkt, so dass sich das „*sie*“nur auf die Tasse beziehen kann.

Auch bei der Thematik Verweisauflösung zeigt sich wieder sehr deutlich, dass der Mensch die Verweise fast immer automatisch bestimmen kann, dieses geschieht wieder unbewusst und führt fast nie zu Missverständnissen. Nur unklare Sätze wie „*Eva ließ die Tasse auf die Platte fallen. Sie zerbrach.*“, wo auch der Kontext fehlt sind selbst für den Menschen nicht eindeutig lösbar, er müsste auch hier nachfragen, was zerbrochen ist oder raten.
Der erste Algorithmus zur Verweisauflösung („*Hobbs*“) wurde 1978 entwickelt und konnte Verweise aus drei verschiedenen Textgenres mit einer Genauigkeit von 92% richtig bestimmen. Einzige Voraussetzung ist ein funktionierender Parser. Der Algorithmus arbeitet wie eine Suche, er sucht Sätze rückwärts, beginnend beim aktuellen Satz, so dass neue in Frage kommende Sätze zuerst betrachtet werden. Innerhalb eines Satzes wird dann mit der Breitensuche von links nach rechts gesucht, so werden die Subjekte vor den Objekten erkannt. Der Algorithmus wählt dann den ersten Kandidaten, der den vorgegebenen skizzierten Bedingungen entspricht, aus.

22.7.2 Struktur kohärenter Gespräche

Hier soll erläutert werden, wie es möglich ist die Sätze innerhalb ihrer Umgebung zu betrachten um zusammenhängende Strukturen zu erkennen. Wenn man ein Buch zehn mal auf zufälligen Seiten öffnet und den ersten Satz heraus schreibt, so ist das Ergebnis eigentlich immer inkohärent, sprich ohne jeglichen Zusammenhang. Dies ist allerdings nur bei Sätze in natürlichen Sprachen der Fall, in der Logik wäre dieses zum Beispiel anders zu bewerten. In der Logik werden die Sätze mittels „*tell*“ zur Wissensbasis hinzugefügt und können durch Konjunktionen miteinander in Verbindung gebracht werden, während es im natürlichen Sprachgebrauch auf die Reihenfolge der Sätze ankommt. Ein ganz simples Beispiel soll dieses Problem verdeutlichen: „*Gehe zwei Häuser weiter und dann nach rechts.*“, sobald man hier die Sätze vertauscht wird der Fragende mit Sicherheit nicht dahin kommen, wohin er eigentlich hin wollte.

Ein Gespräch weist also eine Struktur auf, die mit Hilfe einer Gesprächsgrammatik untersucht werden kann. Ein Gespräch ist also aus mehreren Abschnitten zusammengesetzt, wobei jeder Abschnitt entweder ein Satz oder eine Satzgruppe sein kann. Die Abschnitte selber werden durch eine **Kohärenzrelation** miteinander verknüpft. Bei dem Häuserbeispiel wäre die Kohärenzrelation ermöglicht, da der erste Satz den zweiten erst möglich macht. Da es viele Ansätze gibt, Kohärenzrelationen aufzustellen werden wir nur die wichtigsten nachfolgend betrachten:

- Ermöglichen oder Verursachen

  Satz 1 erzeugt eine Zustandsänderung (auch implizit), wodurch S2 möglich oder verursacht wird. Beispiel: „*Ich ging nach draußen. Ich fuhr zur Fachhochschule*“, das nach draußen gehen erlaubt hierbei implizit ins Auto zu steigen.
- Erklärung

  Entspricht dem Gegenteil von „*Ermöglichen*“, S2 verursacht oder ermöglicht S1. Beispiel: „*Ich kam zu spät zur Schule, ich hatte verschlafen*“.
- Grundszenario

  S1 beschreibt eine Einstellung oder einen Hintergrund für S2. Beispiel: „*Es war eine dunkle und stürmische Nacht...Restliche Geschichte.*“

- Auswertung

  Aus S2 wird abgeleitet, dass S1 Teil des Plans des Sprechers ist. Beispiel: „*Etwas lustiges war passiert. .... Restliche Geschichte*“
- Veranschaulichung

  S2 ist ein Beispiel für das allgemeine Prinzip S1. Beispiel: „*Dieser Algorithmus kehrt eine Liste um. Die Eingabe [ABC] wird in [CBA] umgewandelt*“,
- Verallgemeinerung

  S1 ist ein Beispiel für das allgemeine Prinzip S2. Beispiel: „[ABC] wird in [CBA] umgewandelt, im allgemeinen kehrt der Algorithmus eine Liste um.“
- Verletzte Erwartung

  Leitet $\neg P$ von S2 ab und negiert die normale Inferenz von P aus S1. Beispiel: „Die Arbeit ist schwach, aber auch interessant“
- und viele mehr

Ein ausführliches Beispiel hierzu soll kurz erläutert werden, es besteht aus insgesamt acht Sätzen:

(1) Gestern ist etwas Lustiges passiert.
(2) Franz fuhr zu einem verrückten Restaurant.
(3) Er bestellte die Ente.
(4) Die Rechnung betrug 50 €.
(5) Franz erschrak, als er feststellte, dass er kein Geld bei sich hatte.
(6) Er hatte seine Brieftasche zu Hause gelassen.
(7) Der Ober sagte, es wäre kein Problem, später zu bezahlen.
(8) Seine Vergesslichkeit war ihm sehr peinlich.

- Satz 1 steht als Auswertung für den restlichen Teil des Gesprächs (Metakommentar des Sprechers)
- Satz 2 ermöglicht Satz 3
- Satz 2 und 3 verursachen Satz 4 mit dem impliziten Zwischenzustand, dass Franz die Ente gegessen hat

- Satz 2 bis 4 dienen als Grundlage für das weitere Gespräch
- Satz 6 ist eine Erklärung von Satz 5
- Satz 5 und 6 ermöglichen Satz 7, in diesem Fall ist es ermöglichen und nicht verursachen, da der Ober auch die Möglichkeit gehabt hätte anders zu reagieren
- Satz 5 bis 7 verursachen Satz 8

Dieses Beispiel zeigt, dass die Kohärenzrelationen die Gespräche zusammenbinden. Sie sind eine Hilfe für den Sprecher etwas mitzuteilen und etwas implizit vorauszusetzen. Dies kann wiederum vom Hörer genutzt werden, um die Absicht des Sprechers zu erkennen, wodurch die Kohärenzrelationen als Filter bei Mehrdeutigkeit im Gesamtkontext eingesetzt werden kann. Wenn die Interpretationsmöglichkeit eines Satzes nicht in den Kohärenzkontext passt, dann kann der Prozess der Auflösung der Mehrdeutigkeit diesen Satz von vornherein ausschließen.

22.7.3 Kombination von Verweisauflösung und Kohärenz

Die Verweisauflösung und die Kohärenz sind eng miteinander verbunden, die Theorie von Grosz und Sidner (1986) berücksichtigt die Aufmerksamkeit eines Gesprächs. Dabei existiert ein Stapel von **Aufmerksamkeitsräumen**. Falls eine Äußerung getätigt wird, so kann diese eine Verschiebung der Aufmerksamkeit bedeuten (z.B.: „*Ach übrigens....*“). Die Aufmerksamkeiten können also auf den Stapel gelegt werden, wenn sie neu erzeugt wurden, oder wieder herunter genommen werden. Ein Beispiel wäre „*Franz fuhr zu einem Restaurant*“, wodurch eine neue Aufmerksamkeit auf den Stapel gelegt wird. Nun ist es möglich, statt „*ein Ober*“ „*der Ober*“ zu sagen, da er im Restaurant ist, wäre er nach Hause gefahren, so würde das Restaurant wieder vom Stapel genommen. Nun wäre es hingegen nicht mehr möglich mit „der Ober“ oder „er“ auf den Ober zu verweisen

22.8 Fazit

In diesem Kapitel ging es vor allem darum, wie man mit Hilfe der KI natürliche Sprache

verarbeiten kann. Dies ist, wie gezeigt, keine leichte Aufgabe, da sie viel Know-How aus unterschiedlichen Aufgabenbereichen fordert (Linguistik, Theoretische Informatik, logische und probabilistische Wissensrepräsentation und Inferenz). Zudem beruht der Aufbau der Wissensbasis auf einer empirischen Untersuchung der menschlichen Sprache und des menschlichen Verhaltens, dies ist eine sehr komplexe Aufgabe, die sehr umfangreich ist, insbesondere, da es nicht immer klar abgesteckte Regeln gibt, wie bei den formalen Sprachen. Ein weiteres Hindernis besteht darin, dass die meisten Mehrdeutigkeiten, Verweise und ähnliches von den Menschen komplett unbewusst richtig interpretiert werden, so dass es schwierig fällt diese Thematik zu analysieren.

23 Kapitel 23 – Probabilistische Sprachverarbeitung

Bisher:

- Kommunikation von Agenten auf Grundlage von Äußerungen in einer gemeinsamen Sprache
- vollständige syntaktische und semantische Analyse der Äußerungen
  - erforderlich, um die Bedeutung der Aussage zu extrahieren
  - möglich, da Äußerungen kurz und domänenspezifisch sind

Jetzt:

- Verwendung einfacher, statistischer Sprachmodelle, um Sammlungen von Millionen von Wörtern zu verarbeiten, und nicht nur einfache Sätze

Korpus-basierter Ansatz

Ein Korpus ist eine große Sammlung von Texten (z. B. Seiten des WWW), die von Menschen für Menschen geschrieben wurden. Aufgabe der Software ist es, dem Menschen dabei zu helfen, die richtigen Informationen zu finden. Hierbei werden Statistik -und Lernmethoden verwendet, um den Korpus nutzen zu können.

