Statistische Sprachmodelle

Uwe Reichel

12. Juli 2010

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ein Sprachmodell ist eine Repräsentation der Struktur einer Sprache. Strukturen lassen sich auf unterschiedlichen Ebenen finden und beschreiben (Phonologie, Morphologie, Syntax, Semantik, Pragmatik). Wie sich schon an der Vielzahl linguistischer Grammatiktheorien (Generative Grammatik, Kategorialgrammatik, Head Driven Phrase Structure Grammar, Funktionale Grammatik, Tree-Adjoining-Grammar usw.) sehen lässt, gibt es zahlreiche Möglichkeiten, sprachliche Strukturen zu modellieren.

Die Güte eines Modells hängt von der Anzahl der Strukturphänomene ab, die es beschreiben kann, und von der Validität der Vorhersagen, die man anhand des Modells über Sequenzen von Struktureinheiten machen kann (z.B. Phoneme bei der Phonotaktik, Dialogakte bei Dialogmodellen).

Zu unterscheiden sind im Wesentlichen linguistische und statistische Sprachmodelle. Während die Modellierung bei ersteren manuell seitens eines linguistischen Experten vorgenommen wird, werden sprachliche Strukturen bei zweiteren automatisch aus einem großen Datenkorpus extrahiert. Linguistische Ansätze bieten folgende Vorteile:

- gezielter Einsatz linguistischen Wissens,
- Implementierung und Überprüfung von Theorien,
- für viele linguistische Bereiche (z.B. morphologische Analysen) erfolgreicher als statistische Methoden.

Demgegenüber haben statistische Ansätze unter anderem die folgenden Vorzüge vorzuweisen:

- Anpassung an die Tatsache, dass Sprache nicht rein analytisch zu beschreiben ist, sondern auch Prozessen unterworfen ist, die nur stochastisch erfasst werden können,
- Verwendung größerer Datenmengen möglich,

- weit weniger zeitaufwendig,
- Standardverfahren für unterschiedlichste Problemstellungen,
- sehr viel robuster gegenüber neuen Daten,
- adaptierbar auf andere Domänen/Sprachen,
- automatische Aquisition von Weltwissen.

Die Vorteile des statistischen Ansatzes haben im Laufe der letzten Jahrzehnte dazu geführt, dass im Bereich der automatischen Sprachverarbeitung (**Natural Language Processing, NLP**) vermehrt und mittlerweile hauptsächlich statistische Sprachmodelle zum Einsatz kommen. Beispiele hierfür sind Modelle zur Phonem- und Wortvorhersage in der Spracherkennung, zur Graphem-Phonem-Konvertierung in der Sprachsynthese, zur Vorhersage von Dialogakten in Dialogmodellen und zur maschinellen Übersetzung zwischen verschiedenen Sprachen.

In diesem Seminar sollen solche Modelle nach einer Einführung in die Grundlagen der Wahrscheinlichkeits- und Informationstheorie sowie der Statistik vorgestellt werden. Dabei werden hier vorwiegend Sprachmodelle für **symbolische Daten** behandelt, und auf Signale wird nur gelegentlich eingegangen (vgl. hierzu Seminare zur Sprachsignalverarbeitung).

Kapitel 2

Wahrscheinlichkeitstheorie

Die Wahrscheinlichkeitstheorie liefert Vorhersagen über die Auftretenswahrscheinlichkeit von Ereignissen.

2.1 Grundbegriffe

Stichprobe S Menge von Beobachtungen; z.B. Text 'hallo Herr Kaiser', in der die Wörter 'hallo', 'Herr' und 'Kaiser' beobachtet werden; $S = \{$'hallo', 'Herr', 'Kaiser'$\}$.

Grundgesamtheit G Eine Stichprobe S ist Teilmenge einer Grundgesamtheit (**Population**). G bezeichnet die Menge aller potentiellen Untersuchungsobjekte für eine bestimmte Fragestellung; z.B. Sammlung aller möglichen Wortfolgen in einer Sprache. Die Grundgesamtheit als Ganzes ist in der Regel nicht bekannt. Aussagen, die die Grundgesamtheit betreffen (wie z.B. die Auftretenswahrscheinlichkeit des Worts 'hallo'), lassen sich anhand der bekannten Stichproben nur in Form von **Schätzungen** machen.

Zufallsvariable Variable X, die mit bestimmten Wahrscheinlichkeiten bestimmte Werte annimmt, z.B. $X := \mathit{Wort} * \mathit{tritt\ auf}$ mit den Werten 'hallo', 'Herr' und 'Kaiser'.

Ereignis E Belegung der Zufallsvariablen X mit einem bestimmten Wert w. $E : X = w$.[1]

[1] Notation: Für nicht weiter spezifizierte Ereignisse werden in diesem Skript **Großbuchstaben** verwendet. Für Ereignisse in Form konkreter Variablenbelegungen wird zukünftig $X = w$ mit w (also **Kleinbuchstaben**) abgekürzt. Kleinbuchstaben werden für Ereignisse auch dort verwendet, wo immer der Unterschied zwischen Variablen und Ereignissen deutlich gemacht werden muss.

Wahrscheinlichkeit P Zahl zwischen 0 (Unmöglichkeit eines Ereignisses) und 1 (Sicherheit eines Ereignisses); $P(w)$ wird im einfachsten Fall geschätzt mit der **relativen Häufigkeit** von w: $P(w) = \frac{\#(w)}{N}$.[2] $\#(w)$ ist die beobachtete Häufigkeit von Ereignis (z.B. Wort) w, N ist die Größe der Stichprobe S (z.B. Textlänge). z.B.: $P(\text{'hallo'}) = \frac{1}{3}$.

Dieses Beispiel zeigt deutlich, dass die verwendete Stichprobe auf Grund ihrer geringen Größe nicht **repräsentativ** für die Grundgesamtheit aller Äußerungen der Deutschen Sprache ist, denn nicht durchschnittlich jedes dritte geäußerte Wort ist 'hallo'.

diskret vs. kontinuierlich Variablen können diskrete Werte annehmen, d.h. abzählbar[3] viel verschiedene (z.B. Auftreten von Wort w, Wortlänge) oder kontinuierliche Werte (überabzählbar viel verschiedene; z.B. Amplitudenwert in einem nicht digitalisierten Signal)

Ereignisraum σ Menge aller möglichen Ereignisse; $\sigma = \{X = \text{'hallo'}, X = \text{'Herr'}, X = \text{'Kaiser'}\}$

Wahrscheinlichkeitsverteilung P Funktion, die eine Wahrscheinlichkeitsmasse 1 über den Ereignisraum σ verteilt: $P(\sigma) = 1$;[4] in unserem 'Hallo Herr Kaiser'-Beispiel haben wir es mit einer sog. **Gleichverteilung** zu tun, d.h. alle Ereignisse sind gleich wahrscheinlich, nämlich gleich $\frac{1}{3}$.

2.2 Bedingte Wahrscheinlichkeit, Kettenregel

Bedingte Wahrscheinlichkeit: Wahrscheinlichkeit, mit der Ereignis A eintritt, wenn Ereignis B beobachtet wurde, $P(A|B)$.

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{\frac{\#(A,B)}{N}}{\frac{\#(B)}{N}} = \frac{\#(A,B)}{\#(B)} \tag{2.1}$$

Es werden die Fälle gezählt, in denen neben B auch A auftritt. Die Bezugshäufigkeit ist somit nicht mehr die Stichprobengröße N, sondern die Häufigkeit von B $\#(B)$.

[2] Gemäß der vorangehenden Fußnote wird $P(X = w)$ im Folgenden mit $P(w)$ abgekürzt.

[3] Eine Menge von Zahlen wird als *abzählbar* bezeichnet, wenn nicht für zwei beliebige Zahlen z_1 und z_2 dieser Menge eine dritte Zahl z gefunden werden kann, für die gilt: $z_1 < z < z_2$. Enthält eine Menge ein solches z, wird sie als *überabzählbar* bezeichnet.

[4] Von der Notation her wird hier mit P(*) nicht zwischen einer konkreten Wahrscheinlichkeit und der Wahrscheinlichkeitsfunktion unterschieden.

Kettenregel Durch Umformulieren von Gleichung 2.1 erhalten wir:

$$P(A, B) = P(B)P(A|B) \tag{2.2}$$

Die Generalisierung auf mehr als zwei Ereignisse ergibt die für die Sprachmodellierung wichtige Kettenregel:

$$P(w_1, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \ldots P(w_n|w_1, \ldots, w_{n-1}) \tag{2.3}$$

Diese Regel gibt uns die Wahrscheinlichkeit eines Textes, der aus der Wortfolge $w_1, w_2, \ldots, w_n$ besteht.

2.3 Satz von Bayes

Dieser Satz dient dazu, die Abhängigkeit von A und B umzukehren. Dies ist häufig notwendig, um bedingte Wahrscheinlichkeiten schätzen zu können. Mehr dazu in späteren Sitzungen.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.4}$$

Die einfache Herleitung erfolgt über die Formel für die bedingte Wahrscheinlichkeit:

$$\begin{aligned}
P(A|B) &= \frac{P(A, B)}{P(B)} \\
P(B|A) &= \frac{P(A, B)}{P(A)} \\
\frac{P(A|B)}{P(B|A)} &= \frac{\frac{P(A,B)}{P(B)}}{\frac{P(A,B)}{P(A)}} \\
&= \frac{P(A)}{P(B)} \\
P(A|B) &= \frac{P(B|A)P(A)}{P(B)}
\end{aligned}$$

In den meisten in diesem Seminar zu besprechenden Fällen besteht das Interesse darin, dasjenige Ereignis A zu finden, für das $P(A|B)$ maximal wird. $P(B)$ spielt für die Ermittlung von A somit keine Rolle:

$$\hat{A} = \arg\max_A \big[P(A|B)\big] = \arg\max_A \big[\frac{P(B|A)P(A)}{P(B)}\big] = \arg\max_A \big[P(B|A)P(A)\big] \tag{2.5}$$

Beispiel Spracherkennung Gegeben sei ein akustisches Signal (Ereignis B). Es soll die wahrscheinlichste Wortfolge (Ereignis A) ermittelt werden, der dieses Signal zugrundeliegen könnte. Zur Behandlung dieses Problems wird mit Gleichung 2.4 zunächst die Abhängigkeitsrichtung von A und B umgedreht. Im nächsten Schritt wird diejenige Wortfolge A gesucht, mit der der Ausdruck in Gleichung 2.5 maximiert wird. Das Standardverfahren hierzu ist der sog. **Viterbi-Algorithmus**, den wir später noch kennenlernen werden.

2.4 Unabhängigkeit

Sind zwei Ereignisse A und B unabhängig voneinander, so gilt $P(A|B) = P(A)$. Dies führt zur Vereinfachung diverser Berechnungen:

$$P(A, B) = P(A)P(B) \tag{2.6}$$

$$P(A_1, \ldots, A_n) = P(A_1)P(A_2)\ldots P(A_n) \tag{2.7}$$

$$P(A, B|C) = P(A|C)P(B|C) \tag{2.8}$$

Gleichung 2.7 ist als eine Vereinfachung der Kettenregel (vgl. Gleichung 2.3) zu verstehen. Gleichung 2.8 beschreibt die konditionelle Unabhängigkeit von A und B. Zur Lösung von NLP-Problemen muss häufig als vereinfachende Annahme von einer Unabhängigkeit von Ereignissen ausgegangen werden. So lässt sich das Problem der Berechnung von bedingten Wahrscheinlichkeiten mit komplexen bedingenden Ereignissen $P(C|A, B)$ in einer Annäherung so lösen, dass man über Bayes (vgl. Gleichung 2.4) das Abhängigkeitsverhältnis von C und A, B umdreht und vereinfachend die Unabhängigkeit von A und B annimmt. Diese Aufdröselung der zusammengesetzten Wahrscheinlichkeit für A, B führt dann zu Gleichung 2.8.

Beispiel Information Retrieval Hier geht es unter anderem darum, für eine bestimmte Anfrage q relevante Textdokumente zu finden. Die Relevanz R_q eines Dokuments bezüglich q ergibt sich anhand der enthaltenen Worttypes $w_1, \ldots, w_n$. Als relevant sollen diejenigen Dokumente klassifiziert werden, bei denen die Wahrscheinlichkeit $P(R_q|w_1, \ldots, w_n)$ einen festgelegten Schwellwert überschreitet. Auf Grund zu geringer Beobachtungshäufigkeiten kann man diese Wahrscheinlichkeit nicht direkt für alle möglichen Typekombinationen schätzen. Also muss man versuchen, den Ausdruck $w_1, \ldots, w_n$ aufzudröseln. Das geschieht mit Hilfe von Bayes (Gleichung 2.9)[5] und der

[5]In diesem Zusammenhang stellt der Satz von Bayes ein Verfahren dar, mit dem eine angenommene **A-priori-**Wahrscheinlichkeit für ein Ereignis in eine durch weitere empirische Daten gestützte **A-posteriori-**Wahrscheinlichkeit übergeführt wird. $P(R_q)$ ist hierbei die A-priori-Wahrscheinlichkeit, ein zufällig gewähltes Dokument hinsichtlich q als relevant zu beurteilen, $P(R_q|w_1, \ldots, w_n)$ bezeichnet die A-posteriori-Wahrscheinlichkeit der Relevanz eines Dokuments gegeben seine beobachteten Eigenschaften.

vereinfachenden Unabhängigkeitsannahme (Gleichung 2.10):

$$P(R_q|w_1,\ldots,w_n) = \frac{P(w_1,\ldots,w_n|R_q)P(R_q)}{P(w_1,\ldots,w_n)} \quad (2.9)$$

$$= \prod_{i=1}^{n} \frac{P(w_i|R_q)}{P(w_i)} P(R_q) \quad (2.10)$$

Die Werte für $P(w_i)$, $P(w_i|R_q)$ und $P(R_q)$ werden mittels Befragung von Versuchspersonen gewonnen, die darüber urteilen sollen, welche Dokumente ihnen bei bestimmten Informationsbedürfnissen relevant erscheinen.