23.1 Probabilistische Sprachmodelle

vom logischen zum **probabilistischen Sprachmodell**

- probabilistische Modelle können **aus Daten trainiert** werden (Vorkommen zählen)
- sie sind robuster (können mit kleiner Wahrscheinlichkeit **jede** Zeichenfolge akzeptieren)
- reflektieren die Tatsache, dass nicht alle Sprecher einig sind, welche Sätze Teil einer Sprache sind
- Auflösung von Mehrdeutigkeit: Finden der **wahrscheinlichsten** Interpretation

Ein probabilistisches Sprachmodell definiert Wahrscheinlichkeiten über eine (möglicherweise unendliche) Menge von Zeichenfolgen.

Exkurs: Segmentierung

Die Motivation liegt darin begründet Missverständnisse und Sprache zu verstehen und richtig zu interpretieren. (Beispiel: „*I have a gub*“ in Woody Allens Take the money and run.) Beim Erlernen einer neuen Sprache scheinen zunächst alle Wörter ineinander überzugehen, und nur langsam lernt man, einzelne Wörter aus dem Klangwirrwarr heraus zu hören, bis man das Gesagte schließlich komplett in einen Satz aus verschiedenen Wörtern zerlegen kann. Und dieser Eindruck täuscht nicht: spektographische Analysen zeigen, dass beim Sprechen tatsächlich keine Pausen zwischen Wörtern gemacht werden, diese tatsächlich ineinander übergehen. Ein erstes Problem bei der Spracherkennung und -verarbeitung ist also die korrekte Zerlegung des Inputs in einzelne Wörter.

Ein Satz ist eine Folge von Wörtern

$$w_1..w_n$$

Die Wahrscheinlichkeit für einen Satz ist somit nach der Kettenregel definiert als:

$$P(w_1...w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1w_2)...P(w_n \mid w_1...w_{n-1}) = \prod_{i=1}^{n} P(w_i \mid w_1...w_{i-1})$$

Die meisten dieser Terme sind komplex und schwer abzuschätzen. Deshalb wird zur Annäherung ein Bigram-Modell verwendet.

N-Gram-Modell

Ein ***n*-Gram-Modell** weist jeder Folge von n Wörtern eine Wahrscheinlichkeit P zu.

Unigram-Modell: $P(w)$ ist die Wahrscheinlichkeit des Vorkommens des Wortes *w*. Das Unigram-Modell geht von einer zufälligen Auswahl der Wörter aus. Die

Wahrscheinlichkeit einer Zeichenfolge ist damit das Produkt der Wahrscheinlichkeiten ihrer Wörter $\prod_i P(w_i)$

Bigram-Modell: Ein Bigram-Modell weist jedem Wort bei bekanntem vorhergehenden Wort die Wahrscheinlichkeit

$$P(w_i \mid w_{i-1})$$

zu, um eine Annäherung an

$$P(w_i \mid w_1 ... w_{i-1})$$

zu bestimmen.

Beide Modelle bestimmen die Wahrscheinlichkeiten durch simples Zählen der Vorkommen.

Wörter	Unigramzähler	Vorhergehende Wörter (Bigram-Zähler)							
		of	in	is	on	to	from	model	agent
the	33508	3833	2479	832	944	1365	597	28	24
on	2573	1	0	33	2	1	0	0	6
of	15474	0	0	29	1	0	0	88	7
to	11527	0	4	450	21	4	16	9	82
is	10566	3	6	1	4	2	1	47	127
model	752	8	1	0	1	14	0	6	4
agent	2100	10	3	3	2	3	0	0	36
idea	241	0	0	0	0	0	0	0	0

Abbildung 23.1: Vorkommen der Worte bzw. Wortkombinationen in der englischen Originalausgabe von R/N

„The“ ist das häufigste Einzelwort. Das Bigram „of the“ ist das gebräuchlichste. Durch Ignorieren der Interpunktion haben einzelne Bigrame eine unerwartet hohe Wahrscheinlichkeit (z. B. „on is“), da zum Beispiel Sätze mit „on“ enden und darauf folgende mit „is“ enden können.

Als nächster Schritt kann ein Trigram-Modell definiert werden. Dadurch wird zum

Beispiel der Satz „*Ich esse ein Rübchen*“ wahrscheinlicher als „*Ich esse ein Bübchen*“. Sind Rübchen und Bübchen gleich wahrscheinlich, so weist das Bigram-Modell diesen Dreiwortsätzen die gleiche Wahrscheinlichkeit zu, ein Trigram-Modell würde dies wohl nicht tun.

Ein Unigram-Modell der englischen Originalausgabe von Russel/Norvig erzeugte folgende zufällige Folge von 20 Wörtern:

„*logical are as are confusion may right tries agent goal the was diesel more object then information-gathering search is.*“

Ein Bigram-Modell spuckte folgenden Satz aus:

„*planning purely diagnostic expert systems are very similar computational approach would be represented compactly using tic tac toe a predicate.*“

Der Output eines Trigram-Modells:

„*planning and scheduling are integrated in the success of naive bays model is just a possible prior source by that time.*“

Auch wenn keiner der drei Sätze sinnvoll im eigentlichen Sinne ist, wird schon anhand dieser Stichprobe klar, dass das Trigram- dem Bigram- und das Bigram- dem Unigram-Modell überlegen ist. Auch die Modelle selbst bestätigen das: Das Trigram-Modell weist seiner Zeichenfolge eine Wahrscheinlichkeit von 10^{-10} zu, das Bigram-Modell seinem eine von 10^{-29} und das Unigram-Modell seinem Satz eine Wahrscheinlichkeit von 10^{-59}.

Das Werk von Russel & Norvig enthält mit circa 500000 Wörtern nicht genügend Daten für ein Bigram-, geschweige denn ein Trigram-Modell. Es gibt circa 15000 verschiedene Wörter, also ca. 15000^2 = 25000000 Wortpaare für das Bigram-Modell. Bei mindestens 99,8% dieser Paare wird der Zähler auf 0 stehen. Um zu vermeiden, dass unser Modell

annimmt, all diese Kombinationen seien unmöglich, sollten also die Zählerwerte **geglättet** werden. Die einfachste Weise, diese Glättung durchzuführen, ist eine **+1-Glättung** – jeder Zählerwert wird um 1 inkrementiert.
Wenn der Korpus also N Wörter und B Bigrame enthält, wird jedem Bigram mit dem Zähler c eine Wahrscheinlichkeit von $(c+1)/(N+B)$ zugewiesen. Das eliminiert das Problem der Nullwahrscheinlichkeiten bei n-Gram-Modellen, kann allerdings zu schlechten Schätzungen führen.

Ein weiterer Ansatz ist daher die **Glättung mit Hilfe linearer Interpolation**
Wir definieren die Wahrscheinlichkeitsschätzung als

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = c_3 P(w_i \mid w_{i-2}w_{i-1}) + c_2 P(w_i \mid w_{i-1}) + c_1 P(w_i)$$

mit $c_1+c_2+c_3=1$. Die Parameter c_i können konstant gewählt werden, oder sie können mit einem EM-Algorithmus trainiert werden. Möglich ist zum Beispiel, sie von der jeweiligen Zählerhöhe abhängig zu machen, um so Schätzungen, die von höheren Zählern abgeleitet sind, eine größere Gewichtung geben.

Sprachmodelle bewerten

Eine sinnvolle Methode, Sprachmodelle zu bewerten ist, den Korpus in einen Trainingskorpus und einen Testkorpus zu unterteilen. Das Sprachmodell wird nun mit dem Trainingskorpus trainiert, und anschließend wird die Wahrscheinlichkeit bestimmt, die das Modell dem Testkorpus zuweist. Je höher diese ist, desto besser ist das Modell.
Das Problem bei diesem Ansatz ist, dass $P(w_1 \ldots w_n)$ für lange Zeichenketten sehr klein wird, so dass mit Fließkomma-Fehlern zu rechnen ist, oder die Ergebnisse zumindest schlecht les- und interpretierbar werden.
Um das zu umgehen, kann anstelle der Wahrscheinlichkeit die **Perplexität** bestimmt werden:

$$Perplexität(wörter) = 2^{-\log_2(P(wörter))/N}$$

wobei N die Anzahl der Wörter ist. Je kleiner die Perplexität, desto besser ist das Modell. Eine Perplexität von k bedeutet, dass das Modell jedem Wort eine Wahrscheinlichkeit von

$1/k$ zuweist, die Perplexität kann man sich auch als den durchschnittlichen Verzweigungsfaktor vorstellen.

Anwendungsbeispiel Segmentierung

Segmentierung ist die Ermittlung der Wortgrenzen in einem Text ohne Leerzeichen. (Im Chinesischen und Japanischen Gang und Gäbe, hier aber ein Beispiel aus dem Englischen).

Der Satz

„*Itiseasytoreadwordswithoutspaces*“

ist für Leser die mit dem Englischen vertraut sind, tatsächlich einfach zu lesen. Das liegt jedoch nicht daran, dass wir ein (halbwegs) vollständiges Verständnis über englische Syntax, Semantik und Pragmatik besitzen. Diese Dekodierung kann auch von einem einfachen Unigram(!)-Modell erledigt werden:

Beispiel:

$$P('without') = 0.0004$$
$$P('with') = 0.005;\ P('out') = 0.0008$$
$$P('with\,out') = 0.005 * 0.0008 = 0.000004$$

„*without*“ ist nach dem Unigram-Modell also 100 mal wahrscheinlicher als „with out“.

```
function VITERBI-SEGMENTATION(text, P)
            returns die besten Wörter und ihre Wahrscheinlichkeiten
inputs:     text, eine Zeichenfolge ohne Leerzeichen
            P, eine Unigram-Wahrscheinlichkeitsverteilung über Wörter

n ← LENGTH(text)
words ← leerer Vektor der Länge n + 1
best ← Vektor der Länge n+1, anfänglich überall 0,0
best[0] ← 1,0


// Vektoren best und words durch dynamische Programmierung füllen
for i = 0 to n do
      for j = 0 to i-1 do
      word ← text[j:i]
      w ← LENGTH(word)
      if P[word] * best[i-w] ≥ best[i] then
            best[i] ← P[word] * best[i-w]
            words[i] ← word

// Jetzt die Folge der besten Wörter wieder herstellen
sequence ← die leere Liste
i ← n
while i > 0 do
      schiebe words[i] an den Anfang von sequence
      i ← i – LENGTH(words[i])

// Folge der besten Wörter und Gesamtwahrscheinlichkeit zurückgeben
return sequence, best[i]
```

Abbildung 23.2: Ein auf Viterbi basierender Algorithmus für die Wortsegmentierung

Wir haben hier *n*-Gram-Modelle über Wörter betrachtet, sie lassen sich jedoch vielfältig auch über andere Einheiten verwenden, z. B. über Buchstaben oder Teile der Sprache.

23.1.1 Probabilistische Kontextfreie Grammatiken

n-Gram-Modelle nutzen **Kovorkommen-Statistik** im Korpus, haben aber kein Konzept einer Grammatik für Distanzen > n. Eine möglich Lösung stellen die **probabilistischen kontextfreien Grammatiken (PCFG)** dar.

PCFGs sind CFGs, die jeder Ableitungsregel eine Wahrscheinlichkeit zuweisen, wobei die Summe der Wahrscheinlichkeiten aller Regeln mit derselben linken Seite 1 ergibt.

```
S          → NP VP [1.00]

NP         → Pronomen [0.10]
           | Name [0.10]
           | Substantiv [0.20]
           | Artikel Substantiv [0.50]
           | NP PP [0.10]

VP         → Verb [0.60]
           | VP NP [0.20]
           | VP PP [0.20]

PP         →    Präposition NP [1.00]

Substantiv     →    Wind [0.10] | Wumpus [0.15] | Agent [0.05] | ...
Verb           →    sieht [0.15] | riecht [0.10] | geht [0.25] | ...
Pronomen       →    mich [0.05] | ich [0.25] | es [0.20] | ...
Artikel        →    das [0.30] | ein [0.35] | jedes [0.05] | ...
Präposition    →    zu [0.30] | in [0.25] | auf [0.05] | ...
```

Abbildung 23.3: Beispiel einer PCFG Grammatik mit Lexikon

Im PCFG-Modell ist die Wahrscheinlichkeit einer Folge P(wörter) genau die Summe der Wahrscheinlichkeiten ihrer Parse-Bäume. Mit einem CFG-Chart-Parser[3] können die möglichen Parsing-Verläufe aufgelistet und die Wahrscheinlichkeiten addiert werden. Ist man nur am wahrscheinlichsten Parsing-Verlauf interessiert, so kann eine Variante des Viterbi Algorithmus oder eine Best-first-Suchstrategie angewandt werden.

3 Parse Verfahren, das mächtiger als Top-Down oder Bottom-Up ist

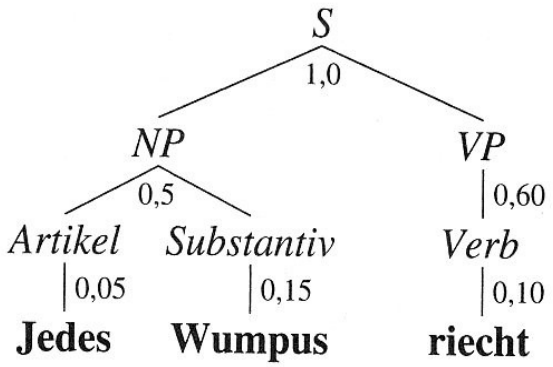

Abbildung 23.4: Parse-Baum für den Satz „Jedes Wumpus riecht“

Ein Problem bei PCFGs ist ihre Kontextunabhängigkeit. So wird der Unterschied in der Wahrscheinlichkeit zwischen zwei Sätzen, die sich nur in einem Wort unterschieden nur von der Wahrscheinlichkeit ebendieses Wortes bestimmt. P(„*eine Banane essen*“) unterscheidet sich von P(„*eine Bandana essen*“) beispielsweise nur durch die Wahrscheinlichkeiten P(„*Banane*“) und P(„*Bandana*“), nicht von der Relation zwischen „*essen*“ und den betroffenen Objekten. Um diese Art der Beziehung zu erhalten, benötigt man ein kontextabhängiges Modell. Hierzu dienen beispielsweise **lexikalisierte PCFGs.** Bei lexikalisierten PCFGs spielt der Kopf einer Phrase[4] eine Rolle für die Wahrscheinlichkeiten umschließender Phrasen. Mit ausreichend vielen Trainingsdaten können wir die Regel für die VP → VP NP auf den Kopf der eingebetteten VP („*essen*“) und den Kopf der eingebetteten NP („*Banane*“) konditionieren. Lexikalisierte PCFGs decken einige Kovorkommen von n-gram-Modellen ab, ebenso die grammatikalischen Einschränkungen von CFG-Modellen.

Ein weiteres Problem kann sein, dass PCFGs kürzeren Sätzen im Allgemeinen eine höhere Wahrscheinlichkeit geben als längeren. In verschiedenen Kontexten kann das zu unerwünschten Ergebnissen führen. Beispiel: Im Wall Street Journal beträgt die durchschnittliche Satzlänge 25 Wörter. Eine PCFG würde dem Satz „*Er schlief.*“ eine sehr hohe Wahrscheinlichkeit zuweisen, im WSJ würde man aber wohl eher einen Satz wie „*Aus zuverlässiger Regierungsquelle wurde berichtet, dass ein Schlafen seinerseits nicht zu leugnen sei*“ lesen – die Sätze sind anscheinend nicht kontextfrei, die Schreiber kennen

4 der Kopf einer Phrase ist das wichtigste Wort, z.B. ein Substantiv bei einer Substantivphrase

anscheinend die erwartete Satzlänge. Dies ist in einer PCFG schwer darzustellen.

23.1.2 Wahrscheinlichkeiten für PCFGs lernen

Das Erzeugen einer PCFG erfordert den gleichen Aufwand wie der einer CFG, wobei zusätzlich noch die Wahrscheinlichkeiten bestimmt werden müssen. Dies legt nahe, dass das Lernen der Grammatik einem Wissens-Engineering-Ansatz vorzuziehen ist. Wie bei der Spracherkennung gibt es zwei Sorten von Daten, aus denen gelernt werden kann: geparste und nicht geparste. Das Lernen aus einem Korpus von Linguisten geparsten Bäumen ist sehr viel einfacher: Für jedes Nonterminalsymbol werden alle verschiedenen Kindknoten gezählt (und evtl. wird geglättet).
Beispiel: Gibt es beispielsweise 100.000 Vorkommen des Symbols NP und 20.000 mal davon sind die Kindknoten [NP,PP], so wird die Regel NP → NP PP [0.20] erzeugt.

23.1.3 Regelstruktur für PCFGs lernen

Für eine unbekannte Grammatik gilt: Der Raum möglicher Regelmengen ist **unendlich**. Dieses kann mit Hilfe der **Chomsky Normalform** (nur für kleinere Grammatiken effektiv) umgangen werden:

Jede Regel befindet sich in der Form X → Y Z oder X → t. Jede kontextfreie Grammatik lässt sich als CNF-Grammatik ausdrücken die dieselbe Sprache erkennt. Beschränkung auf n Nonterminalsymbole ergibt $n^3 + nv$ Regeln, v ist die Anzahl der Terminalsymbole.

Bayessches Modellmischen ist ein Ansatz für umfangreichere Grammatiken. Hierbei werden lokale Modelle für jeden Satz erstellt und dann Modelle anhand minimaler Beschreibungslänge gemischt. Vergleichbar mit SEQUITUR-Modell (Abschnitt 22.8).

23.2 Informationsermittlung – Information Retrieval (IR)

Informationsermittlung ist die Aufgabe, Dokumente zu finden, die den Informationsbedarf eines Benutzers erfüllen. (z.B. Suchmaschinen im WWW).

Informationsermittlungssysteme (IR) werden durch folgende Komponenten charakterisiert:

- Eine **Dokumentensammlung**
  Je nach System kann ein Dokument verschiedenes sein: ein Absatz, eine Seite, ein mehrseitiger Text,...
- Eine **in der Abfragesprache formulierte Abfrage**
  Die Abfrage spezifiziert, was der Benutzer wissen will. Abfragesprache kann eine einfache Liste von Wörtern sein: [KI Buch], Folge von Wörtern: [„KI Buch“], sie kann boolesche Operatoren enthalten: [KI AND Buch], oder auch nicht-boolesche Operatoren [KI Buch SITE:aaai.org]
- eine **Ergebnismenge**
  Die Untermenge der Dokumente die das IR für **relevant** hält. Relevant heißt, das Dokument ist von wahrscheinlichem Nutzen für die Person, die die Abfrage gestellt hat, für die Information, die in der Abfrage angefordert wurde
- eine **Präsentation der Ergebnismenge**
  zum Beispiel eine einfache geordnete Liste, oder auch auf einen dreidimensionalen Raum projizierte Abbildung der Ergebnismenge

Aufbau eines IR-Systems:

Naiver Ansatz: Dokumentensammlung in eine Wissensbasis logischer Sätze parsen, jede Abfrage parsen und Wissensbasis mit ASK nach Antworten fragen. Es ist jedoch zu schwierig ein Lexikon und eine Grammatik aufzubauen die eine große Dokumentensammlung abdecken. Dies führt dazu, dass alle IR-Systeme einfache Sprachmodelle verwenden.
Die ersten IR-Systeme verwenden das so genannte Boolesches Schlüsselwortmodell. Jedes

Wort in der Dokumentensammlung ist ein boolesches Merkmal, das für ein Dokument wahr ist, wenn das Wort in ihm vorkommt. Abfragesprache ist die Menge der booleschen Ausdrücke über den Merkmalen. Ein Dokument ist relevant, wenn der Ausdruck wahr ist. Diese ist zwar einfach zu implementieren und zu erklären, aber die Relevanz wird durch ein Bit bestimmt, wodurch es keine Sortierung der Relevanz gibt. Außerdem ist nicht immer klar, welche Abfrage versucht werden soll: Wenn „*A AND B AND C AND D*“ die leere Menge zurück gibt, kann „*A OR B OR C OR D*“ eine viel zu große Menge zurückgeben, und die beste Kombination ist nicht unbedingt klar.