Da das Auftreten von Wörtern in einem Text nicht unabhängig voneinander ist (sonst würde der Text keinen Sinn ergeben), handelt es sich bei Schätzung von $P(R_q|w_1,\ldots,w_n)$ nur um eine Annäherung und nicht um eine exakte Lösung.

2.5 Schätzen von Wahrscheinlichkeiten

Wir haben für die Entwicklung unserer Sprachmodelle nicht alle Textdaten der Welt zur Verfügung, sondern stets nur einen Ausschnitt. Daher lassen sich die Modellparameter wie Wahrscheinlichkeiten und Interpolationsgewichte nicht ohne Weiteres exakt bestimmen, sondern nur schätzen. Zur Schätzung der Wahrscheinlichkeiten unterscheidet man zwischen parametrischen und nicht-parametrischen Verfahren.

2.5.1 Nicht-parametrisch

Die einfachste Form, Wahrscheinlichkeiten zu schätzen, haben wir im Abschnitt 2.1 schon kennengelernt. Dort wurde einfach die relative Häufigkeit eines Ereignisses w herangezogen: $P(w) = \#(w)/N$ bei Stichprobenumfang N. Dabei wurden vorab keine Annahmen darüber gemacht, wie die Wahrscheinlichkeitsmasse auf die Ereignisse verteilt sein könnte. Dieser simple Ansatz über die relative Häufigkeit wird auch als **Maximum Likelihood Schätzung (MLE)** bezeichnet, da keine Wahrscheinlichkeitsmasse für in der Stichprobe ungesehene Ereignisse abgezogen wird und damit maximal viel dieser Masse auf die gesehenen Ereignisse verteilt wird. Allerdings hat dann beispielsweise das Wort 'Onkel' anhand der in Abschnitt 2.1 gegebenen Daten die Auftretenswahrscheinlichkeit 0. Ein Text, in dem das Wort 'Onkel' vorkäme, hätte gemäß der Kettenregel dann ebenfalls die Wahrscheinlichkeit 0. Dies ist aber nicht im Sinne der NLP, die ja auch mit begrenztem Trainingsmaterial reliable Aussagen für beliebige Texte machen möchte. Um also ungesehenen Ereignissen auch eine Wahrscheinlichkeit zukommen zu lassen, kommen sog. **Smoothing**-Techniken zum Einsatz.

Weiter verfeinern lässt sich eine nicht-parametrische Schätzung mittels Kombination verschiedener Schätzer durch **lineare Interpolation**. Zu diesen Methoden später mehr.

2.5.2 Parametrisch

Lassen sich über den stochastischen Prozess, mit dem die beobachteten Ereignisse zustandekommen, Aussagen machen, so kann man die Auftretenswahrscheinlichkeit der Ereignisse auch parametrisch schätzen, d.h. unter Zuhilfenahme von theoretischen Wahrscheinlichkeitsverteilungen. Zu unterscheiden sind Verteilungen für diskrete und kontinuierliche Zufallsvariablen.

Diskret: z.B. Binomialverteilung

Eine Binomialverteilung liegt vor, wenn:

- die Zufallsvariable binär ist, also nur zwei Ereignisse auftreten können; z.B. Wort w tritt auf/ tritt nicht auf (**Bernoulli-Experiment**), und
- die auftretenden Ereignisse unabhängig voneinander sind.

Ein klassisches Beispiel hierfür ist der Münzwurf. Auch bei NLP-Anwendungen wie dem Extrahieren von Kollokationen[6] kommt diese Verteilung zum Einsatz.

Die durch eine Binomialverteilung gegebenen Wahrscheinlichkeit b sind folgendermaßen zu ermitteln:

$$b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}, \text{ wo} \binom{n}{k} = \frac{n!}{(n-k)!k!} \tag{2.11}$$

Angegeben wird hier die Wahrscheinlichkeit dafür, dass ein Ereignis bei n Versuchen k mal auftritt, wenn es die Auftretenswahrscheinlichkeit p hat. Für das Münzwurfbeispiel würde also die Frage beantwortet werden, wie hoch die Wahrscheinlichkeit b ist, dass eine n mal geworfene Münze k mal auf der Kopfseite landet, wenn die Wahrscheinlichkeit für die Kopflandung gleich p ist. $\binom{n}{k}$ gibt die Anzahl der verschiedenen Kombinationsmöglichkeiten an, k Objekte (hier Kopflandungen) aus n (hier Würfen) zu wählen.

Kontinuierlich: z.B. Normalverteilung

Kontinuierliche Verteilungen sind vor allem bei Modellen für Signale von großer Relevanz, da man es hier mit kontinuierlichen Variablen zu tun hat (Cepstral-Koeffizienten usw.). Sprachmodelle für symbolische Daten, so wie sie in diesem Seminar besprochen werden, operieren im Wesentlichen auf diskreten Variablen, für die oben besprochene diskrete Verteilungen zum Einsatz kommen.

Kontinuierliche Verteilungen werden aber beispielsweise benötigt, um verschiedene Sprachmodelle vergleichend zu evaluieren, wenn die zu evaluierende Messgröße kontinuierliche Werte liefert (wie beispielsweise das Evaluierungsmaß *Entropie*, vgl. Abschnitt 3.2).

[6] Grob definiert als stark zusammenhängende Wörter; später mehr dazu.

Eine dieser Verteilungen ist die Normalverteilung (Glockenkurve), die von zwei Parametern abhängt, dem arithmetischen Mittelwert $\bar{x}$ und der Standardabweichung s.

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \tag{2.12}$$

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{2.13}$$

N bezeichnet die Stichprobengröße, x_i den i-ten Messwert. Mit der Normalverteilung wird die Wahrscheinlichkeit für den Wert x anhand der Parameter $\bar{x}$ und s wie folgt bestimmt:

$$p(x; \bar{x}, s) = \frac{1}{\sqrt{2\pi}s} e^{-\frac{(x-\bar{x})^2}{2s^2}} \tag{2.14}$$

Kapitel 3

Informationstheorie

Die Informationstheorie wurde in den 40er Jahren aus dem Interesse heraus entwickelt, die Menge an Information, die über einen verrauschten Kanal (z.B. Funkverbindung) übertragen wird, zu maximieren. Zu ermitteln sind hierzu allgemein die **Kompressionsfähigkeit** der Daten (mit Hilfe der **Entropie**) und die **Kanalkapazität**.

In vielen Bereichen der NLP-Forschung macht man sich die Erkenntnisse der Informationstheorie zu nutze, beispielsweise bei der maschinellen Übersetzung, der Evaluierung von Modellen und der automatischen Segmentierung sprachlicher Daten, bei der beispielsweise Wörter morphologisch so segmentiert werden, dass damit eine höchstmögliche Datenkompression erreicht wird.

3.1 Information

Der Informationsgehalt IC eines Ereignisses w ist definiert als die Anzahl der *Bits* (also der Ja-Nein-Fragen), die zur Codierung von w benötigt werden. Um eine maximale Kompression der Daten zu erreichen, werden häufige Ereignisse mit weniger Bits belegt als seltene Ereignisse. Der Informationsgehalt eines Ereignisses ist also umgekehrt proportional zu dessen Auftretenswahrscheinlichkeit.

$$IC(w) = log_2 \frac{1}{P(w)} = -log_2 P(w) \quad [Bit] \tag{3.1}$$

So ist beispielsweise im Internationalen Morsealphabet das häufig auftretende ‘e’ ($\cdot$) mit nur einem Bit codiert, während für das seltenere ‘q’ ($--\cdot-$) vier Bits reserviert sind. Gewünschter Effekt dieser Aufteilung sind möglichst kurze Codierungslängen von Nachrichten.

Intuitiv kann man die Information bezeichnen als Grad der Überraschung, der mit dem Auftreten eines Ereignisses einhergeht. Je seltener ein Ereignis, desto überraschender ist es und damit umso informativer.

3.2 Entropie

Unter Entropie $H(X)$ versteht man den Informationsgehalt einer Zufallsvariablen X (der **Informationsquelle**). Dieses Maß gibt an, wieviel Bits im Durchschnitt benötigt werden, um einen Wert w von X, also ein **Ereignis** als Teil einer Nachricht zu codieren.

$$H(X) = -\sum_{w \in X} p(w) \log_2 p(w) \quad [Bit] \tag{3.2}$$

Je höher die Entropie ist, je mehr Bits also durchschnittlich benötigt werden, desto unsicherer ist die Vorhersage eines Ereignisses.

Im Spracherkennungsfall haben wir es mit einer **Sequenz von Zufallsvariablen** $W_{1,n} = W_1, \ldots, W_n$ zu tun, die mit Wörtern belegt werden, die Nachricht besteht hier also in einer **Wortsequenz** $w_1, \ldots, w_n$.

Formuliert man für diesen Fall Gleichung 3.2 um, so erhält man:[1]

$$H(W_{1,n}) = -\sum_{w_{1,n} \in W_1 \mathrm{x} \ldots \mathrm{x} W_n} p(w_{1,n}) \log_2 p(w_{1,n}) \tag{3.3}$$

Die Entropie ergibt sich hier also anhand des durchschnittlichen Informationsgehalts aller möglichen Wortsequenzen $w_{1,n}$. Je höher die Entropie, desto schwerer ist die Vorhersage einer konkreten Sequenz und desto schwerer fällt auch die Spracherkennung.

Die Entropie wächst mit der Länge n der Nachricht (der Wortsequenz). Da man in der Regel Texte verschiedener Länge miteinander vergleichen möchte, muss die Entropie auf n normiert werden. Man erhält die wortweise Entropie, oder **Entropie-Rate**:

$$H_{\mathrm{rate}}(W_{1,n}) = -\frac{1}{n} \sum_{w_{1,n}} p(w_{1,n}) \log_2 p(w_{1,n}) \tag{3.4}$$

Um die Entropie einer **Sprache** L anzugeben, lässt man n gegen unendlich gehen, berücksichtigt also alle Äußerungen, die je in der Sprache L getätigt worden sind und noch getätigt werden.

$$H(L) = \lim_{n \to \infty} \frac{1}{n} H(W_{1,n}) \tag{3.5}$$

[1] $W_1 \mathrm{x} \ldots \mathrm{x} W_n$ bezeichnet das **Karthesische Produkt** der Wertebereiche der Variablen $W_1, \ldots, W_n$, also die Menge aller möglichen Belegungskombinationen, in diesem Fall Wortkombinationen. Aus Übersichtlichkeitsgründen wird dieser Ausdruck zukünftig weggelassen.

3.3 Bedingte Entropie

Dieses Maß kommt bei einigen statistischen Klassifikatoren zum Einsatz, beispielsweise bei Entscheidungsbäumen. Die bedingte Entropie $H(Y|X)$ gibt an, wieviel Information im Durchschnitt *zusätzlich* zu dem Wissen darüber, dass die Variable X den Wert w angenommen hat, nötig ist, um den Wert v der Variablen Y vorherzusagen.

$$\begin{aligned} H(Y|X) &= \sum_{w \in X} p(w) H(Y|X = w) \\ &= \sum_{w \in X} p(w)[- \sum_{v \in Y} p(v|w) \log_2 p(v|w)] \end{aligned} \tag{3.6}$$

3.4 Relative Entropie (Kullback-Leibler-Divergenz)

Die Relative Entropie $D(p||m)$ gibt die Divergenz (die Verschiedenheit) zweier Wahrscheinlichkeitsverteilungen p und m für denselben Ereignisraum X an.

$$D(p||m) = \sum_{x \in X} p(x) \log_2 \frac{p(x)}{m(x)} \tag{3.7}$$

Die Divergenz wird ausgedrückt in der Anzahl an Bits, die zusätzlich benötigt werden, wenn Ereignisse mit der Verteilung p mit Hilfe eines Codes, der auf m basiert, ausgedrückt werden sollen. Zur Erinnerung: zum Erreichen der größtmöglichen Datenkompression stellt ein Code für Ereignisse mit niedriger Wahrscheinlichkeit mehr Bits bereit als für solche mit hoher Wahrscheinlichkeit. Liegen nun unterschiedliche Wahrscheinlichkeitsverteilungen p und m vor, so liefert ein auf m optimierter Code für p keine maximale Kompression mehr, es werden also zusätzliche Bits benötigt.

Die Konditionale Relative Entropie ist folgendermaßen gegeben:

$$D(p(y|x)\|m(y|x)) = \sum_{x} p(x) \sum_{y} p(y|x) \log_2 \frac{p(y|x)}{m(y|x)} \tag{3.8}$$

Anwendung findet dieses Maß beispielsweise beim Vergleich der Originalphonotaktik in einem Aussprachelexikon mit der Phonotaktik, die ein Graphem-Phonem-(G2P)-Konvertierer mit denselben Daten liefert. Hier ist $p(y|x)$ die Wahrscheinlichkeit, mit der im Referenzlexikon Phonem y auf die Phonemvorgeschichte x folgt. $m(y|x)$ liefert die Wahrscheinlichkeit, wie sie anhand der Graphem-Phonem-Konvertierung ermittelt worden ist. Je näher p und m, desto näher sind sich auch G2P-Output und Referenz, und desto besser gibt somit der Konvertierer die Originalphonotaktik des Aussprachelexikons wieder.