Lösung: die meisten IR verwenden statistische Modelle, die auf Wortzählungen (und manchmal anderen einfachen Funktionsmerkmalen) basieren. Für eine Abfrage sollen die Dokumente gefunden werden, die am wahrscheinlichsten relevant sind, das heißt P (R = true | D,Q) soll maximiert werden mit

- R: Zufallsvariable, die die Relevanz angibt
- D: Dokument
- Q: Abfrage

Die Ergebnismenge soll als geordnete Liste mit absteigender Wahrscheinlichkeit der Relevanz präsentiert werden. Es gibt mehrere Möglichkeiten die gemeinsame Verteilung P (R = true | D,Q) zu zerlegen. Eine dieser Möglichkeiten ist der Ansatz der Sprachmodellierung. Für jedes Dokument wird ein Sprachmodell abgeschätzt und dann unter dessen Verwendung für jede Abfrage die Wahrscheinlichkeit der Abfrage berechnet. Wenn r für R = true steht, kann man die Wahrscheinlichkeiten wie folgt umschreiben:

$$
\begin{aligned}
P\{r|D,Q\} &= \frac{P\{D,Q|r\}P\{r\}}{P\{D,Q\}} \; durch\; Bayessche\; Regel \\
&= \frac{P\{Q|D,r\}P\{D|r\}P\{r\}}{P\{D,Q\}} \; durch\; Kettenregel \\
&= \frac{\alpha P\{Q|D,r\}P\{r|D\}}{P\{D,Q\}} \; Bayessche\; Regel\; für\; feste\; D
\end{aligned}
$$

Wir versuchen, P (r | D,Q) zu maximieren, aber wir können auf äquivalente Weise auch das Chancenverhältnis P(r | D,Q) / P (¬ r | D,Q) maximieren. Das bedeutet, wir können die Dokumente nach deren Bewertung anordnen:

$$\frac{P\{r|D,Q\}}{P\{\neg r|D,Q\}} = \frac{P\{Q|D,r\}P\{r|Q\}}{P\{Q|D,\neg r\}P\{\neg r|D\}}$$

Dieses führt zu der Annahme, dass ein Dokument für irrelevante Dokumente von der Abfrage unabhängig, bzw. wenn ein Dokument für eine Abfrage irrelevant ist, lässt sich durch Kenntnis des Dokuments nicht herausfinden, wie die Abfrage aussieht:

$$P\{D,Q|\neg r\} = P\{D|\neg r\}P\{Q|\neg r\}$$

Damit erhalten wir:

$$\frac{P\{r|D,Q\}}{P\{\neg r|D,Q\}} = P\{Q|D,r\}\,x\,\frac{P\{r|D\}}{P\{\neg r|D\}}$$

Der Faktor P (r | D) / P (¬r | D) ist die von der Abfrage unabhängige Chance, dass das Dokument relevant ist – also ein Maß für die Dokumentenqualität. (Für Artikel in akademischen Zeitschriften zum Beispiel Anzahl der Zitate, für WWW-Seiten Anzahl der Hyperlinks (bzw. die Qualität der Quellen noch mit einfaktorisieren), evtl. Alter, etc..

Der erste Faktor P (Q | D, r) ist die Wahrscheinlichkeit einer Abfrage für ein bestimmtes relevantes Dokument. Um diese abzuschätzen muss ein Sprachmodell gewählt werden, das bestimmt in welcher Beziehung Abfragen zu relevanten Dokumenten stehen. Besonders gebräuchlich ist die Darstellung von Dokumenten mit Hilfe eines Unigram-Wortmodells. Im Bereich der IR spricht man auch vom Wortmultimengen-Modell, da es auf die Häufigkeit jedes Wortes im Dokument ankommt und nicht auf ihre Reihenfolge („*Mann beißt Hund*“ verhält sich identisch zu „*Hund beißt Mann*“. Der Sinn der Sätze ist zwar

verschieden, aber bei Abfrage nach Hund und beißen sind beide durchaus relevant).
Um die Wahrscheinlichkeit einer Abfrage für ein bestimmtes relevantes Dokument zu berechnen, multiplizieren wir die Wahrscheinlichkeit der Wörter in der Abfrage gemäß dem Dokument-Unigram-Modell. (naives Bayessches Modell der Abfrage). Sei Qj das j-te Wort in der Abfrage, so erhalten wir:

$$P(Q|D,r)=\prod_j P(Q_j|D,r)$$

Vereinfachung vornehmen:

$$\frac{P\{r|D,Q\}}{P\{\neg r|D,Q\}}=\prod_j P(Q_j|D,r)\frac{P\{r|D\}}{P\{\neg r|D\}}$$

Beispiel:

Abfrage [Bayes information retrieval model] an eine Sammlung auf fünf Kapiteln des KI Buches. Wir gehen von gleicher Qualität der Kapitel aus, interessieren uns also nur für die Wahrscheinlichkeit der Abfrage für jedes Dokument. Dafür werden zwei Modelle verwendet zum einen das Di Modell – ein nicht geglätteter Schätzwert und zum anderen das D'i Modell unter Verwendung von +1-Glättung. Das geglättete Modell ist unempfindlicher gegenüber Rauschen und kann einem Dokument, dass nicht alle Wörter enthält eine Wahrscheinlichkeit > 0 zuweisen. Das nicht geglättete Modell hat den Vorteil, dass Sammlungen mit vielen Dokumenten einfacher zu berechnen sind. Dazu wird ein Index erstellt, der auflistet, welche Dokumente jedes Wort erwähnen, anschließend wird eine Schnittmenge erzeugt und P (Q | Di) für Dokumente innerhalb der Schnittmenge berechnet.

Wörter	Abfrage	Kap. 1	Kap. 13	Kap. 15	Kap. 22	Kap. 23
Bayes	1	5	32	38	0	7
information	1	15	18	8	12	39
retrieval	1	1	1	0	0	17

model	1	9	7	160	9	63
N	4	14680	10941	18186	16397	12574
P(Q|Di, r)		1,5 E-14	2,8 E-13	0	0	1,2 E-11
P(Q|D'i, r)		4,1 E-14	7,0 E-13	5,2 E-13	1,7 E-15	1,5 E-11

Abbildung 23.5: Beispiel für ein probabilistisches IR-Modell

Die Abbildung 23.5 zeigt ein probabilistisches IR-Modell für die Abfrage „Bayes information retrieval model“. Als Grundlage wurde eine Dokumentensammlung aus fünf Kapiteln aus Russel/Norvig verwendet. N bezeichnet den Wortzähler für das gesamte Dokument, in den Zeilen darüber stehen die Wortzähler für die einzelnen Wörter. Di ist ein nicht geglättetes Unigram-Wortmodell des i-ten Dokumentes und D'i ist dasselbe Modell mit +1-Glättung. Das Kapitel 23 ist mehr als 200 mal wahrscheinlicher als alle anderen Kapitel.

23.2.1 IR-Systeme bewerten

Um zu bewerten, ob ein IR-System zufriedenstellende Leistungen vollbringt, muss ein Experiment durchgeführt werden, in dem ein System eine Menge von Abfragen übergeben wird und die Ergebnismengen im Hinblick auf menschliche Relevanzbeurteilungen bewertet werden. Dafür gibt es zwei Maße: **Recall** und **Precision**. Die Genauigkeit (Precision) gibt an, welcher Anteil der Dokumente in der Ergebnismenge tatsächlich relevant ist. Der Recall gibt an, welcher Anteil der relevanten Dokumente in der Sammlung auch tatsächlich in der Ergebnismenge auftauchen. Irrelevante Dokumente in der Ergebnismenge werden auch als „false positives“ und relevante Dokumente die nicht berücksichtigt wurden als „false negatives“ bezeichnet.

Beispielabfrage:

	In der Ergebnismenge	Nicht in der Ergebnismenge
Relevant	30	20
Nicht relevant	10	40

Für diese Beispielabfrage liegt die Genauigkeit bei 0,75 und der Recall bei 0,60.

Genauigkeit = 30 / (30 + 10) = 0,75

false positives = 1 – 0,75 = 0,25

Recall = 30 / (30 + 20) = 0,60

false negative = 1 – 0,60 = 0,40

Recall ist in sehr großer Dokumentenmenge schwer zu berechnen (siehe WWW). Daher muss man den Recall entweder durch Stichproben abschätzen oder ihn ganz ignorieren und nur die Genauigkeit beurteilen. Die Abwägung der beiden Größen Recall und Genauigkeit ist nicht ganz einfach. Gibt ein System beispielsweise alle Dokumente zurück so erzielt es einen Recall von 1, wird aber eine sehr geringe Genauigkeit haben. Gibt ein System hingegen nur ein relevantes Dokument zurück, so erzielt es eine Genauigkeit von 1 bei einem sehr niedrigen Recall.

Recall und Genauigkeit sind aus der Zeit, als IR hauptsächlich für Bibliothekare eingesetzt wurden, die ein besonderes Interesse an sorgfältigen Ergebnissen haben. Heute sind die meisten Suchanfragen von Internetbenutzern, die weniger an einer großen Sorgfalt interessiert sind, als daran sofort eine Antwort zu erhalten.