Eine weitere Anwendung besteht darin, semantische Ähnlichkeiten zwischen Wortpaaren anhand ähnlicher Distribution in den Trainingsdaten festzustellen (vgl. Abschnitt 7.3).

3.5 Kreuzentropie

Entgegen dem gerade gegebenen Beispiel verhält es sich häufig so, dass zwar die Wahrscheinlichkeitsverteilung m des zu bewertenden Modells bekannt ist, nicht aber die den Daten tatsächlich zugrundeliegende Verteilung p. In diesem Fall operiert man mit der Kreuzentropie.

Ziel bei der Evaluierung von Sprachmodellen ist, die **Anpassungsgüte eines Modells** an die gegebene Wortsequenz $w_{1,n}$ zu ermitteln.

Die Kreuzentropie einer Wortvariablensequenz $W_{1,n}$ mit der zugrundeliegenden unbekannten Wahrscheinlichkeitsfunktion p und der durch ein Modell geschätzten Wahrscheinlichkeitsfunktion m ist wie folgt definiert:

$$\begin{aligned} H(W_{1,n}, m) &= H(W_{1,n}) + D(p||m) && (3.9)\\ &= -\sum_{w_{1,n}} p(w_{1,n}) \log_2 m(w_{1,n}) && (3.10) \end{aligned}$$

Zur Entropie der Wortfolge $W_{1,n}$, die auf der unbekannten zugrundeliegenden Verteilung p basiert, wird somit die relative Entropie $D(p||m)$ hinzuaddiert, also die durchschnittliche Anzahl zusätzlich benötigter Bits zur Codierung von $W_{1,n}$, wenn die Codierung von der Verteilung m abgeleitet ist.

Durch Normierung auf Sequenzlänge n ergibt sich die längenunabhängige Kreuzentropie-Rate:

$$H_{\text{rate}}(W_{1,n}, m) = -\frac{1}{n} \sum_{w_{1,n}} p(w_{1,n}) \log_2 m(w_{1,n}) \tag{3.11}$$

Wie in Abschnitt 3.2 können wir auch die Kreuzentropie für die Sprache L, also eine unendlich lange Wortsequenz, angeben:

$$H(L, m) = -\lim_{n\to\infty} \frac{1}{n} \sum_{w_{1,n}} p(w_{1,n}) \log_2 m(w_{1,n}) \tag{3.12}$$

Wie lässt sich nun das nicht ermittelbare $p(w_{1,n})$ aus der Formel entfernen? Der Ausdruck $\sum_* p(*)*$ bedeutet: "nehme den gewichteten Mittelwert aller $*$" – in unserem Fall von $\log_2 m(w_{1,n})$. Da aber n gegen unendlich geht, ist es nicht mehr nötig, über diverse Stichproben zu mitteln, da ohnehin die komplette Grundgesamtheit, also L, vollständig abgedeckt wird.[2] Da nun also die Mittelwertgewichtung überflüssig wird, erhalten wir:

[2]Voraussetzung hierfür ist, dass L ergodisch und stationär ist, zwei vereinfachende Annahmen, auf die hier nicht weiter eingegangen werden soll.

$$H(L,m) = -\lim_{n\to\infty} \frac{1}{n} \log_2 m(w_{1,n}) \tag{3.13}$$

Da wir natürlich nicht alle Wortfolgen von L zur Verfügung haben, lässt sich in unserer Berechnung die Kreuzentropie nur annähern durch:

$$H(L,m) \approx -\frac{1}{n} \log_2 m(w_{1,n}) \tag{3.14}$$

Je besser das Modell, desto ähnlicher sind sich p und m und desto sicherer sind somit die Vorhersagen bezüglich der Wortsequenz, was sich in einer niedrigeren Kreuzentropie widerspiegelt.

3.6 Perplexität

Die Perplexität als alternatives Evaluierungsmaß für Sprachmodelle lässt sich direkt aus der Kreuzentropie herleiten:

$$PP(L,m) = 2^{H(L,m)} \tag{3.15}$$

Auch hier gilt: je besser das Modell, desto niedriger die Perplexitätswerte.

3.7 Mutual Information (Transinformation)

Die Mutual Information $I(X,Y)$ zwischen den beiden Variablen X und Y gibt die Information an, die diese beiden Variablen übereinander enthalten. Diese wird widergespiegelt durch das Ausmaß der Reduzierung der Unsicherheit über den Wert einer Variablen, wenn der Wert der anderen bekannt ist. Diese Reduzierung der Unsicherheit ist gleich der Differenz zwischen der Entropie von $H(Y)$ ohne Wissen über den Wert von X und der bedingten Entropie $H(Y|X)$ mit Wissen über den X-Wert.

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{3.16}$$

Wie in Gleichung 3.16 zu sehen, ist dieses Maß symmetrisch, d.h. X enthält ebensoviel Information über Y wie Y über X. $I(X,Y)$ ist gleich 0, wenn X und Y unabhängig voneinander sind. Dann nämlich ist $H(Y)$ gleich $H(Y|X)$, d.h. Wissen über X trägt nichts zur Vorhersage von Y-Werten bei.

Es gibt auch eine Mutual Information zwischen Ereignissen w und v, die sog. **Pointwise Mutual Information**:

$$I(w,v) = \log_2 \frac{p(w,v)}{p(w)p(v)} \tag{3.17}$$

Dieses Maß kommt beispielsweise bei der Extrahierung von Kollokationen zum Einsatz.

3.8 Noisy-Channel-Modell

Das Noisy-Channel-Modell lässt sich mit folgendem Diagramm wiedergeben:

Abbildung 3.1: Noisy-Channel-Modell

Dargestellt wird, wie eine Nachricht W encodiert und als Code I durch einen Kanal gesendet wird. Der Kanal ist verrauscht, daher kommt der Code auf der anderen Seite verfremdet als Code O heraus, auf dessen Grundlage die Nachricht W rekonstruiert werden soll: man erhält die Nachrichtenrekonstruierung $\hat{W}$.

Der Empfänger kennt nur Code O, nicht aber Code I. Zum Erhalt von I (und falls gewünscht, auch von W), muss er dasjenige I suchen, womit $P(I|O)$ maximiert wird.

Dieses Modell bietet den formalen Rahmen für eine Vielzahl von NLP-Problemen.

Beispiel Part-of-Speech-(POS)-Tagging Beim POS-Tagging wird versucht, die einer beobachteten Wortfolge O zugrundeliegende POS-Sequenz I zu finden. Die Wortfolge wird also betrachtet als ein auf Grund des verrauschten Kanals "fehlerhafter" Output der eingegeben POS-Sequenz. Die eingegebene POS-Sequenz ist ihrerseits auf Seite des Empfängers unbekannt. Daher muss sie rekonstruiert werden ($\hat{I}$), was sich durch Maximierung von $P(I|O)$ erzielen lässt. Bewerkstelligt wird diese Maximierung mit Hilfe des Satzes von Bayes (vgl. Abschnitt 2.3). Details folgen in einer späteren Sitzung.

$$\hat{I} = \arg\max_I [P(I|O)] = \arg\max_I [P(O|I)P(I)] \tag{3.18}$$

Beispiel Maschinelle Übersetzung Bei der Übersetzung eines Textes aus der Sprache $L1$ in die Sprache $L2$ nimmt man an, dass der Inputtext I in der Zielsprache $L2$ verfasst wurde und auf der anderen Seite des verrauschten Kanals "fehlerhaft" als Text O in der Sprache $L1$ herausgekommen ist. Analog zum obigen POS-Fall kennt der Empfänger Text I nicht und muss ihn daher durch Maximierung von $P(I|O)$ rekonstruieren. Das semantische Modul der Übersetzung sorgt dann für die Rekonstruktion der Nachricht W.

Kapitel 4

Korpusarbeit

4.1 Types, Tokens

Die Wörter in einem Textkorpus werden als **Tokens** bezeichnet, die verschiedenen Wörter als **Types**. Die Anzahl der Tokens entspricht also der Korpusgröße, die Anzahl der Types der Anzahl der verschiedenen Wörter, die in diesem Korpus auftreten. Beide Zahlen sind unter anderem abhängig davon, ob Interpunktion ignoriert wird oder nicht (bei Part-of-Speech-Taggern ist sie beispielsweise wichtig). Die Anzahl der Types hängt unter anderem davon ab, ob zwischen Groß- und Kleinschreibung unterschieden wird oder nicht.

4.2 Tokenizing

Unter Tokenizing versteht man die Segmentierung eines Texts in Tokens. Hierzu bedarf es auch einer Satzsegmentierung, welche im Deutschen im Wesentlichen auf einer erfolgreichen Disambiguierung des Punkts beruht, der auch Abkürzungen und Ordnungszahlen markieren kann.

4.3 Zipf'sches Gesetz

Das Zipf'sche Gesetz besagt folgendes: wenn man die Types eines Texts ihrer Häufigkeit f nach ordnet und ihnen dabei jeweils einen Rang r zuweist, dann gilt folgendes:

$$f \cdot r = k \tag{4.1}$$

$f \cdot r$ ergibt also stets einen konstanten Wert k. Wenn in einem Textkorpus beispielsweise der häufigste Type 10000 mal vorkommt ($r = 1, f = 10000$), dann kommt bereits der 10000-häufigste Type nur noch einmal vor ($r = 10000, f = 1$). Diese Häufigkeitsverteilung ist in Abbildung 4.1 dargestellt.

Daraus ergibt sich das sogenannte *Large number of rare events*-Problem (LNRE) was u.a. zur Folge hat, dass in einem Korpus eine sehr hohe Anzahl an Types nicht ausreichend häufig vorkommt für verlässliche Wahrscheinlichkeitsschätzungen. In den Kapiteln 6 und 7 werden Lösungsansätze für dieses Problem besprochen.

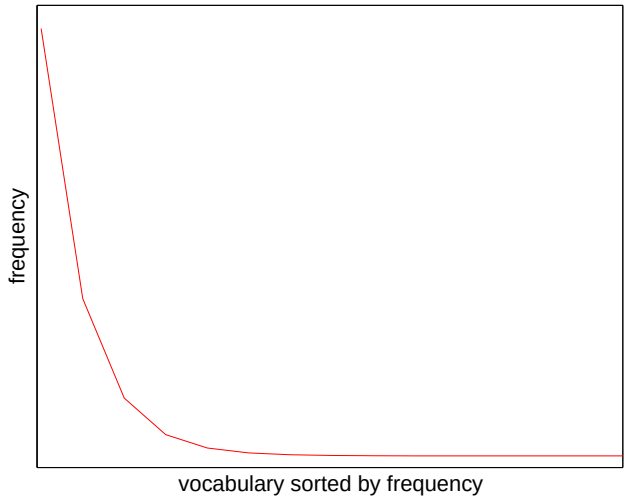

Abbildung 4.1: LNRE-Verteilung

4.4 Korpusgröße

Grundsätzlich gilt, dass Korpora zur Entwicklung tauglicher Sprachmodelle so groß wie möglich sein sollten (mehrere Hunderttausend, besser noch Millionen Tokens), um a) so viele Ereignisse wie möglich abzudecken und b) für möglichst viele Ereignisse ausreichend große Zählwerte zum Schätzen reliabler Wahrscheinlichkeiten zu erhalten (vgl. Zipf'sches Gesetz). Egal wie groß das Korpus auch sein mag: ein Modell wird immer auch mit ungesehenen Daten umgehen müssen (vgl. Kapitel 6).

Eine Obergrenze der Korpusgröße ist in der Speicherkapazität für die Modellparameter gegeben, so steigt die Anzahl der zu speichernden Wahrscheinlichkeiten in einem N-Gramm-Modell (vgl. Kapitel 5) exponentiell zu den hinzukommenden Types.

4.5 Korpusaufteilung

4.5.1 Training, Entwicklung, Test

Für Training und Evaluierung eines statistischen Sprachmodells bedarf es mindestens eines Trainings- und eines Testkorpus, die voneinander unabhängig sein müssen.

In vielen Fällen benötigt man zusätzlich ein Entwicklungskorpus, beispielsweise zum Vergleich mehrerer trainierter Modelle, von denen dann dasjenige, das für das Entwicklungskorpus am besten abgeschnitten hat, auf dem Testkorpus evaluiert wird.

4.5.2 Partitionierung der Trainingsdaten

Steht ein ausreichend großes Trainingskorpus zur Verfügung, empfiehlt es sich das Korpus zu partitionieren, die Modellparameter auf den Partitionen getrennt zu berechnen und dann über die Partitionen gewichtet zu mitteln.

4.5.3 Leaving-One-Out-Methode

Ein Verfahren, das auch mit weniger Trainingsdaten funktioniert, ist die sog. Leaving-One-Out-Methode. Hier werden für ein Trainingskorpus mit N Tokens N Partitionen der Größe $N-1$ erstellt, indem jedes der Tokens in einer der Partitionen weggelassen wird. Auch hier werden die Modellparameter wieder für jede Partition getrennt berechnet und anschließend gemittelt.

4.5.4 Kreuzvalidierung

Um zur vergleichenden Evaluierung zweier Modelle die Varianzen der Modellgüte ermitteln zu können (vgl. Kapitel 8), empfiehlt es sich, das Korpus mehrmals zufällig in Trainings- und Testdaten aufzuteilen und für jede dieser Partitionierungen Training und Test durchzuführen und somit mehrere Performanzwerte zu ermitteln. Dieses Vorgehen wird als Kreuzvalidierung bezeichnet.

Kapitel 5

N-Gramm-Modelle

Für viele NLP-Anwendungen ist es essentiell, Wortsequenzen mit einer gewissen Wahrscheinlichkeit vorhersagen zu können. In der Spracherkennung beispielsweise ist es nicht möglich, das akustische Signal eindeutig in eine Wortfolge zu überführen, was vor allem an seiner hohen Variabilität liegt. Sprachmodelle ergänzen die akustische Erkennung um das Wissen, welche Wortsequenzen überhaupt plausibel sind. Zur Vorhersage von Wörtern sind mehrere Möglichkeiten denkbar:

- Jedes Wort ist gleich wahrscheinlich. Würde der Wortschatz einer Sprache 10000 Wörter umfassen, läge also die Wahrscheinlichkeit jedes Worts bei 1/10000. Mit einer solchen Gleichverteilung ist aber nichts gewonnen.
- Die Vorhersagbarkeit eines Worts beruht auf seiner Auftretenshäufigkeit. Tritt ein Wort in einem Text von 10000 Wörtern 100 mal auf, dann beträgt seine Wahrscheinlichkeit 1/100. Dieser Ansatz wird auch als **Unigramm-Modell** bezeichnet.
- Die Vorhersagbarkeit eines Worts beruht auf dem Wortkontext, im Zusammenhang mit der Spracherkennung also auf den vorangehenden Wörtern, der **Wortvorgeschichte (History)**.

Dieser letzte Ansatz soll jetzt weiter vertieft werden. Die Wahrscheinlichkeit einer Wortsequenz $w_1, \ldots, w_k$ lässt sich mit Hilfe der Kettenregel (vgl. 2.3) folgendermaßen umformulieren:

$$\begin{aligned} P(w_1, \ldots, w_k) &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_k|w_1, \ldots, w_{k-1}) \\ &= P(w_1)\prod_{i=2}^{n} P(w_i|w_1, \ldots, w_{i-1}) \end{aligned} \tag{5.1}$$

Dabei ist die Wahrscheinlichkeit jedes Worts w_i abhängig von der Wortvorgeschichte $w_1, \ldots, w_{i-1}$. Die Wortvorgeschichte beginnt damit jedes Mal

beim Textanfang. Auf diese Weise tritt allerdings ein Problem auf, dass man zum einen keine ausreichend hohen Zählwerte mehr erhält, um die bedingten Wahrscheinlichkeiten überhaupt schätzen zu können. Wenn man für das Trainingsmaterial nur einen Text heranzieht, beträgt jede der bedingten Wahrscheinlichkeiten 1 (warum?). Zum anderen ist unter Verwendung dieser sehr speziellen Wortvorgeschichten der Anteil der gesehenen Daten viel zu gering, als dass das Modell auch auf ungesehene Daten angewendet werden könnte.

Diese Probleme lassen sich dadurch abmildern, dass man $P(w_1, \ldots, w_k)$ nur annäherungsweise löst, in dem man die Länge der Wortvorgeschichte auf m beschränkt. Diese Begrenzung wird als **Markov-Annahme** bezeichnet.

$$P(w_1, \ldots, w_k) = P(w_1)P(w_2|w_1) \prod_{i=3}^{k} P(w_i|w_{i-m}, \ldots, w_{i-1}) \tag{5.2}$$

Setzt man beispielsweise m gleich 2, erhält man eine **Markovkette 2. Ordnung**. Das Wort w_i zusammen mit der Wortvorgeschichte $w_{i-m}, \ldots, w_{i-1}$ wird dann als **N-Gramm** bezeichnet, wobei $N = m + 1$. Bei einer Vorgeschichte der Länge 1 werden sie als **Bigramme** bezeichnet, bei der Länge 2 als **Trigramme**.

Bei der Multiplikation aller Wahrscheinlichkeiten in der Markovkette wird man in längeren Texten früher oder später den Wert 0 erreichen, da nur Werte kleiner 1 aufmultipliziert werden. Um dies zu vermeiden, logarithmiert man die Wahrscheinlichkeiten und addiert sie. Dies ist zulässig aufgrund der Äquivalenz zwischen Multiplikation nicht-logarithmierter und Addition logarithmierter Werte.

Bei der Bestimmung der Länge der Vorgeschichte besteht folgender Trade-Off: je länger, desto stärker lässt sich die Menge der Nachfolgekandidaten einschränken, je kürzer, desto verlässlicher die Schätzung der Wahrscheinlichkeiten aufgrund der höheren Zählwerte. Eine Vorgeschichte der Länge > 3 ist somit nur möglich, wenn das Trainingskorpus sehr groß ist (etwa > 1 Mio. Tokens) und die Schätzung mit den im nächsten Kapitel beschriebenen Methoden weiter verfeinert wird.

Die N-Gramm-Wahrscheinlichkeit von Wort w_i lässt sich am einfachsten als **Maximum Likelihood** gemäß Gleichung 2.1 wie im folgenden Trigramm-Beispiel schätzen:

$$\begin{aligned} P(w_i|w_{i-2}w_{i-1}) &= \frac{\#(w_{i-2}w_{i-1}w_i)}{\sum_x \#(w_{i-2}w_{i-1}w_x)} \\ &= \frac{\#(w_{i-2}w_{i-1}w_i)}{\#(w_{i-2}w_{i-1})} \end{aligned} \tag{5.3}$$

x läuft dabei über alle Worttypes. Die Anzahl der Trigramme, die mit der Wortvorgeschichte $w_{i-2}w_{i-1}$ beginnen, ist gleich der Vorkommenshäufigkeit dieser Vorgeschichte, wodurch die einfachere Berechnung in Gleichung 5.3 möglich wird.

Kapitel 6

Smoothing und Interpolation von Sprachmodellen

Aufgrund der Beschränktheit der Trainingsdaten stößt man bei der Anwendung eines Sprachmodells mit großer Sicherheit früher oder später auf ein ungesehenes N-Gramm. Mit einem nutzbringenden Modell sollten auch solche Wortfolgen erkannt werden, die nicht aus den Trainingsdaten bekannt sind.

Eine simple Maximum-Likelihood-Schätzung von Wahrscheinlichkeiten wie in Gleichung 5.3 bringt aber das Problem mit sich, dass für im Trainingskorpus ungesehene N-Gramme keine Wahrscheinlichkeitsmasse mehr zur Verfügung steht. Die Wahrscheinlichkeit einer Wortfolge, die ein solches N-Gramm enthält, wäre gemäß der Kettenregel gleich 0, beziehungsweise ergäbe das Logarithmieren der 0-Wahrscheinlichkeit $-\infty$.

Lösen lässt sich dieses Problem der **Überadaption** auf die Trainingsdaten auf zweierlei (auch kombinierbare) Weisen:

1. Ziehe von der Wahrscheinlichkeitsmasse der beobachteten Ereignisse etwas ab und stelle es für die ungesehenen Ereignisse bereit. Auf diese Weise wird eine Glättung der Wahrscheinlichkeitsfunktion dahingehend erreicht, dass sie nicht mehr so steil zu ungesehenen Ereignissen hin abfällt. Diese Glättung wird als **Smoothing** bezeichnet.

2. Kombiniere N-Gramm-Modelle höherer Ordnung mit Modellen niedrigerer Ordnung, also mit kürzerer Wortvorgeschichte, die eher in den Trainingsdaten beobachtet wurden. Methoden hierzu sind das **Backoff**-Verfahren und die **Lineare Interpolation**.

6.1 Smoothing

6.1.1 Absolutes Discounting

Beim Absoluten Discounting wird von der Häufigkeit jedes beobachteten N-Gramms ein konstanter sehr kleiner Absolutwert abgezogen, der Discount. Die Wahrscheinlichkeiten der N-Gramme werden dann auf Grundlage der reduzierten Häufigkeiten ermittelt. Die Summe der Discounts wird bei der Anwendung unter den ungesehenen N-Grammen als Häufigkeitswerte verteilt.

Beispiel In den Trainingsdaten kommen 10000 N-Gramm-Tokens und 1000 N-Gramm-Types vor. Von jeder der beobachteten Häufigkeiten wird der Discount-Wert 0.1 abgezogen. Wir erhalten eine Discount-Summe von $1000 \cdot 0.1 = 100$. Die Wahrscheinlichkeit eines dreimal beobachteten N-Gramms beträgt nun $\frac{2.9}{10000}$. In den Testdaten treten 200 im Training ungesehene N-Gramme auf. Auf diese N-Gramme wird die Discount-Summe gleichmäßig verteilt, jedem wird also die Häufigkeit $\frac{100}{200} = 0.5$ zugeordnet. Daraus ergibt sich eine Wahrscheinlichkeit von $\frac{0.5}{10000}$.

6.1.2 Good-Turing Discounting

Grundidee dieses Discounting-Verfahrens ist es, von der Wahrscheinlichkeitsmasse häufigerer Ereignisse etwas abzugeben für seltenere und ungesehene Ereignisse. Die Häufigkeiten c werden hier folgendermaßen angepasst:

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{6.1}$$

N_c steht hier für die Anzahl der N-Gramm-Types, die c Mal auftreten. In einem Bigramm-Modell wäre also N_3 die Anzahl der Bigramm-Types, die dreimal in den Trainingsdaten aufgetreten sind.

In der Regel wird eine Obergrenze k (z.B. 5) eingeführt, bis zu der Häufigkeiten c modifiziert werden. Die Anpassungsformel für Häufigkeiten ist dann folgendermaßen zu korrigieren:

$$\begin{aligned} c > k: \quad c^* &= c \\ \text{else}: \quad c^* &= \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}} \end{aligned} \tag{6.2}$$

Die neu geschätzte Häufigkeit für ungesehene Ereignisse 0^* lautet:

$$0^* = \frac{1^*}{N} \tag{6.3}$$

Die Schätzung der N-Gramm-Wahrscheinlichkeiten anhand der neu ermittelten Häufigkeiten c^* ergibt sich schließlich wie folgt (Beispiel: Bigramm):

$$P(w_i|w_{i-1}) \quad = \quad \frac{c^*(w_{i-1}w_i)}{c(w_{i-1})}, \tag{6.4}$$

und ist somit ein wenig niedriger als die uns bisher bekannte Maximum-Likelihood-Schätzung.

Bemerkung Dieses Verfahren setzt voraus, dass für die Häufigkeiten annähernd eine Zipf'sche-Verteilung gilt (vgl. Abbildung 4.1), und dass alle Häufigkeiten c von 1 bis k N_c größer 0 sind. Sollte letzteres nicht zutreffen, lassen sich fehlende N_c-Werte mittels linearer Regression wie folgt schätzen (Gale, 1994):

$$\log(N_c) \quad = \quad a + b \cdot \log(c) \tag{6.5}$$

6.2 Backoff

Backoff-Modelle kombinieren N-Gramm-Modelle verschiedener Ordnung. Im Falle eines ungesehenen **N**–Gramms ziehen Backoff-Modelle die Wahrscheinlichkeit des **(N-1)**–Gramms heran bis hin zu einem Default-Wert, sollte auch das Unigramm nicht in den Trainingsdaten vorgekommen sein. Die dem N-Gramm zugewiesene Wahrscheinlichkeit P_{bo} wird rekursiv folgendermaßen ermittelt:

$$P_{bo}(w_i|w_{i-h},\ldots,w_{i-1}) = \begin{cases} \frac{\#_{d_h}(w_{i-h},\ldots,w_i)}{\#(w_{i-h},\ldots,w_{i-1})} & : \quad \#(w_{i-h},\ldots,w_i) > 0 \\ \alpha_h P_{bo}(w_i|w_{i-h+1},\ldots,w_{i-1}) & : \quad \text{sonst} \end{cases} \tag{6.6}$$

$\#_{d_h}(s)$ ist die Häufigkeit der Sequenz s nach Abzug eines Discountwerts d_h, der von der Länge h der Vorgeschichte abhängt. α_h ist ein Normalisierungsfaktor, der ebenfalls von h abhängt und sicherstellt, dass nur jeweils die Wahrscheinlichkeitsmasse, die nach dem Discounting frei geworden ist, über die N-Gramme niedrigerer Ordnung verteilt wird.

Ohne diese Normalisierung würde folgendes passieren: Wenn durch Backoff ein ungesehenes Trigramm TG1 zu einem gesehenen Bigramm BG1 reduziert wird, das im Training häufiger beobachtet wurde als ein gesehenes Trigramm TG2, dann würde auch dem ungesehenen TG1 eine höhere Wahrscheinlichkeit zukommen als dem gesehenen TG2. Das ist natürlich nicht gewollt. Anstelle der Wahrscheinlichkeitsmasse 1 wird nach dem Backoff für die Bigramme also nur noch die Gesamtwahrscheinlichkeit $\alpha_h = \frac{\sum d_h}{N}$ bereitgestellt (d_h = Discount bei Trigrammhäufigkeiten, N = Textlänge).

6.3 Lineare Interpolation

Unter linearer Interpolation versteht man im NLP die gewichtete Kombination verschiedener statistischer Modelle. Im Gegensatz zum Backoff-Verfahren kommen die interpolierten Modelle gleichzeitig und nicht nacheinander zum Einsatz.