Es ist somit ein neues Maß erforderlich, die reziproke Bewertung. Wenn das erste Dokument relevant ist, erhält das System für die Abfrage eine Bewertung von 1, sind das erste und zweite irrelevant und erst das dritte ist relevant, so erhält es eine Bewertung von 1/3.

Ein alternatives Kriterium ist die mittlere Antwortzeit (wie lange dauert es, bis ein Benutzer die gewünschte Antwort für ein Problem erhält) – Das Problem das sich hierbei

ergibt ist, dass Tests mit immer neuen Versuchsgruppen durchgeführt werden müssen.

23.2.2 IR-Verbesserungen

Das Unigram-Modell behandelt alle Wörter als vollständig voneinander unabhängig, aber bestimmte Wörter stehen in einer Beziehung zueinander – zum Beispiel ist „Bett“ eng mit „Betten“ oder auch mit „Schlafsofa“ verwandt. Viele IR-Systeme versuchen, das zu berücksichtigen.

Eine erste Verbesserung wäre es, die Anfrage in Kleinbuchstaben umwandeln. Eine nächste Verbesserung wäre die Verwendung eines Stamm-Algorithmus. Wörter wie „Betten“ werden auf ihren Stamm „Bett“ reduziert. Damit erhält man normalerweise eine Steigerung des Recalls (für Englisch ca. 2%), die Genauigkeit kann aber leiden. (vgl. „stock“ und „stocking“ - Genauigkeit von Abfragen über Fußbekleidung und Aktienmärkte sinkt, Recall von Abfragen über Warehousing steigt). Stamm-Algorithmen, die auf Regeln basieren (zum Beispiel entfernen von „-ing“) können dieses Problem nicht vermeiden. Algorithmen, die auf Wörterbüchern basieren (und die Anpassungen nur dann vornehmen, wenn das Wort so nicht im Wörterbuch steht) jedoch schon. Stammbildung ist im Gegensatz zum Englischen gerade in Sprachen wie der deutschen, in der frei neue Worte durch Zusammensetzung gebildet werden können, oder auch zum Beispiel dem Finnischen, das rekursive morphologische Regeln[5] hat, besonders wichtig.
Nächster Schritt wäre es, Synonyme zu erkennen. Wie bei der Stammbildung sind hier kleine Gewinne beim Recall möglich, unter der Gefahr, an Genauigkeit einzubüßen. („*Sprachen verabscheuen absolute Synonymheit wie die Natur das Vakuum*“).
Des weiteren können Rechtschreibkorrekturroutinen angewandt werden, um Fehler in Dokumenten und in Abfragen zu finden und zu verbessern.
Außerdem können Metadaten berücksichtigt werden, sofern sie vorhanden sind.

5 Morphologische Regeln sind Wortbildungsregeln.

23.2.3 Präsentation von Ergebnismengen

Nach dem Prinzip der Wahrscheinlichkeitsordnung wird die Ergebnismenge als geordnete Liste, sortiert nach Relevanz präsentiert. Dies ist sinnvoll, wenn der Nutzer alle relevanten Dokumente schnell finden möchte. Es kann jedoch zu Problemen führen, da der *Nutzen* nicht berücksichtigt wird: Gibt es zwei identische Dokumente, so haben beide die selbe Relevanz – das zweite Dokument ist nach Kenntnis des ersten für den Benutzer aber ohne Nutzen. Viele IR-Systeme besitzen daher Mechanismen um zu ähnliche Ergebnisse zu eliminieren.

Die Leistung eines IR-Systems lässt sich durch Unterstützung eines Relevanz-Feedbacks erheblich verbessern. Hierbei beurteilt der Benutzer, welche Dokumente aus einer ursprünglichen Ergebnismenge relevant waren. Hierauf kann das System einerseits eine zweite Ergebnismenge präsentieren, deren Dokumente Ähnlichkeiten mit den zuvor als relevant klassifizierten haben, es kann außerdem für zukünftige Abfragen lernen, welche Dokumente relevant waren.

Ein alternativer Ansatz zur Präsentation als Liste ist die Darstellung als **beschrifteter Baum**. Um die Dokumente zu ordnen gibt es hierbei zwei Ansätze: **Dokumentenklassifizierung** und **Dokument-Clustering**. Für die Klassifizierung werden die Dokumente in eine vordefinierte Thementaxonomie eingeordnet. (Beispiel: Sammlung neuer Nachrichten – Ordnung in zum Beispiel Weltnachrichten, lokale Nachrichten, Wirtschaft, Unterhaltung und Sport). Dieses Vorgehen ist besonders geeignet, wenn eine Dokumentensammlung eine kleine Menge von Themen gibt. Für allgemeinere Sammlungen wie das WWW eignet sich das Clustering besser. Hierbei wird der Kategoriebaum für jede Ergebnismenge neu erzeugt.

Die Klassifizierung ist ein Problem des überwachten Lernens (siehe Kapitel 18) und kann mit den dort beschriebenen Methoden gelernt werden, ein gebräuchlicher Ansatz sind die Entscheidungsbäume. Wenn es nur wenige Kategorien gibt, wird für eine korrekt beschriftete Trainingsmenge ein einziger Baum erzeugt, dessen Blätter das Dokument

einer Kategorie zuordnen. Ist die Kategorienmenge größer, so wird für jede Kategorie ein Baum erzeugt – die Blätter weisen das Dokument entweder als der Kategorie zugehörig aus oder nicht. Für gewöhnlich sind die Merkmale die an den Knoten getestet werden einzelne Wörter, wie zum Beispiel „Basketball“ für die Kategorie Sport. Für die Klassifizierung von Text wurden „boosted“ Entscheidungsbäume, naive Bayessche Modelle und Support-Vektor-Maschinen verwendet – in vielen Fällen liegt die Genauigkeit für die Boolesche Klassifizierung zwischen 90% und 98%.

Das Clustering ist ein Problem des nicht überwachten Lernens. Der EM-Algorithmus (Kapitel 20.2) kann genutzt werden, um basierend auf einer Mischung eines Gaußschen Modells die erste Schätzung eines Clusterings zu verbessern. Das Dokumenten-Clustering gestaltet sich allerdings als schwieriger, da wir nicht wissen, dass die Daten durch ein sauberes Gaußsches Modell erzeugt wurden, und weil wir es mit einem sehr viel höher dimensionalen Raum zu tun haben. Für dieses Problem wurden mehrere Ansätze entwickelt:

Agglomeratives Clustering

Hierbei wird ein Baum aufgebaut, über den die Cluster bis hinunter zu den einzelnen Dokumenten verfolgt werden können. Zuerst sind alle Dokumente auf einer Ebene – jedes Dokument ist ein separates Cluster. Von nun an werden die beiden Cluster zu einem zusammengefasst die einem Distanzmaß gemäß am nächsten zueinander liegen, bis nur noch ein Cluster übrig ist. Das Distanzmaß zwischen zwei Dokumenten ist ein gewisses Maß der Überlappung zwischen den Wörtern in den Dokumenten. Dies kann man zum Beispiel bestimmen, in dem man die Euklidsche Distanz zwischen den Wortzählervektoren bestimmt. Das Distanzmaß zwischen zwei Clustern kann die Distanz von Mitte zu Mitte (der Cluster) sein, oder aber auch die mittlere Distanz zwischen den Dokumenten in den Clustern. Agglomeratives Clustering benötigt $O(n^2)$ Zeit – bezogen auf die Anzahl der Dokumente.

K-Mittel-Clustering

K-Mittel-Clustering erzeugt eine flache Menge von k Kategorien. Dabei werden zuerst k Dokumente zufällig ausgewählt. Diese Dokumente dienen als Repräsentanten von k Kategorien. Nun wird jedes verbleibende Dokument einem Cluster zugeordnet. Nach diesem Schritt werden die Mittel der Cluster bestimmt und als neue Repräsentanten der k Kategorien verwendet. Nun werden die Zuordnung und Bestimmung der Mittel wiederholt bis zur Konvergenz. K-Mittel benötigt im Schnitt O(n) Zeit, was ein Vorteil gegenüber dem Agglomerativen Clustering ist. Teilweise wird berichtet, die Ergebnisse von K-Mittel seien schlechter als die des AC, teilweise soll es allerdings auch fast dieselbe Leistung erbringen.

Unabhängig vom verwendeten Algorithmus besteht beim Clustering im Gegensatz zur Klassifizierung die Schwierigkeit, die Kategorien zu benennen. Bei der Klassifizierung sind diese vorgewählt. Eine Möglichkeit dieses Problem zu lösen ist, eine Liste mit den häufigsten Wörtern im Cluster darzustellen, eine andere, die Dokumententitel derjenigen Dokumente zu präsentieren, die nahe der Mitte des Clusters liegen.

23.2.4 IR-Systeme implementieren

Das Ziel ist vor allem die Effizienz. Suchmaschinen im Internet sollen beispielsweise aus einer Sammlung mehrerer Milliarden Seiten innerhalb von Zehntelsekunden Ergebnisse zurückgeben. Ein IR-System besteht hauptsächlich aus zwei Komponenten: Dem Lexikon, in dem alle Wörter in der Dokumentensammlung aufgelistet sind, und dem invertierten Index, der alle Positionen auflistet, an denen die Wörter in der Dokumentensammlung auftauchen.

Das **Lexikon** ist eine Datenstruktur, die eine Operation unterstützt: Für ein bestimmtes Wort gibt sie die Position im invertierten Index zurück. Das Lexikon sollte mit Hilfe einer Hash-Tabelle oder einer ähnlichen Datenstruktur implementiert werden, die ein schnelles Nachschlagen gewährleistet. Eine Entscheidung beim Entwurf des Lexikons ist, ob so

genannte Stoppwörter (der, die das, sein, ein, usw.) die wenig Informationsgehalt tragen, ignoriert werden. Sie nehmen Platz ein und verbessern das Suchergebnis kaum. Sollen allerdings Satzabfragen möglich sein, kann es sinnvoll sein, sie beizubehalten. Beispielsweise sind Stoppwörter erforderlich um Abfragen wie „Sein oder nicht sein“ zu beantworten.