6.3.1 Mit konstanten Gewichten

Die Wahrscheinlichkeit einer Wortfolge $w_{1,n}$ lässt sich mit linear interpolierten Modellen folgendermaßen schätzen:

$$P(w_{1,n}) = \prod_{i=1}^{n} P_{hybrid}(w_i) = \prod_{i=1}^{n} \sum_{j} \lambda_j P_j(w_i) \tag{6.7}$$

Die geschätzte Likelihood der Wortfolge setzt sich zusammen aus dem Produkt der Wahrscheinlichkeiten der betrachteten Einzelereignisse (Auftreten von Wort w_i). Zur Ermittlung der Wahrscheinlichkeiten für w_i kann man verschiedene Modelle heranziehen (beispielsweise mit variierender Länge der Wortvorgeschichte) und die durch sie vorhergesagten Wahrscheinlichkeiten additiv zu P_{hybrid} kombinieren. $P_j(w_i)$ ist hierbei die Wahrscheinlichkeit von w_i im Modell M_j. Sei M_1 ein Unigramm-, M_2 ein Bigramm- und M_3 ein Trigramm-Modell zur Vorhersage der Wahrscheinlichkeit von w_i, dann ergibt sich oben eingesetzt:

$$P_{\text{hybrid}}(w_i) = \lambda_1 P_1(w_i) + \lambda_2 P_2(w_i|w_{i-1}) + \lambda_3 P_3(w_i|w_{i-2}w_{i-1}) \tag{6.8}$$

Die Gewichte λ_j werden mit Hilfe des **EM-Algorithmus** ermittelt. Hierbei gelten die folgenden beiden Randbedingungen: $\sum_j \lambda_j = 1$ und $0 \leq \lambda_j \leq 1$.

6.3.2 Der EM-Algorithmus

Der EM-Algorithmus stellt einen allgemeinen Rahmen für diverse NLP-Algorithmen bereit, beispielsweise in Form des **Forward-Backward-Algorithmus** für das Berechnen von Wahrscheinlichkeiten in einem Hidden--Markov-Modell (vgl. Kapitel 10), in Form des **Inside-Outside-Algorithmus** für das Berechnen von Wahrscheinlichkeiten einer Probabilistischen Kontextfreien Grammatik und eben auch zur Schätzung der Interpolationskoeffizienten. Das EM im Namen steht für **Expectation** und **Maximization**. Abstrakt formuliert funktioniert dieses Verfahren wie in Abbildung 6.1 beschrieben.

L($data$;$\{\lambda_j^{[k]}\}$) bezeichnet die Wahrscheinlichkeit (*Likelihood*) der Daten *data* gegeben die Parameterbelegung $\{\lambda_j^{[k]}\}$). Das Modell ist so zu justieren,

definiere die relevanten Statistiken zur Parameterschätzung
initialisiere Parameter $\{\lambda_j^{[0]}\}$
iteriere (über k)

E-Schritt: berechne die relevanten Statistiken anhand der Daten *data* und der Parameterwerte $\{\lambda_j^{[k]}\}$

M-Schritt: ermittle $\{\lambda_j^{[k+1]}\}$ durch Maximum-Likelihood-Schätzung basierend auf den im E-Schritt ermittelten relevanten Statistiken

terminiere, wenn $\mathrm{L}(data;\{\lambda_j^{[k+1]}\}) \approx \mathrm{L}(data;\{\lambda_j^{[k]}\})$

Abbildung 6.1: EM-Algorithmus in allgemeiner Form.

dass die Likelihood der Daten möglichst hoch wird, denn: je höher die Likelihood, desto besser passt das Modell zu den Daten. Der EM-Algorithmus garantiert die Konvergenz in einem lokalen Optimum, die Position dieses Optimums ist durch die Initialisierung der Parameter beeinflussbar.

Im Fall der lineraren Interpolation bilden die λs die Parameter zur Modellanpassung. Sie stellen die Wahrscheinlichkeiten $P(M_j|w_{1,n})$ dar, mit denen die entsprechenden Modelle M_j bei gegebenen Daten $w_{1,n}$ zum Einsatz kommen.

$$\lambda_j \quad = \quad P(M_j|w_{1,n})$$

Die Generierung der Wortfolge $w_{1,n}$ ist als Zufallsprozess zu verstehen, dem mehrere Wahrscheinlichkeitsmodelle M_j zugrundeliegen. Es ist aber nicht bekannt, wann (also für welches Wort) welches Modell zum Einsatz kommt. Zu beobachten ist also nur die Wortfolge, nicht aber ihr Zustandekommen. Bezogen auf das *Noisy-Channel-Paradigma* aus Abschnitt 3.8 lässt sich die Folge $w_{1,n}$ am Kanalausgang beobachten, von der aus wir auf die unbekannte Kanaleingabe, das Wirken der Modelle M_j, rückschließen müssen.

Die Anwendung des EM-Algorithmus auf die Ermittlung der λ's zeigt nun Abbildung 6.2.

Relevante Statistik Die relevante Statistik zur Ermittlung der Parameter λ_j ist die erwartete Häufigkeit, mit der Modell M_j für die beobachteten Daten $w_{1,n}$ zu wählen ist: $E(M_j|w_{1,n})$.

Initialisierung Ist vorab kein Wissen darüber verfügbar, ob eines der Modelle besser auf die Daten passt als ein anderes, empfiehlt es sich, die Modelle initial gleich zu gewichten, also alle λs auf den gleichen Wert zu setzen. Bei der linearen Interpolation eines Trigramm-, Bigramm und Unigramm--Modells wäre dieser Wert $\frac{1}{3}$.

definiere relevante Statistik für $\lambda_j := E(M_j|w_{1,n})$ (erwartete Häufigkeit, mit der Modell M_j gewählt wird.)
initialisiere λ_j für Modell M_j, $1 \leq j \leq m$ bei m Modellen; z.B. auf $\frac{1}{m}$
iteriere (über k)

E-Schritt:

$$\begin{aligned} E_{\lambda^{[k]}}(M_j|w_{1,n}) &= \sum_{i=1}^{n} P_{\lambda^{[k]}}(M_j|w_i) \\ &= \sum_{i=1}^{n} \frac{\lambda_j^{[k]} P_j(w_i)}{\sum_o \lambda_o^{[k]} P_o(w_i)} \end{aligned}$$

M-Schritt:

$$\lambda_j^{[k+1]} = \frac{E_{\lambda^{[k]}}(M_j|w_{1,n})}{n}$$

terminiere, wenn $\mathrm{L}(data;\{\lambda_j^{[k+1]}\}) \approx \mathrm{L}(data;\{\lambda_j^{[k]}\})$

Abbildung 6.2: Schätzen der Werte der Interpolationskoeffizienten.

Iteration Solange sich die Likelihood der Trainingsdaten anhand des linearen Interpolationsmodells mit den Parametern $\{\lambda_j\}$ einigermaßen deutlich erhöht, werden mit Hilfe des E- und M-Schritts die λs weiter modifiziert. Unbedeutende Erhöhungen der Likelihood sind eher mit einer Überadaption des Modells an die Daten verbunden, hier sollte die Iteration dann abgebrochen werden.

E-Schritt Hier sind die erwarteten Einsatzhäufigkeiten $E(M_j|w_{1,n})$ für alle Modelle zu schätzen. Je höher die Likelihood, die Modell M_j den Daten zukommen lässt, desto höher ist seine zu erwartetende Einsatzhäufigkeit bei der Generierung dieser Daten. Anders formuliert: wäre es seltener zum Einsatz gekommen, hätten wir andere Daten erhalten. Die Einsatzhäufigkeit von Modell M_j ergibt sich durch die Summe seiner gewichteten Einsatzwahrscheinlichkeiten über die einzelnen Tokens $\sum_{i=1}^{n} P_{\lambda^{[k]}}(M_j|w_i)$. Die Einsatzwahrscheinlichkeit für M_j gegeben Token w_i ist der Anteil der gewichteten w_i-Wahrscheinlichkeit in Modell M_j an der w_i-Wahrscheinlichkeit im Interpolationsmodell bei der im aktuellen Iterationsschritt k gegebenen λ-Belegung: $\frac{\lambda_j^{[k]} P_j(w_i)}{\sum_o \lambda_o^{[k]} P_o(w_i)}$. Je höher die w_i-Wahrscheinlichkeit im Modell M_j, desto größer der Wert dieses Bruchs und desto höher damit die zu erwartende Einsatzhäufigkeit von M_j.

M-Schritt Zur Neuschätzung der λs werden die zu erwartenden Einsatzhäufigkeiten der Modelle jeweils durch die Anzahl der Trainings-Tokens geteilt: $\frac{E_{\lambda^{[k]}}(M_j|w_{1,n})}{n}$. Dadurch erhält man für jedes Modell M_j die Maximum-

Likelihood-Schätzung seiner Einsatzwahrscheinlichkeit in den gegebenen Daten $P(M_j|w_{1,n})$ und damit sein Gewicht λ_j.

6.3.3 Mit variablen Gewichten

Ermittelt man wie soeben besprochen für die λ's konstante Werte, so stößt man auf das Problem, dass häufigen und damit reliablen Wortvorgeschichten dasselbe Gewicht zukommt wie seltenen gleicher Länge. Wünschenswert ist allerdings eine höhere Gewichtung der reliableren Vorgeschichten. Dies lässt sich erreichen, indem man die λ's allgemein als Funktion der History h bestimmt:

$$P_{\text{hybrid}}(w|h) = \sum_j \lambda_j(h) P_j(w|h) \tag{6.9}$$

Konkret beziehen sich die λ's auf die Historyhäufigkeiten. Hierzu wendet man den in Abbildung 6.2 beschriebenen Algorithmus zur Ermittlung der Gewichte für Histories unterschiedlicher Häufigkeit getrennt an. Bei einem komplexen Trigramm-Modell (mit λ's für Tri-, Bi- und Unigramme) erhielte man somit o Koeffizientensätze in Abhängigkeit der o beobachteten unterschiedlichen Trigrammhistory-Häufigkeiten. Für Trigrammhistories, die 9-mal auftreten, erhält man beispielsweise den Satz: $[\lambda_{\text{tg}}(9), \lambda_{\text{bg}}(9), \lambda_{\text{ug}}(9)]$ (tg, bg, ug = Trigramm, Bigramm, Unigramm).

Kapitel 7

Klassenbasierte Modelle

7.1 Das Sprachmodell

Häufig lassen sich einfache N-Gramm-Modelle verbessern, wenn man Types zu Klassen zusammenfasst. Der Reiz solcher Modelle besteht darin, dass durch diese Zusammenfassung die Häufigkeiten höhere Werte annehmen und somit verläßlichere Schätzungen der Wahrscheinlichkeiten ermöglichen. Ein Klassenbasiertes N-Gramm-Modell hat folgende Wahrscheinlichkeit der Wortfolge $w_{1,n}$ zu berechnen:

$$P(w_{1,n}) = \prod_{i=1}^{n} P(c_i|c_{\text{hist}_i})P(w_i|c_i) \tag{7.1}$$

c_i steht hier für die Klasse des Types w_i, c_{hist_i} für die Klassen der Types der History.

Einfache N-Gramm-Modelle lassen sich auch mit klassenbasierten Modellen mittels linearer Interpolation (vgl. Kapitel 6.3) kombinieren.

Zur Aufteilung eines Vokabulars in Wortklassen bedarf es statistischer Klassifikatoren, die in den folgenden Abschnitten vorgestellt werden.

7.2 Statistische Klassifikatoren

7.2.1 Grundlegendes

Definition Statistische Klassifikatoren sind Instrumente **maschinellen Lernens**. Ziel ist es, den Zusammenhang zwischen Zielwerten für Objekte (Kategorien oder kontinuierliche Werte) und deren Eigenschaften zu erlernen.

Beispiele Bei der **Graphem-Phonem-Konvertierung** wird für die Graphem-Objekte anhand ihrer Attribute wie Graphemidentität, umgebende Grapheme, Position innerhalb der Silbe usw., der zugehörige Phonem-Zielwert vorhergesagt.

Bei der **Textklassifizierung** wird für Textobjekte anhand der Wortfolgen, die darin vorkommen, vorhergesagt, um welche Textsorte (z.B. Fachtext für Schnellboote) es sich handelt.

Die **Sprachidentifizierung** von Wortobjekten operiert auf Grundlage der enthaltenen Buchstabenfolgen.

Die Klassifizierung von Wörtern anhand ihrer Distribution in Texten ist eine Vorarbeit für die in diesem Kapitel behandelten **Klassenbasierte Sprachmodelle**.

Training, Anwendung Im **Training** wird der Zusammenhang zwischen einer oder mehreren **unabhängigen Variablen** (z.B. Graphemeigenschaften) und einer **abhängigen Variablen** (Phonem) erlernt. Im Falle von Klassifikatoren ist die abhängige Variable **kategorial**, d.h. ihr Wertebereich lässt sich mit einer **endlichen** Menge (z.B. Phoneme des Deutschen) beschreiben (im Gegensatz zu kontinuierlich, vgl. Abschnitt 2).

Bei der **Anwendung** wird der (unbekannte) Wert der abhängigen Variablen anhand der gegebenen unabhängigen Variablen vorhergesagt.

Objektrepräsentation Objekte werden als **Merkmalsvektoren** (**Featurevektoren**) repräsentiert, der ihre Eigenschaften (die Werte der unabhängigen Variablen) beinhaltet. Diese Variablenwerte können **kategorial** (z.B. Graphem-Identität) oder **kontinuierlich** (z.B. Wahrscheinlichkeit, dass Text x zu Klasse y gehört) sein.

7.2.2 Gängige Klassifikatoren

Es ist zu unterscheiden zwischen **überwachtem** und **unüberwachtem Lernen**. Beim überwachten Lernen sind die Werte der abhängigen Variable in den Trainingsdaten bekannt, beim unüberwachten Lernen ergeben sich die Kategorien durch Zusammenfassen von Objekten in Klassen anhand ihrer Ähnlichkeit.