Der **invertierte Index** besteht aus einer Menge von Trefferlisten, die die Positionen, an denen die einzelnen Wörter auftauchen speichern. Für das Boolesche Schlüsselwortmodell ist eine Trefferliste einfach eine Liste von Dokumenten. Für das Unigram-Modell ist es eine Liste aus Dokument-Zähler-Paaren. Um die Phrasensuche zu unterstützen muss die Liste außerdem die Position der Wörter *innerhalb des Dokumentes* speichern.

Handelt es sich bei der Abfrage um ein einziges Wort (ca. ein Viertel der Abfragen), so ist die Verarbeitung schnell.

```
function getResultList(Wort, Lexikon, Index, R)
begin
     trefferListe <- Index.get(Lexikon.where(Wort))
     resultList <- leere PriorityQueue mit R Slots
     foreach Dokument, Zaehler in trefferListe do
     if resultList.size < R then
          resultList.add(Dokument, Zaehler)
     else
          resultList.remove(resultList.min)
          resultList.add(Dokument, Zaehler)
     end
     return resultList
end
```

Abbildung 23.6: Routine um eine geordnete Ergebnisliste für eine Ein-Wort-Abfrage zu erzeugen

Die Bearbeitung der Abfrage benötigt so O (H + R log R), wobei H die Anzahl der Dokumente in der Trefferliste ist. Enthält die Abfrage mehr als ein Wort, so müssen wir für n Wörter n Trefferlisten kombinieren, was den Aufwand auf O (nH + R log R) wachsen lässt.

Die bisherige Betrachtung der IR-Systeme erfolgte unter Verwendung des probabilistischen Modells. Reale IR-Systeme verwenden aber wahrscheinlich ein anderes Modell – das **Vektorraum-Modell**. Es verwendet denselben Wortmultimengen-Ansatz wie das probabilistische Modell. Jedes Dokument wird als Vektor von Unigram-Worthäufigkeiten repräsentiert, genau so wie die Abfrage. Die Beispiel-Abfrage [Bayes information retrieval model] ist ein Vektor wie

[0, ..., 1, 0, ..., 1, 0, ..., 1, 0, ..., 1, 0, ...]

Das heißt, dass jedes Wort in der Dokumentensammlung eine Dimension darstellt, die Abfrage enthält für jede Dimension eine Bewertung von 0, außer für die Dimensionen, die tatsächlich in der Abfrage auftauchen. Relevante Dokumente werden ausgesucht, indem die Dokumentenvektoren gesucht werden, die die nächsten Nachbarn zum Abfragevektor sind. Ein Ähnlichkeitsmaß ist das Punktprodukt der Vektoren (geometrisch der Kosinus ihres Winkels - maximiert man den Kosinus zweier Vektoren im selben Quadranten, geht ihr Winkel gegen 0). Algebraisch erhalten wir hohe Bewertungen für Wörter, die häufig im Dokument und in der Abfrage vorkommen.

23.3 Informationsextraktion

Informationsextraktion ist der Prozess, Information in Form von Datenbankeinträgen aus Texten auszulesen, indem der Text durchsucht wird nach bestimmten Objekten, Ereignissen und Relationen zwischen diesen. Informationsextraktionssysteme liegen zwischen Informationsermittlungssystemen und vollständigen Text-Parsern, da sie mehr tun als Dokumente als Wortmultimenge zu betrachten und weniger, als jeden Satz vollständig zu analysieren.

Die einfachsten Systeme zur Informationsextraktion sind **attributbasierte Systeme**. Sie gehen davon aus, dass sich der komplette zu verarbeitende Text auf ein bestimmtes Objekt

bezieht und versuchen Attribute dieses Objektes zu extrahieren. Zum Beispiel soll ein attributbasiertes Extraktionssystem aus dem Text „17 Zoll SXGA Monitor für nur 249,99€“ folgende Relation extrahieren:

$$\exists m\, m \in ComputerBildschirme \wedge Größe(m, Zoll(17)) \wedge Preis(m, €(249{,}99)) \wedge Auflösung(m, 1280, 1024)$$

Attributbasierte Systeme können als Reihe regulärer Ausdrücke aufgebaut werden, mit einem Ausdruck je Attribut. Wird ein Teil gematcht, kann er als Attributwert extrahiert werden, wird kein Teil gematcht, kann über das Attribut nichts ausgesagt werden. Problematisch wird es, wenn mehrere Teile des Textes gematcht werden – dann wird eine Strategie benötigt, um den „richtigen“ Teil auszuwählen. Zum Beispiel könnte man für ein Attribut mehrere Ausdrücke definieren und mit Prioritäten versehen. Dann könnte absteigend nach Priorität gematcht werden, bis ein Textteil gefunden wird. Für einen Preis könnte zum Beispiel erst nach etwas wie „Unser Preis: “ + Zahl + € gesucht werden, und nur bei Nichtzutreffen nach allgemeineren Formulierungen. Eine andere Möglichkeit ist, alle Matches aufzulisten und nach einem bestimmten Verfahren zu wählen. Im Beispiel des Preises könnte man zum Beispiel den niedrigsten Preis auswählen der innerhalb von 50% des höchsten Preises liegt. Das würde bei Texten wie „Listenpreis 99€, Sonderpreis 79€, Versand 3€“ zum Erfolg führen.

Einen Schritt weiter sind die **relational basierten Extraktionssysteme**. Sie beschäftigen sich mit mehreren Objekten und mit den Relationen zwischen ihnen. Diese Systeme müssen z.B. nicht nur erkennen, dass „249€“ ein Preis ist, sondern auch zu welchem Objekt dieser Preis gehört. Ein typisches Beispiel ist FASTUS, das Nachrichten über Unternehmensfusionen und Akquisitionen verarbeitet. Es kann beispielsweise folgende Geschichte lesen:

„Bridgestone Sports Co. Berichtet am Freitag, dass es ein Joint Venture in Taiwan mit einem lokalen Unternehmen und einem japanischen Handelspartner eingerichtet hat, um Golfschläger zu produzieren, die nach Japan geliefert werden.“

und erzeugt daraus einen Datensatz wie den folgenden:

$$e \in JointVentures \wedge Produkt(e, \text{Golfschläger}) \wedge Datum(e, \text{Freitag}) \wedge .$$
$$. \wedge Unternehmen(e, \text{Bridgestone Sports Co.}) \wedge Unternehmen(e, \text{einem lokalen Unternehmen}) \wedge .$$
$$. \wedge Unternehmen(e, \text{einem japanischen Handelspartner})$$

Relationale Extraktionssysteme werden häufig als Kaskade endlicher Transducer angeordnet, also als Folge endlicher Automaten, die einen Text als Eingabe erhalten, ihn in ein anderes Format übersetzen und zum nächsten Automaten weiterleiten. FASTUS zum Beispiel besteht aus fünf Phasen:

1. Token-Bildung
2. komplexe Wortverarbeitung
3. grundlegende Gruppenverarbeitung
4. komplexe Phrasenverarbeitung
5. Strukturverarbeitung

Token-Bildung: Der Eingabezeichenstrom wird in einzelne Token (Wörter, Zahlen, Interpunktionszeichen) unterteilt. Für das Englische und auch das Deutsche ist dieser Schritt sehr einfach: Man trennt den Strom an Leerzeichen und Interpunktionszeichen. Im Japanischen muss zum Beispiel erst einmal eine Segmentierung angewandt werden (beispielsweise. mit dem Viterbi-Segmentierungsalgorithmus).

Die **komplexe Wortverarbeitung** erkennt komplexe Wörter einschließlich Wortkombinationen wie zum Beispiel „so genannt“, „Joint Venture“ oder Namen wie „Bundeskanzlerin Angela Merkel“ oder wie im obigen Beispiel „Bridgestone Sports Co.“. Diese Erkennung erfolgt durch eine Kombination lexikalischer Einträge und endlicher Grammatikregeln. Beispiel für eine Regel für Firmennamen:

GroßgeschriebenesWort + („Company“ | „Co.“ | „GmbH“ | „AG“)

Bei der Erzeugung dieser Regeln ist besondere Sorgfalt geboten, sonst ergeht es einem wie dem kommerziellen System, dass „Intel Chairman Andy Grove“ nicht als Person sondern als Ort erkannte, da es eine Regel für Ortsnamen wie die folgende gab:

GroßgeschriebenesWort+ („Grove“ | „Forest“ | „Village“ | ...)

Bei der **grundlegenden Gruppenverarbeitung** werden aus den Wörtern grundlegende Gruppen wie Substantivgruppen, Verbgruppen, etc. gebildet, um sie den späteren Phasen zur Weiterverarbeitung zu übergeben. Eine Substantivgruppe kann aus einem Substantivkopf, optional einem Artikel und eventuell anderen Modifikatoren bestehen. Eine Verbgruppe aus einem Verb und zugeordneten Hilfswörtern, lässt aber zum Beispiel Objekt- und Präpositionalphrasen außen vor. Aufgrund dieser Einschränkungen zu den Nominalphrasen (NP) in ε_1 reichen hier reguläre Grammatikregeln aus, zudem kann diese Phase auch von einem endlichen Automaten erledigt werden. Die Aufteilung des Beispielsatzes wäre:

NG („Bridgestone Sports Co.“), VG(„berichtet“), NG(„Freitag“), NG(„es“), VG(„eingerichtet hat“), NG(„ein Joint Venture“), PR(„in“), NG(„Taiwan“), PR(„mit“), NG(„einem lokalen Unternehmen“),...