Hinsichtlich dieser Lernkategorien lassen sich gängige Klassifikatoren folgendermaßen einteilen:

- **überwacht**: C4.5-Entscheidungsbäume, CART, Neuronale Netze, Baye'sche Klassifikatoren
- **unüberwacht**: Clustering, Neuronale Netze

Hinsichtlich der erlaubten Typen an unabhängigen Variablen sind diese Klassifikatoren folgendermaßen zu charakterisieren:

- **C4.5:** kategorial, kontinuierlich
- **CART:** kategorial, kontinuierlich

- **Neuronale Netze:** kontinuierlich, kategorial (binärkodiert)
- **Clustering:** kontinuierlich, kategorial (binärkodiert)
- **Baye'sche Klassifikatoren:** kontinuierlich (Wahrscheinlichkeiten)

Von den erwähnten Klassifikatoren soll nun das Clustering ein wenig genauer behandelt werden. Baye'sche Klassifikatoren werden in Kapitel 12 vorgestellt. Detailliertere Beschreibungen diverser maschineller Lernverfahren sind beispielsweise hier zu finden:
www.phonetik.uni-muenchen.de/~reichelu/kurse/machine_learning/machine_learning.html.

7.2.3 Clustering

Clustering bezeichnet die automatische Partitionierung einer Menge von Objekten anhand ihrer Ähnlichkeit. Die Kategorien sind initial unbekannt und ergeben sich erst durch den Clustering-Prozess. **Hartes** Clustering weist die Objekte genau einem Cluster zu, nämlich dem ähnlichsten, **weiches** Clustering dagegen weist die Objekte allen Clustern zu, deren Ähnlichkeit einen festgelegten Schwellwert überschreitet.

Nicht-hierarchisches Clustering

Beim nicht-hierarchischen (flachen) Clustering werden entstandene Cluster nicht weiter zusammengefasst. Ein Standardverfahren für nicht-hierarchische, hartes Clustern ist die der sogenannte **Kmeans**-Algorithmus.

$X \leftarrow$ Menge der zu clusternden Objekte
$k \leftarrow$ Anzahl der zu gewinnenden Cluster
init: bestimme k Clusterzentren
until alle Cluster stabil
    foreach $x \in X$
        $c \leftarrow$ ähnlichstes Cluster
        $c \leftarrow [c, x]$
        update Clusterzentrum(c)
    endforeach
enduntil

Abbildung 7.1: Kmeans.

Zu Beginn werden k Clusterzentren festgelegt. Die Objekte x werden nun nacheinander dem ähnlichsten Cluster zugewiesen, wobei nach jeder Zuweisung die Repräsentation des gewählten Clusters aktualisiert wird. Der Durchlauf durch die Objektmenge wird solange wiederholt, bis ein Abbruchkriterium erfüllt ist.

Stellen sich nun folgende Fragen:

1. Wie wird ein Objekt mit einem Cluster verglichen?
2. Wie wird Ähnlichkeit gemessen?
3. Wie legt man zu Beginn Clusterzentren fest?
4. Wie sieht das Abbruchkriterium aus?

Objekt- und Clusterrepräsentation

Objekte x und Cluster c sind in Form von numerischen Vektoren gegeben, wobei die Vektorlänge (die **Dimensionalität**) gleich der Anzahl der unabhängigen Variablen ist, durch die Objekte beschrieben werden.

Cluster lassen sich als Menge von Objekten und somit als Menge der zugehörigen Vektoren verstehen. Um ein Objekt nun mit einem Cluster vergleichen zu können, wird bei **Kmeans** der **Zentroid** des Clusters berechnet. Darunter versteht man den Mittelwertsvektor aller enthaltenen Vektoren. Verglichen wird dann der Objektvektor mit dem Cluster-Zentroiden.

Rechenaufwendigere alternative Clusterrepräsentationen sind der dem Objekt ähnlichste Vektor im Cluster (**single-link clustering**) oder der unähnlichste (**complete-link clustering**). **Single-link clustering** führt zu gedehnten, **complete-link clustering** zu kompakten Clustern.

Abstandsmaße

Zur Ermittlung der Ähnlichkeit zwischen den Vektoren $\vec{x}$ und $\vec{c}$ werden **Abstandsmaße** (oder **Ähnlichkeitsmaße**) herangezogen, wie beispielsweise nach **Euklid** $\sqrt{\sum_i (x_i - c_i)^2}$, Kosinus, Dice, City Block, Jaccard usw. Näheres zu den Eigenschaften von Abstandsmaßen findet sich in Abschnitt 7.3.

Vorabermittlung der Clusterzentren

Clusterzentren können zufällig oder systematisch beispielsweise mit Hilfe des sogenannten **Mountain-** oder des **subtraktiven Clusterns** ermittelt werden. Grob gesagt werden bei diesen Verfahren iterativ im Merkmalsraum Regionen mit hoher Objektdichte gesucht, die dann als Clusterzentren festgelegt werden.

Abbruchkriterium

Die erzielte Partitionierung ist stark abhängig von der Zuordnungsreihenfolge der Objekte. Dem kann durch wiederholtes **Reallozieren** begegnet werden. Hierbei wird nach jedem Iterationsschritt für jedes Objekt x erneut das ähnlichste der Cluster ermittelt und x entsprechend verschoben, falls sein aktuelles Cluster nicht mehr das ähnlichste sein sollte. Der Clustervorgang ist beendet, wenn sich keines der Objekte mehr auf diese Weise verschieben lässt.

Hierarchisches Clustering

Beim hierarchischen Clustering werden im **Bottom-Up-Verfahren** jeweils Paare von Objekten bzw. Clustern verbunden, bis alle Objekte in einem Cluster (auf der obersten Hierarchieebene) enthalten sind.

Klassifizierung

Zur Klassifizierung neuer Objekte wird das ähnlichste Cluster ermittelt (gemäß der Wahl der Clusterrepräsentation) und die entsprechende Kategorie zugewiesen.

7.3 Kategorisierung der Wörter

7.3.1 Clustering

Da die Wortkategorien anfangs nicht bekannt sind, ist zu ihrer Gewinnung eine Methode des unüberwachten Lernens wie beispielsweise das soeben besprochene Clusteringverfahren zu wählen.

Grundidee ist, dass sich die Ähnlichkeit der Wörter anhand der Ähnlichkeit ihrer Distribution in Texten ergibt. Wörter die im selben Wortkontext auftreten gehören demnach eher zur selben Klasse als Wörter, die keine gemeinsamen Kontexte haben. Hierbei kann jedes Worttype w_i als zu clusterndes Objekt durch folgenden Merkmalsvektor $f(w_i)$ beschrieben werden: $f(w_i) = ([P(w_i|w_1), P(w_i|w_2), \ldots P(w_i|w_n)]$ (bei insgesamt n betrachteten Types). Es werden für w_i also die Auftretenswahrscheinlichkeiten im Kontext der anderen Types gesammelt. Diese bedingten Wahrscheinlichkeiten können beispielsweise in Form von Bigrammwahrscheinlichkeiten oder Zählung von Wortkookkurenzen in Textfenstern bestimmter Länge ermittelt werden.

Verwendet man nun ein Distanzmaß wie den oben vorgestellten Euklid'schen Abstand d, dann ist er für zwei Types w_i und w_j dann groß, wenn sich ihre Auftretenskontexte und damit die Wahrscheinlichkeiten $P(w_i|w_x)$ und $P(w_j|w_x)$ (x über alle Types) stark unterscheiden. Anhand dieser Information ist nun ein Clustering der Types wie im vorangegangenen Kapitel besprochen möglich.

7.3.2 Informationsradius

Zieht man zur Charakterisierung der Worttypes die Auftretenswahrscheinlichkeiten im Kontext aller im Trainingsmaterial vorkommenden Types heran, ergeben sich sehr lange Merkmalsvektoren, was das Clusteringverfahren sehr rechenaufwendig und langsam macht. Weniger aufwendig ist die Arbeit mit dem in diesem Abschnitt vorgestellten Maß, dem **Informationsradius**. Auch hier ergibt sich die Ähnlichkeit zweier Types über ihre Distribution:

zwei Types sind umso ähnlicher, je ähnlicher die Kontexte sind, in denen sie auftreten.

Eine Möglichkeit, die Distribution zweier Types zu vergleichen, bietet die **Relative Entropie** (vgl. Abschnitt 3.4):

$$D(p||q) = \sum_i p_i \log_2 \frac{p_i}{q_i} \tag{7.2}$$

p bezeichnet die Wahrscheinlichkeitsverteilung für Type w_p in den betrachteten Kontexten, und q die Verteilung für Type w_q. Als Kontexte können beispielsweise Bigramme gewählt werden, in diesem Fall würde für alle Types $\{w_i\}$ in den Trainingsdaten die bedingten Wahrscheinlichkeiten $p_i = P(w_p|w_i)$ und $q_i = P(w_q|w_i)$ ermittelt und über Gleichung 7.2 zueinander ins Verhältnis gesetzt.[1]

Je kleiner $D(p||q)$ desto ähnlicher die Distribution von w_p und w_q, was eine Zusammenfassung der Types zur selben Kategorie nahelegt.

Unschön ist allerdings, dass die Relative Entropie kein **Abstands-** sondern nur ein **Divergenzmaß** ist.

Abstandsmaße müssen folgende Bedingungen erfüllen:

- Positivität, Definitheit: d(x,y) $\geq$ 0; d(x,y)=0 genau dann wenn x=y
- Symmetrie: d(x,y)=d(y,x)
- Dreiecksungleichung: d(x,y) $\leq$ d(x,z)+d(z,y)

Die relative Entropie erfüllt zwar die Bedingungen Positivität und Definitheit, nicht aber die der Symmetrie ($D(p||q) \neq D(q||p)$) und der Dreiecksungleichung. Vor allem die fehlende Symmetrieeigenschaf widerspricht der Intuition: warum sollte Type sich w_i von w_j nicht gleichermaßen unterscheiden wie w_j von w_i?

Dem kann Abhilfe geschafft werden, da sich aus der Relativen Entropie ein echtes Abstandsmaß ableiten lässt: der **Informationsradius IRad**:

$$\text{IRad} = D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2}) \tag{7.3}$$

IRad gibt die Information an, die verlorengeht, wenn wir zwei Ereignisse (also w_p und w_q) mit ihrer durchschnittlichen Wahrscheinlichkeitsverteilung beschreiben, also für jede Kontextwahrscheinlichkeit den Mittelwert bilden. Auch hier gilt: je kleiner der Wert, desto ähnlicher die Verteilungen von w_p und w_q, und desto eher sind sie der selben Kategorie zuzuschlagen.

[1] Vorausgesetzt ist hierbei ein Smoothing der Wahrscheinlichkeitsfunktionen.

7.4 Linguistischer Bezug

Mit Hilfe der im vorangehenden Abschnitt beschriebene Wortklassifizierung lassen sich **paradigmatische** Beziehungen zwischen Wörtern untersuchen, also die Frage, welche Wörter einander im gleichen Kontext ersetzen können. Kandidaten hierfür sind beispielsweise **Kohyponyme**, also Wörter, die einen gemeinsamen Oberbegriff (Hyperonym genannt) besitzen. Wochentagbezeichnungen 'Montag', 'Dienstag', ... sind beispielsweise solche Kohyponyme, die sich das Hyperonym 'Wochentag' teilen.

Das hier beschriebene Verfahren ist zugleich eine Umsetzung der **Distributionsanalyse** im **Amerikanischen Strukturalismus** (vgl. z.B. Harris, 1951), die der Gewinnung linguistischer Einheiten dient.

Kapitel 8

Evaluierung von Sprachmodellen

8.1 Gütemaße

Sprachmodelle werden wie in Kapitel 4 bereits dargelegt anhand eines von den Trainingsdaten unabhängigen Testkorpus evaluiert. Evaluierungsmaße sind wie in Kapitel 3 beschrieben die **Kreuzentropie** und die **Perplexität**.

Zur Evaluierung eines N-Gramm-Modells M wird die Kreuzentropie für Testdaten $w_1, \ldots, w_n$ durch Einsetzen der Wahrscheinlichkeit m berechnet, die Modell M den Testdaten verleiht.

$$H(L, m) \approx -\frac{1}{n} \log_2 m(w_{1,n}) \tag{8.1}$$

Je höher die Wahrscheinlichkeit m, desto besser passt das Modell zu den Testdaten und desto besser ist es zur Wortvorhersage geeignet, was sich wiederum in einem geringeren Entropiewert (und einer geringeren Perplexität) widerspiegelt.

8.2 Vergleichende Evaluierung I

Möchte man zwei Modelle hinsichtlich ihrer Güte vergleichend evaluieren, genügt es nicht, einfach 2 Entropiewerte miteinander zu vergleichen, da diese auch rein zufällig verschieden groß ausfallen können.

Was wir benötigen, sind zwei Stichproben von Entropiewerten, eine für jedes Modell, die wir miteinander vergleichen können.

Hierfür unterteilt man das Testkorpus zufallsgeleitet in mehrere kleine Subkorpora und evaluiert jedes der Modelle auf jedem dieser Subkorpora. Stehen nicht ausreichend Testdaten zur Verfügung, lässt sich der gleiche Effekt mittels Kreuzvalidierung (vgl. Abschnitt 4.5) herstellen.

Auf Grundlage dieser Stichproben ist es nun möglich, zu testen, ob die Performanzmittelwerte signifikant (also nicht nur zufällig) unterschiedlich voneinander sind, ob also ein Modell wirklich besser ist als das andere. Die hierzu nötigen Testverfahren werden in den nächsten Abschnitten besprochen.

8.3 Statistisches Testen

8.3.1 Einige Begriffe

Stichprobe

Unter einer **Stichprobe** wird eine Menge von Messwerten verstanden, beispielsweise Entropiewerte eines Sprachmodells. Hierfür muss man das Modell auf mehrere verschiedene Datensätze anwenden. Beim Vergleich zweier Modelle $M1$ und $M2$ erhält man somit zwei Stichproben.