In der vierten Phase werden die grundlegenden Gruppen zu **komplexen Phrasen** kombiniert. Auch hier wird versucht, endliche Regeln zu erhalten, die schnell verarbeitet werden können. Am einfachsten sind Kombinationsregeln für domänenspezifische Ereignisse. Beispielregel um die Bildung von Joint Ventures zu beschreiben:

Unternehmen+ Einrichten JointVenture („mit“ Unternehmen+)?

Die vierte Phase ist die erste, in der die Ausgabe sowohl in den Ausgabestrom als auch in eine Datenbankschablone platziert wird.

Die letzte Phase **mischt die Strukturen** die in der vierten Phase erzeugt wurden. Folgt auf

den Beispielsatz der Satz „Das Joint Venture beginnt im Januar mit der Produktion“ wird in dieser Phase erkannt, dass es zwei Verweise auf ein Joint Venture gibt, und vermischt sie zu einem einzigen.

Die Informationsextraktion funktioniert dann gut, wenn sie auf begrenzte Domänen angewandt wird, für die im Voraus festgelegt werden kann, welche Themen diskutiert werden, und wie diese beschrieben werden. In zahlreichen Domänen hat sie sich als praktisch erwiesen, ist jedoch kein Ersatz für vollständige natürliche Sprachverarbeitung.

23.4 Maschinenübersetzungen

Maschinenübersetzung ist die automatische Übersetzung aus einer natürlichen Sprache in eine andere.

Grobübersetzung – das Ziel ist, den Inhalt einer Quelle zu verstehen. Grammatikfehler oder unelegante Sätze werden toleriert, so lange die Bedeutung verständlich ist. Anwendungsbeispiel: Im Web sind Nutzer häufig mit einer Grobübersetzung einer fremdsprachigen Seite zufrieden. Unter Umständen kann eine grobübersetzte Quelle von einem einsprachigen Mitarbeiter überarbeitet werden, ohne die Quelle zu kennen – das kann Geld sparen.

Übersetzung für beschränkte Quellen – Thema und Format des Quelltextes sind genau vordefiniert und stark eingeschränkt. Zum Beispiel TAUM-METEO, ein System dass Wetterberichte aus dem Englischen ins Französische übersetzt. (Es funktioniert da Wetterberichte eine genau definierte Terminologie einhalten).

Vorbearbeitete Übersetzungen – die Quelle wird vor der Übersetzung von einem Menschen derart überarbeitet, dass ihre Sprache einer genau definierten Untermenge der natürlichen Sprache entspricht. Zum Beispiel Handbücher von Caterpillar oder Xerox.

Literaturübersetzung - noch nicht von Maschinen machbar

Problem beim maschinellen Übersetzen: Worte sind meist mehrdeutig, und die Mehrdeutigkeiten „überlappen“ sich in verschiedenen Sprachen. Um immer das richtige Wort zu finden, müsste die Quelle vorbearbeitet werden, zum Beispiel müsste für Englisch „*hard*“ definiert werden, ob es im Sinne von „*hart*“ oder „*schwierig*“ gemeint ist. Beim Übersetzen muss also der Kontext bekannt und verstanden werden – man kann sich nicht entscheiden, ob „*you*“ mit Sie oder Du übersetzt werden soll, ohne sich zwischen formell und informell zu entscheiden (Ob mit „*you*“ der Singular oder der Plural gemeint ist, lässt sich oft aus dem Satz erkennen).

23.4.1 Maschinelle Übersetzungssysteme

Bei maschinellen Übersetzungssystemen gibt es zwei Vorgehensweisen: die einen versuchen, den Text in eine Interlingua-Repräsentation zu überführen und von da in die Zielsprache zu übersetzen. Der Nachteil dieser Methode ist, dass ein vollständiges Sprachverständnis nötig ist, und eventuell keine Ausgabe generiert werden kann, wenn die Analyse fehlschlägt. Der Vorteil ist, dass zwischen n Sprachen mit dem Aufwand O(n) statt $O(n^2)$ übersetzt werden kann. Ein anderer Ansatz ist die Übertragung: dabei wird eine Datenbank mit Regeln oder Beispielen geführt, und sobald die Regel oder das Beispiel eine Übereinstimmung ergibt, wird direkt übersetzt. Diese Übertragungen können auf lexikalischer, syntaktischer oder semantischer Ebene stattfinden. Man bezeichnet Übertragungen die direkt von einem Satz in einen anderen erfolgen als **speicherbasierte Übersetzungsmethode**, da das Konzept ist, sich eine große Menge von Paaren zu merken. Diese Methode ist robust, da sie immer eine Ausgabe erzeugt und immer einige Wörter richtig sein müssen.

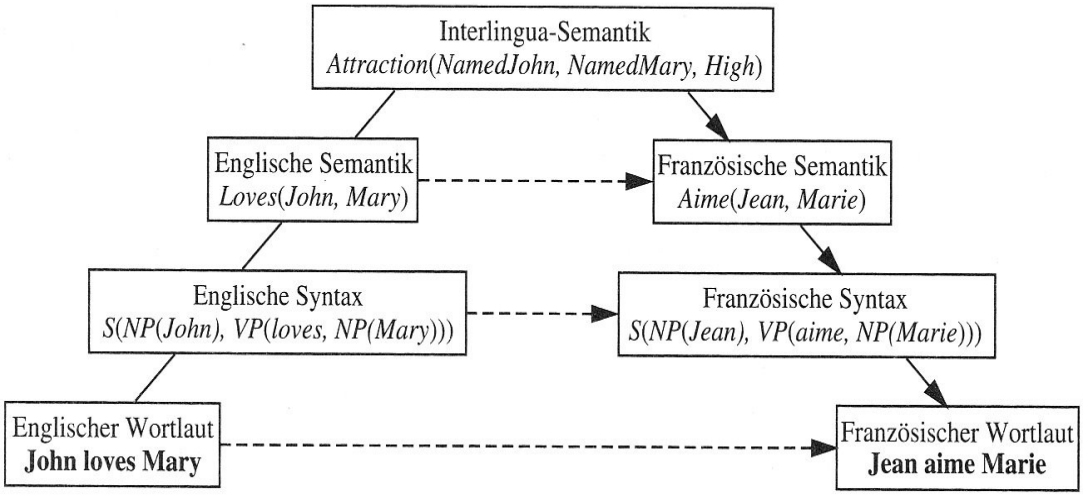

Abbildung 23.7: Schematische Darstellung des Auswahlmöglichkeiten für ein maschinelles Übersetzungssystem

Ein auf Interlingua basierendes System (Abbildung 23.7) folgt den durchgezogenen Linien, ein auf Übertragung basierendes System verwendet die gestrichelten Linien als Abkürzung. Unterschiedliche Systeme nehmen die Übertragung an unterschiedlichen Punkten vor, einige nehmen sie an mehreren Punkten vor.

23.4.2 Statistische Maschinenübersetzung

Problem: einen englischen Satz E in einen französischen Satz F übersetzen:

$$\text{argmax}_F\, P(F|E) = \text{argmax}_F\, P(E|F)P(F)/P(E)$$
$$= \text{argmax}_F\, P(E|F)P(F)$$

Das bedeutet, wir sollen alle französischen Sätze F betrachten und den Satz auswählen der das Produkt P(E|F)P(F) maximiert. P(E) kann ignoriert werden, da er für jedes F gleich ist. Der Faktor P(F) ist das **Sprachmodell** für Französisch (wie wahrscheinlich ist ein bestimmter Satz im Französischen). P(E|F) ist das **Übersetzungsmodell**. Es besagt, wie wahrscheinlich ein englisches Wort als Übersetzung für einen bestimmten französischen Satz ist.

Warum wird P(F|E) im Hinblick auf P(E|F) definiert? Weil ein kausales Modell verwendet werden soll. (Beispiel: Verwendung des kausalen Modells P (Symptome | Krankheit) um P (Krankheit | Symptome) zu bestimmen). Trotzdem: bei Übersetzungen ist keine Richtung mehr oder weniger kausal als die andere. Der Grund, die Bayessche Regel trotzdem anzuwenden ist der, dass ein Sprachmodell P(F) gelernt werden kann, das genauer ist als das Übersetzungsmodell P(E|F) und genauer als die direkte Abschätzung P(F|E). Dadurch wird das Übersetzungsproblem in zwei Schritte geteilt:

1. Übersetzungsmodell P(F|E) verwenden um geeignete französische Sätze zu finden, die die richtigen Konzepte aus den englischen Sätzen darstellen, die aber vielleicht kein „korrektes“ Französisch sind.
2. Sprachmodell P(F) verwenden (für das viel bessere Wahrscheinlichkeitsschätzungen existieren) um die beste Möglichkeit auszuwählen.

Das Sprachmodell P(F) kann jedes Modell sein, das einem Satz eine Wahrscheinlichkeit zuweist. Für Grobübersetzungen sind Bigram-Modelle häufig ausreichend.

Das Übersetzungsmodell ist weitaus schwieriger zu erstellen. Es existiert keine fertige Sammlung Englisch-Französischer Satzpaare, und die Komplexität des Modells ist ungleich größer, da das Kreuzprodukt von Sätzen und nicht nur Sätze betrachtet werden.

Start mit einfachem Unigram-Modell. Für ganz einfache Phrasen ausreichend:

P(the dog|le chien) = P(the|le) * P(dog|chien)

Größtenteils schlägt es jedoch fehl – z.B. Wortreihenfolge: chien = dog und brown = brun aber brown dog = chien brun. Ein weiteres Problem kann eine andere Abbildung als eine 1:1 Abbildung sein. Beispiel: à la maison <-> home. (1:3 bzw. 3:1)

„IBM Model 3“ - ein ebenfalls sehr vereinfachtes Modell, das aber in ca. der Hälfte der

Fälle akzeptable Ergebnisse liefert.