Man spricht von **abhängigen Stichproben**, wenn die beiden Sprachmodelle auf dieselben Datensätze angewendet worden sind. Auf diese Weise erhält man **Messwertpaare**, also jeden der Datensätze jeweils von $M1$ und $M2$ modelliert.

Unterscheiden sich die Datensätze, handelt es sich um **unabhängigen Stichproben**.

Skalenniveau

Zu unterscheiden sind **nominal-, ordinal und kardinalskalierte** Variablen.

Nominal Nominalskalierte Variablen können als Werte nur diskrete Kategorien annehmen (beispielsweise die Variable *Wortart*). Ein Spezialfall sind **dichotome** oder (**binäre**) Variablen, die genau zwei Werte annehmen können (z.B. *Wortart*: "+/– Verb"). Die Werte dieses Variablentyps lassen sich nicht ordnen.

Ordinal Die Werte von ordinalskalierten Variablen sind ebenfalls kategorial, lassen sich aber ordnen (z.B. *Altersstufen*: "jung", "mittelalt", "alt"; *Qualität*: "schlecht", "mittel", "gut").

Kardinal Kardinalskalierte (**metrische**) Variablen nehmen ganz- oder reellzahlige Werte an (z.B. *Alter, Entropie*). Diese Werte können nicht nur geordnet, sondern auch ihr Unterschied quantitativ erfasst werden.

Hypothesen, ein- vs. zweiseitige Tests

Statistische Tests dienen dazu, zu prüfen, ob eine **Nullhypothese** zugunsten einer **Alternativhypothese**, die das Gegenteil besagt, verworfen oder beibehalten werden muss.

Beispiel: Der Performanzmittelwert von Modell $M1$ sei μ_1, der von $M2$ μ_2. Bei einem **zweiseitigen Test** lautet die Nullhypothese $\mu_1 = \mu_2$, d.h. $M1$ und $M2$ unterscheiden sich nicht signifikant hinsichtlich ihrer Performanz. Bei einem **einseitigen Test** lautet die Nullhypothese $\mu_1 \leq \mu_2$, Die Performanz von $M1$ ist also schlechter oder gleich der von $M2$ ($\geq$ analog).

Signifikanzniveau

Das **theoretische Signifikanzniveau** α gibt die **Irrtumswahrscheinlichkeit** an, mit der man eine Nullhypothese verwirft. Für zweiseitige Tests beträgt diese Wahrscheinlichkeit beispielsweise bei $\alpha = 0.05$ 5%, bei $\alpha = 0.01$ 1%. Für einseitige Tests beträgt die Irrtumswahrscheinlichkeit 2α.

Das **empirische Signifikanzniveau**, d.h. das Niveau, das man mit der anhand der gegebenen Daten ermittelten **Testgröße** und der **Freiheitsgrade** in einer Tabelle nachschlagen (oder auch direkt berechnen) kann, muss also bei zweiseitigen Tests $\leq \alpha$ sein und bei einseitigen Tests $\leq \alpha/2$, damit die Nullhypothese zugunsten der Alternativhypothese verworfen werden kann.

Anders formuliert übersteigt die ermittelte Testgröße den Wert in der Tabelle zu zugehörigem Signifikanzniveau und Freiheitsgraden, wenn die Stichprobenmittelwerte ausreichend (signifikant) verschieden voneinander sind.

Die Anzahl der Freiheitsgrade *df* ergibt sich bei unabhängigen Stichproben im Allgemeinen über die Stichprobengrößen n_i: $\mathrm{df} = \sum_i (n_i - 1)$. Bei abhängigen Stichproben, die ja alle die gleiche Größe n haben, gilt im Allgemeinen: $\mathrm{df} = n - 1$.

8.3.2 Mittelwert-Tests

Wahl eines Tests

Für die Wahl des richtigen Tests müssen vorab folgende Sachverhalte geklärt werden:

- Anzahl der Stichproben
- Skalenniveau der betroffenen Variablen
- Größe der Stichproben, Verteilung der Werte
- Abhängigkeit der Stichproben

Methoden zum Vergleich der Mittelwerte von **mehr als 2 Stichproben** (z.B. **Varianzanalyse**), wie man sie zum Vergleich von mehr als 2 Modellen benötigt, werden im Rahmen dieses Seminars aus Platzgründen nicht besprochen. Daher ist die betrachtete Stichprobenanzahl hier stets gleich **2**.

Im folgenden werden Tests besprochen, die zur vergleichenden Evaluierung von Sprachmodellen herangezogen werden. Die zugehörige Variable *Entropie* ist **kardinalskaliert**.

Damit verbleibt vor der vergleichenden Evaluierung von Sprachmodellen folgender Entscheidungsbaum:

if |Stichprobe| ≥ 50[1] $\vee$ Normalverteilung der Werte (**K.S.-Anpassungstest**)
    if Stichproben unabhängig
        if Varianzhomogenität (**Levene-Test**)
            t-Test für unabhängige Stichproben
        else
            Welch-Test
    else
        t-Test für abhängige Stichproben
else
    if Stichproben unabhängig
        Mann-Whitney-Test
    else
        Wilcoxon-Test für Paardifferenzen

Abbildung 8.1: Wahl des passenden statistischen Tests zum Vergleich der Mittelwerte zweier Stichproben.

Der **t-Test** und der **Welch-Test** werden als **parametrische Tests** bezeichnet, da sie auf den Parametern Mittelwert und Standardabweichung operieren und damit auch eine bestimmte Verteilung der Werte verlangen. Liegt diese Verteilung nicht vor, kommen **nicht-parametrische** (**verteilungsfreie**) Tests wie der **Mann-Whitney-** und der **Wilcoxon-Test für Paardifferenzen** zum Einsatz. Parametrische Tests sind **effizienter** als nicht-parametrische. Effizienz bedeutet dabei grob die Deutlichkeit, mit der sich Mittelwerte unterscheiden müssen, damit sie als signifikant verschieden interpretiert werden können. Je effizienter ein Test, desto weniger deutlich muss der Unterschied ausfallen.

Die hier aufgelisteten Tests werden in den folgenden Abschnitten kurz vorgestellt. In den zugehörigen Formeln bezeichnen n_x, μ_x und s_x stets Größe, Mittelwert und Standardabweichung der Stichprobe x.

t-Test für unabhängige Stichproben

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}}\sqrt{\frac{n_1 n_2}{n_1+n_2}} \tag{8.2}$$

Welch-Test

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n} + \frac{s_2^2}{n}}} \tag{8.3}$$

Die Ermittlung der Anzahl der Freiheitsgrade *df* ist hier ein wenig komplizierter:

$$\mathrm{df} = \frac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{\frac{1}{n_1-1}(\frac{s_1^2}{n_1})^2 + \frac{1}{n_2-1}(\frac{s_2^2}{n_2})^2} \tag{8.4}$$

8.3.3 t-Test für abhängige Stichproben

$$t = \frac{\overline{d}}{s_d}\sqrt{n} \tag{8.5}$$

$\overline{d}$ bezeichnet das arithmetische Mittel und s_d die Standardabweichung der Messwertpaardifferenzen.

Mann-Whitney-Test (U-Test)

$$z = \frac{U - \frac{n_1 n_2}{2}}{\sqrt{\frac{n_1 n_2 (n_1+n_2+1)}{12}}} \tag{8.6}$$

$$U = \min\left[n_1 n_2 - \frac{n_1(n_1+1)}{2} - R_1, n_1 n_2 - \frac{n_2(n_2+1)}{2} - R_2\right] \tag{8.7}$$

R_1, R_2 bezeichnen die Rangsummen der jeweiligen Stichprobe.

Rangsumme fasst man die Stichprobenwerte zusammen und ordnet sie, so erhält man für jede Stichprobe die Ränge ihrer Werte. Die Summe dieser Ränge bildet die Rangsumme. Bei **Rangbindungen**, also dem Vorkommen von Rängen, die von mehreren gleichen Werten belegt werden, ist z bei den verteilungsfreien Mann-Whitney- und Wilcoxon-Tests etwas anders zu berechnen. Dies sollte bei Bedarf in einem Statistik-Buch nachgeschlagen werden.

Wilcoxon-Test für Paardifferenzen

$$z = \frac{R - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}} \tag{8.8}$$

8.3.4 Verteilungs- und Varianztests

Um den t-Test anwenden zu können, der effizienter ist als die anderen hier besprochenen Tests, müssen die Stichproben vorab jeweils daraufhin geprüft werden, ob ihre Werte normalverteilt sind. Das erledigt der **Kolmogorov-Smirnov-Anpassungstest**. Bei unabhängigen Stichproben ist zusätzlich die Varianzhomogenität der Stichproben zu prüfen. Hierfür lässt sich beispielsweise der **Levene-Test** heranziehen.

Kolmogorov-Smirnov-Anpassungstest

Dieser Anpassungstest prüft, ob eine hypothetisch erwartete theoretische Verteilungsfunktion F_X^e einer Variablen X (in unserem Fall eine Normalverteilung) der tatsächlich beobachteten Verteilung von X hinreichend ähnlich ist. Die Testgröße wird wie folgt ermittelt:

$$z = \sqrt{n} \max |F_X - F_X^e| \tag{8.9}$$

n bezeichnet die Stichprobengröße und wird multipliziert mit der maximalen Differenz zwischen tatsächlich beobachtetem Wert und dem durch die theoretische Verteilungsfunktion vorhergesagten Wert. Ab einer bestimmten Größe dieser Differenz (in Tabelle nachzuschlagen) muss die Nullhypothese verworfen werden, die beobachtete Verteilung entspricht dann nicht der theoretischen.

Levene-Test

Zur Prüfung der Varianzhomogenität werden die absoluten Abweichungen der Stichprobenwerte zu den jeweiligen Medianen[2] ermittelt und mit Hilfe der **Einfachen Varianzanalyse (ANOVA)** auf signifikante Unterschiede geprüft. Details in jedem besseren Statistikbuch.

8.3.5 Zusammenhangstests

Ein Test auf Zusammenhänge ist der sog. χ^2**-Test**. Er wird in Kapitel 9 genauer behandelt.

8.4 Vergleichenden Evaluierung II

Für zwei Sprachmodelle liegt nun durch Kreuzvalidierung jeweils eine Verteilung von Performanzwerten (in Form von Entropie-Werten) vor.

Sind die Performanzwerte normalverteilt oder übersteigt die Anzahl der Subkorpora eine gewisse Größe (≈ 50), lässt sich ein **parametrischer Test**,

[2]Median: mittlerer Wert einer Stichprobe, unter- und oberhalb dem jeweils 50 % der Werte liegen.

hier der **t-Test** heranziehen. Je nachdem, ob man für beide Modelle dieselbe Partitionierung des Korpus vorgenommen hat oder nicht, wählt man den t-Test für **abhängige** oder **unabhängige Stichproben**. Bei Varianzinhomogenität der Performanzwertverteilungen muss der t-Test für unabhängige Stichproben durch den **Welch-Test** ersetzt werden.

Für den Fall, dass die Performanzwerte nicht normalverteilt sind oder keine ausreichend hohe Anzahl an Test-Subkorpora erstellt werden kann, kommen **nicht-parametrische Tests** zum Einsatz. Bei abhängigen Stichproben wäre dies der **Wilcoxon-Test**, bei unabhängigen Stichproben der **Mann-Whitney-Test**.

Kapitel 9

Kollokationen

9.1 Aufgabenstellung

Definition Kollokationen sind konventionelle feste Verbindungen von Wörtern. Dazu zählen Redensarten und Stützverbkonstruktionen (*Kritik üben*). Sie sind durch **Semikompositionalität** gekennzeichnet, das heißt, die Bedeutung der Wortverbindung kann nicht aus den Bedeutungen ihrer Bestandteile vorhergesagt werden. Statistisch lassen sich Kollokation als überzufällig häufig miteinander auftretende Wörter definieren.

Die Aufgabe besteht also im Aufdecken von **syntagmatischen** Beziehungen zwischen Wörtern, im Gegensatz zu den in Kapitel 7 besprochen Klassenbasierten Sprachmodellen, mit Hilfe derer **paradigmatische** Beziehungen zwischen Wörtern entdeckt werden können.[1]

Nutzen der Extrahierung Die Extrahierung von Kollokationen ist in diversen NLP-Bereichen nutzbringend, beispielweise ist ihre Semikompositionalität bei der maschinellen Übersetzung zu berücksichtigen. Bei der Vorhersage der prosodischen Struktur in einer Äußerung im Rahmen der Text--to-Speech-Synthese ist die Wahrscheinlichkeit prosodischer Phrasengrenzen innerhalb von Kollokationen benachbarter Wörter stark herabgesetzt.

Zur Vereinfachung werden im Folgenden nur Zwei-Wort-Kollokationen zwischen w_i und w_j behandelt, die unten beschriebenen Verfahren lassen sich aber auf mehrere Wörter verallgemeinern, indem man w_j durch die Sequenz $\bigcap_j w_j$ ersetzt.

Es gibt eine Vielzahl statistischer Verfahren zur Extrahierung von Kollokationen, drei davon sollen in diesem Seminar vorgestellt werden: Die Extrahierung mittels χ^2-Test, mittels *Likelihood-Verhältnis* und mittels *Pointwise Mutual Information*.

[1] Syntagmatisch: welche Wörter treten gemeinsam auf? Paradigmatisch: welche Wörter können einander ersetzen?