IBM Model 3 bleibt auch bei einem Unigram-Modell, fügt aber Komplikationen hinzu, um es aufzuwerten. Um zum Beispiel das Problem der 3:1 Abbildung für à la maison <-> home zu lösen, führt das Modell das Konzept der **Fruchtbarkeit** ein. À hat eine Fruchtbarkeit von 0, la ebenfalls und maison hat eine Fruchtbarkeit von 1 – daraufhin wird maison in home übersetzt. Das erscheint sinnvoll: à und la sind Wörter mit wenig Inhalt, deren Übersetzung man vernachlässigen kann. Umgekehrt wirkt das ganze zweifelhafter: home erhielte eine Fruchtbarkeit von 3 und würde in home home home überführt – das erste home würde mit à, das zweite mit la und das dritte mit maison übersetzt. Innerhalb des Übersetzungsmodells hätte à la maison dieselbe Wahrscheinlichkeit wie maison la à. (Was vom Sprachmodell ausgeglichen wird). Home direkt in à la maison zu übersetzen erforderte allerdings weitaus mehr Parameter und es wäre schwierig diese aus dem Korpus zu bekommen. Der nächste Schritt im IBM Model 3 ist das umpositionieren der Wörter. Dafür wird Wörtern ein **Offset** zugewiesen. Bei der Übersetzung von „*chien brun*“ in „*brown dog*“ erhält dog einen Offset von +1 und brown einen Offset von -1. Naheliegender Gedanke: Offset vom Wort abhängig machen, z.B. Adjektive im Französischen häufig einen positiven Offset zuweisen, da sie dort hinter den Substantiven stehen. Aber auch dafür wären sehr viele Parameter abhängig, so dass IBM Model 3 den Offset von der Position des Wortes im Satz und von der Satzlänge abhängig macht. Es schätzt die folgenden Parameter ab:

P (Offset = 0 | Position = p, EnglischeLänge = m, FranzösischeLänge = n)

Um den Offset von „*brown*“ in „*brown dog*“ zu ermitteln betrachten wir P (Offset | 1, 2, 2), was +1 mit der Wahrscheinlichkeit 0,3 und 0 mit der Wahrscheinlichkeit 0,7 ergibt. Das Offset-Modell scheint noch zweifelhafter als das Fruchtbarkeitsmodell. Sein Vorteil ist, dass es die vorhandenen Daten sinnvoll nutzt. Das insgesamt also sehr mittelmäßige Übersetzungsmodell muss dann durch ein gutes französisches Sprachmodell gerettet werden.

Französische Quelle	Le	chien	brun	n'	est	pas	allé	à	la	maison
Fruchtbarkeitsmo.	1	1	1	1	1	0	1	0	0	1
Umgewandeltes FR	Le	chien	brun	n'	est		allé			maison
Wortauswahlmodell	The	dog	brown	not	did		go			home
Offset-Modell	0	+1	-1	+1	-1		0			0
Zielenglisch	The	brow n	dog	did	not		go			home

Jetzt können wir die Wahrscheinlichkeit P(F|E) für jedes Paar von (Französisch,Englisch) Sätzen berechnen. Das eigentliche Problem war jedoch, für einen bestimmten englischen Satz den französischen Satz zu finden, der diese Wahrscheinlichkeit maximiert. Da es zu viele Wörter und somit zu viele Sätze der Länge n im Französischen gibt, können wir nicht einfach alle auflisten. Hierzu kann eine Variante des A*-Algorithmus verwendet werden.

23.4.3 Wahrscheinlichkeiten für Maschinenübersetzungen lernen

Modell P(F|E) mit vier Parametereingaben:

Sprachmodell $P(wort_i \mid wort_{i-1})$
Fruchtbarkeitsmodell $P(Fruchtbarkeit = n \mid wort_F)$
Wortauswahlmodell $P(wort_E|wort_F)$
Offsetmodell $P(Offset = 0 \mid Position, Länge_E, Länge_F)$

Selbst bei einem mittleren Vokabular von 1000 Wörtern sind für dieses Modell Millionen von Parametern erforderlich – diese müssen offensichtlich aus Daten gelernt werden, wahrscheinlich aus einem zweisprachigen Korpus.

Segmentierung in Sätze – Die Übersetzungseinheit ist ein Satz, der Korpus wird also in Sätze zerlegt. Satzsegmentierung kann mit 98% Genauigkeit erfolgen. (Normalerweise beenden Punkte einen Satz, aber: „Dr. J. R. Smith from Rodeo St. arrived.“)

Abschätzung des französischen Sprachmodells – $P(wort_i \mid wort_{i-1})$: Die französische Hälfte

des Korpus wird betrachtet und Bigram-Zähler geführt, und eine Glättung vorgenommen.

Sätze zuordnen – Jedem Satz in der englischen Version wird ein Satz in der französischen Version zugeordnet. Meist entspricht der nächste englische Satz dem nächsten französischen, aber es kann Unterschiede geben (2:1 Abbildungen z.B. oder Sätze werden in der Reihenfolge vertauscht – 2:2) Durch Betrachtung der Satzlänge kann z.B. mit Hilfe einer Variante des Viterbi-Algorithmus die Satzzuordnung mit einer Genauigkeit von 90%-99% erfolgen. Bessere Ergebnisse erzielt man unter Berücksichtigung von Meilensteinen – z.B. Zahlen, Eigennamen, oder bestimmte Wörter mit eindeutiger Übersetzung.

Parameter des Übersetzungsmodells:

Abschätzung des anfänglichen Fruchtbarkeitsmodells: P(Fruchtbarkeit = n | $wort_F$): Für einen französischen Satz der Länge m der einem englischen Satz der Länge n zugeordnet ist betrachten wir als Evidenz, dass jedes französische Wort des Satzes eine Fruchtbarkeit von m/n besitzt. Nun muss die gesamte Evidenz über alle Sätze betrachtet werden, um eine Fruchtbarkeits-Wahrscheinlichkeitsverteilung für jedes Wort zu erhalten.

Abschätzung des anfänglichen Wortauswahlmodells: P($wort_E$|$wort_F$): Für $wort_F$ = brun werden alle französisch Sätze untersucht, die das Wort „*brun*“ enthalten. Das Wort, das in den meisten englischen Sätzen die diesem Satz zugeordnet sind, auftaucht, ist die wahrscheinlichste Wort/Wort-Übersetzung für brun.

Abschätzung des anfänglichen Offset-Modells: P(Offset = n | pos, $länge_E$, $länge_F$). Für einen englischen Satz der Länge m der einem französischen Satz der Länge n zugeordnet ist, betrachten wir jedes französische Wort im Satz (an der Stelle i) und jedes englische Wort im Satz (an der Stelle j), das eine wahrscheinliche Wortauswahl für das französische Wort ist und nehmen dies als Evidenz für P(Offset = i-j | i, n, m).

Verbesserung aller Schätzungen – Unter Verwendung eines EM-Algorithmus (Erwartungs-Maximierung) werden die Schätzungen verbessert. Die verborgene Variable ist ein

Wortzuordnungsvektor zwischen Satz-zugeordneten Satzpaaren. Der Vektor gibt für jedes englische Wort die Position des entsprechenden französischen Wortes im französischen Satz an. Beispiel:

Franz. Quelle	Le	chien	brun	n'	est	pas	allé	à	la	maison
Ziel-Engl.	The	brown	dog	did	not		go			home
Wortzuordnung	1	3	2	5	4		7			10

Als Erstes erzeugen wir unter Verwendung der aktuellen Abschätzungen der Parameter einen Wortzuordnungsvektor für jedes Satzpaar. Auf diese Wiese können wir bessere Abschätzungen vornehmen. Das Fruchtbarkeitsmodell wird abgeschätzt indem gezählt wird, wie oft ein Element des Wortzuordnungsvektors mehreren Wörtern oder 0 Wörtern zugeordnet wird. Das Wortauswahlmodell muss jetzt nur noch die Wörter betrachten, die einander zugeordnet sind, statt alle Wörter im Satz, und das Offset-Modell kann jede Position im Satz betrachten, um festzustellen, wie oft es gemäß dem Wortzuordnungsvektor verändert wird. Leider wissen wir nicht sicher, wie die korrekte Zuordnung aussieht und es gibt zu viele davon, um sie aufzulisten. Daher müssen wir nach einigen wenigen Zuordnungen mit hoher Wahrscheinlichkeit suchen und sie nach ihren Wahrscheinlichkeiten gewichten, wenn wir Evidenzen für die neuen Parameterabschätzungen sammeln. Das ist alles, was wir für den EM-Algorithmus benötigen. Aus den Anfangsparametern berechnen wir die Zuordnung, und aus den Zuordnungen verbessern wir die Parameterabschätzungen. Dies wiederholen wir bis zur Konvergenz.

23.5 Fazit

Kontextfreie Grammatiken können durch Erweiterung zu probabilistischen kontextfreien Grammatiken aus Daten gelernt werden, und in die Lage versetzt werden, Mehrdeutigkeiten aufzulösen.

Probabilistische Sprachmodelle, die auf n-gramen basieren, bieten trotz deren offensichtlicher Beschränktheit viele Informationen über eine Sprache. Informationsermittlungssysteme verwenden sehr einfache Sprachmodelle, die auf Unigram-Modellen basieren und bieten für sehr große Text-Korpus eine gute Leistung in Hinblick auf Recall und Genauigkeit. Systeme zur Informationsextraktion verwenden komplexere Modell, die in begrenztem Maße Notationen für Syntax und Semantik beinhalten.

Maschinenübersetzungssysteme wurden mit verschiedensten Techniken implementiert, von vollständiger syntaktischer und semantischer Analyse bis zu statistischen Techniken, die auf Worthäufigkeiten basieren. Bei letzteren lohnt es sich, ein Modell abzuleiten, das verfügbare Daten bestmöglich nutzt, auch wenn es stark vereinfacht.