9.2 Fensterung

Kollokationen werden nicht nur von benachbarten Wörtern gebildet ("**übt** er diesmal keine **Kritik**?"). Daher erfolgt das Zählen der Kookkurrenzen von w_i und w_j innerhalb eines Wortfensters vorab festgelegter Länge. Dieses Fenster wird wortweise über den Text geschoben, wobei w_i am rechten (oder linken) Rand des Fensters liegen muss, sonst würden wir jedes gemeinsame Auftreten von w_i und w_j doppelt zählen (warum?). Die Anzahl der Beobachtungen N entspricht dabei weiterhin der Anzahl der Tokens (der Textlänge). Auf diese Weise lassen sich im Gegensatz zu den konventionellen N-Gramm-Modellen, wie wir sie weiter oben kennengelernt haben, auch Fernabhängigkeiten (sog. **long distance dependencies**) in einem Text modellieren.

9.3 Extrahierung mittels χ^2-Test

Eine Möglichkeit der Extrahierung von Kollokationen besteht in der Anwendung des in Abschnitt 8.3.5 kurz erwähnten χ^2-Tests zur Prüfung der Abhängigkeit zweier Ereignisse w_i und w_j. Die tatsächlich beobachteten Häufigkeiten dieser Ereignisse werden verglichen mit den zu erwartenden Häufigkeiten, die mit der Nullhypothese $H0$ "w_i und w_j sind unabhängig voneinander" einhergehen.

Die **beobachteten** und bei $H0$ **zu erwartenden** Häufigkeiten bei der Textlänge N lassen sich mit Hilfe einer 2x2-Kontingenztabelle (s. Tabelle 9.1) angeben:[2]

	w_i		$\neg w_i$	
	O	E	O	E
w_j	c_{ij}	$\frac{c_i}{N}\frac{c_j}{N}N$	$c_j - c_{ij}$	$\frac{N-c_i}{N}\frac{c_j}{N}N$
$\neg w_j$	$c_i - c_{ij}$	$\frac{c_i}{N}\frac{N-c_j}{N}N$	$N - c_i - c_j + c_{ij}$	$\frac{N-c_i}{N}\frac{N-c_j}{N}N$

Tabelle 9.1: Kontingenztabelle mit den beobachteten (O) und bei Unabhängigkeit zu erwartenden (E) Häufigkeiten der Ereignisse w_i und w_j.

Die zu erwartenden Häufigkeiten ergeben sich hier einfach über die **relativen Häufigkeiten** für w_i und w_j, die bei gemeinsamem Auftreten miteinander multipliziert werden (da ja die Unabhängigkeit dieser Ereignisse angenommen wird; vgl. Abschnitt 2.4) und durch Multiplikation mit der Stichprobengöße N auf absolute Häufigkeiten zurückgeführt werden.

Die Testgröße ergibt sich nun wie folgt:

[2] Allgemein prüft der χ^2-Test die Abhängigkeit zweier kategorialer Variablen. Kann die erste Variable m und die zweite n Werte annehmen, ergibt sich eine Kontingenztabelle der Größe mxn.

$$\chi^2 = \sum_{m,n} \frac{(O_{mn} - E_{mn})^2}{E_{mn}}, \tag{9.1}$$

wo O_{mn} die beobachtete und E_{mn} die gemäß $H0$ zu erwartende Häufigkeit im Feld $[m, n]$ der Kontingenztabelle bezeichnen. Für ausreichend große zu erwartende Häufigkeiten (in keinem Feld < 5) ist χ^2 auch χ^2-verteilt, und es können Signifikanzaussagen getroffen werden. Übersteigt χ^2 einen gewissen Wert, bei einem Freiheitsgrad und einem gewählten Signifikanzniveau $\alpha = 0.005$ beträgt er 7.88, kann $H0$ verworfen werden und von einer Abhängigkeit, also einer kollokativen Verbindung zwischen w_i und w_j ausgegangen werden.

9.4 Extrahierung mittels *Pointwise Mutual Information*

Die *Pointwise Mutual Information* $I(w_i, w_j)$ ist wie in Abschnitt 3.7 dargestellt ein symmetrisches Maß für den Zusammenhang zweier Types w_i und w_j. Sie gibt an, wieviel Information über das Auftreten von w_j durch das Auftreten von w_i bereitgestellt wird und umgekehrt:

$$I(w_i, w_j) = \frac{P(w_i, w_j)}{P(w_i)P(w_j)} = \frac{P(w_i|w_j)}{P(w_i)} = \frac{P(w_j|w_i)}{P(w_j)} \tag{9.2}$$

Je größer $I(w_i, w_j)$, desto höher die Abhängigkeit, die zwischen w_i und w_j angenommen werden kann. Allerdings muss beachtet werden, dass $I(w_i, w_j)$ auch umgekehrt proportional von der Häufigkeit der Types w_i und w_j abhängt. Treten w_i und/oder w_j selten auf, tendiert dieses Maß dazu, ihren Zusammenhang zu überschätzen (warum?).

9.5 Extrahierung mittels Likelihood-Verhältnis

Hypothesen Ins Verhältnis gesetzt werden die Voraussagen zweier gegensätzlicher Hypothesen hinsichtlich des Auftretens der Types w_i und w_j.

$$\text{H0:} \quad P(w_i|w_j) = p = P(w_i|\neg w_j) \tag{9.3}$$

$$\text{H1:} \quad P(w_i|w_j) = p_1 \neq p_2 = P(w_j|\neg w_j) \tag{9.4}$$

Mit $H0$ ist die Unabhängigkeitsannahme von w_i und w_j formuliert, mit $H1$ die Abhängigkeitsannahme. Ist w_i im hohen Maße abhängig von w_j und damit anhand diesem vorhersagbar, dann liegt der Verdacht nahe, dass es sich hierbei um eine Kollokation handelt.

Schätzung der Wahrscheinlichkeiten Die Maximum-Likelihood-Schätzung von p, p_1 und p_2 ergibt:

$$p = P(w_i) = \frac{c_i}{N} \qquad p_1 = P(w_i|w_j) = \frac{c_{ij}}{cj} \qquad p_2 = P(w_i|\neg w_j) = \frac{c_i - c_{ij}}{N - c_j} \tag{9.5}$$

wobei c_* gleich der beobachteten Häufigkeit von Type $*$ und N gleich der Anzahl der Tokens im Trainingskorpus. p ist die Wahrscheinlichkeit von w_i, wie sie unabhängig von w_j gegeben ist. p_1 und p_2 dagegen sind bedingte Wahrscheinlichkeiten, die das Auftreten von w_i vom Vorhandensein, beziehungsweise Nichtvorhandensein von w_j abhängig machen.

Binomialverteilung Man nimmt nun vereinfachend an, dass die Auftretenswahrscheinlichkeit von w_i einer Binomialverteilung gehorcht. Das bedeutet wie in Abschnitt 2.5.2 beschrieben: es lässt sich nur w_i oder *nicht* w_i beobachten und jedes Auftreten, bzw. Nicht-Auftreten von w_i ist von allem vorherigen Auftreten, bzw. Nicht-Auftreten w_i's unabhängig. Zur Erinnerung: mit Hilfe der Binomialverteilung lässt sich die Wahrscheinlichkeit $b(k; n, x)$ berechnen für k Treffer (Auftreten von w_i) bei n Versuchen (Auftreten, bzw. Nicht-Auftreten von w_j) und der Trefferwahrscheinlichkeit x (p bei Unabhängigkeit, bzw. p_1, p_2 bei Abhängigkeit).

Likelihood-Verhältnis $H0$ und $H1$ machen nun unterschiedliche Voraussagen über die zu erwartenden Häufigkeiten c_{ij} (Auftreten von w_i zusammen mit w_j) und $c_i - c_{ij}$ (Auftreten von w_i ohne w_j). Für beide Hypothesen lässt sich nun die Likelihood der tatsächlich beobachteten Häufigkeiten berechnen, d.i. die Wahrscheinlichkeit für das Zustandekommen dieser Häufigkeiten im Falle der Gültigkeit von $H0$, beziehungsweise von $H1$.

$$L(H0) = b(c_{ij}; c_j, p)b(c_i - c_{ij}; N - c_j, p) \tag{9.6}$$
$$L(H1) = b(c_{ij}; c_j, p_1)b(c_i - c_{ij}; N - c_j, p_2) \tag{9.7}$$

$L(H0)$ bezeichnet die Likelihood, mit der w_i c_{ij}-mal zusammen mit w_j zu beobachten ist, und $c_i - c_{ij}$ mal ohne w_j, sollte w_i tatsächlich mit gleicher Wahrscheinlichkeit p mit oder ohne w_j auftreten, so wie es $H0$ vorhersagt.

Da gemäß $H1$ die Auftretenswahrscheinlichkeit für w_i von w_j abhängt, unterscheidet sich die Wahrscheinlichkeit p_1 gegeben w_j von p_2 gegeben Nicht-w_j.

Setzt man nun die beiden Likelihoods ins Verhältnis und logarithmiert das ganze, ergibt sich:

$$\begin{aligned} \ln \lambda &= \ln \frac{L(H0)}{L(H1)} \\ &= \ln L(c_{ij}, c_j, p) + \ln L(c_i - c_{ij}, N - c_j, p) \\ &\quad - \ln L(c_{ij}, c_j, p_1) - \ln L(c_i - c_{ij}, N - c_j, p_2) \end{aligned} \tag{9.8}$$

wobei $L(k, n, x) = x^k(1-x)^{n-k}$.

Interpretation Man erhält einen Wert, der angibt wieviel mal wahrscheinlicher die beobachteten Daten unter Annahme der einen Hypothese zustande kämen, als unter Annahme der anderen. Ist $\ln \lambda$ größer 0, so lassen sich die tatsächlich beobachteten Häufigkeiten von w_i und w_j besser mit der Unabhängigkeitshypothese erklären. Ist $\ln \lambda$ kleiner 0, gilt vielleicht das Gegenteil. Um zu testen, ob wirklich davon auszugehen ist, dass w_i von w_j abhängt, multipliziert man $\ln \lambda$ mit -2, da $-2 \ln \lambda$ annähernd χ^2-verteilt ist und anhand des χ^2-Werts Signifikanzaussagen getroffen werden können (vgl. Abschnitt 9.3). Bei einem Freiheitsgrad und dem Signifikanzniveau von $\alpha = 0.005$ ist der zu übertreffende Wert gleich 7.88. Ist $-2 \ln \lambda$ größer als dieser Wert, kann es sich bei w_i und w_j um eine Kollokation handeln.

Beispiel Im SI1000P-Korpus beträgt der aus dem Likelihood-Verhältnis abgeleitete χ^2-Wert für die Types *de* und *Gaulle* 1597.73. Das lässt sich so interpretieren, dass die Abhängigkeitshypothese $H1$ die tatsächlichen Häufigkeiten von *de/NE* und *Gaulle/NE* $e^{0.5 \cdot 1597.73}$ mal besser erklärt als $H0$.

9.6 Filterung

Um zu vermeiden, dass der Zusammenhang selten auftretender Wörter überbewertet wird, wird für c_{ij} in der Regel eine Mindestanzahl festgelegt.

Kapitel 10

Hidden-Markov-Modelle

Hidden-Markov-Modelle (HMMs) sind Netzwerke aus Zuständen und Übergängen, wobei die Übergänge jeweils mit einer gewissen Wahrscheinlichkeit passiert werden können und an den Zuständen mit einer gewissen Wahrscheinlichkeit bestimmte Ausgabesymbole generiert werden.[1]

HMMs eignen sich ausgezeichnet zur statistischen Sprachmodellierung, da für sie eine Reihe effizienter Algorithmen existieren, die garantiert in einem (wenigstens lokalen) Optimum konvergieren, also zu bestmöglichen Ergebnissen führen. Sie können immer dann zum Einsatz kommen, wenn ein **Noisy-Channel**-Problem gegeben ist (vgl. Abschnitt 3.8): Es liegt ein beobachteter Output vor (beispielsweise eine Wortfolge), und das HMM liefert den am wahrscheinlichsten zugehörigen Input (z.B. POS-Sequenz) anhand der durchlaufenen Zustandssequenz, also des Wegs durch das Modell. Vor diesem Hintergrund ist das HMM als ein **Erzeugungsmodell** zu betrachten, dass die beobachtete Sequenz mit einer gewissen Wahrscheinlichkeit generiert hat.

Den Namen "Markov" haben wir schon kennengelernt im Zusammenhang mit der Annahme, das die Wahrscheinlichkeit eines Ereignisses nur von einer beschränkten Ereignisvorgeschichte abhängt (**Limited Horizon**). Hinzu kommt hier noch eine weitere vereinfachende Annahme: die der **Zeitinvarianz**, bedingte Wahrscheinlichkeiten ändern sich nicht im Laufe der Zeit (also zum Beispiel in Abhängigkeit der Stelle im Text).

"Hidden" ist das Markov-Modell deswegen, weil die durchlaufene Zustandssequenz, die tatsächlich der Beobachtung zugrundeliegt, unbekannt ist (*noisy channel*), es lässt sich für jede Zustandssequenz nur eine Wahrscheinlichkeit ermitteln.

[1] Formal entspricht ein HMM einem **probabilistischen Automaten**. Ob die Ausgabesymbole an den Zuständen oder den Übergängen emittiert werden, ist dabei unerheblich.

10.1 Aufbau

Ein HMM ist beschreibbar als 5-Tupel $< Q, K, S, A, B >$bestehend aus:

- Menge von Zuständen $Q = \{q_i\}$
- Ausgabealphabet K (kategorial oder kontinuierlich)
- Startwahrscheinlichkeiten $S = \{s_i\}$: dass man sich zu Beginn im Zustand i befindet
- Übergangswahrscheinlichkeiten $A = \{a_{ij}\}$: von Zustand i zu Zustand j
- Emissionswahrscheinlichkeiten (*observation likelihoods*) $B = \{b_{jo_t}\}$: im Zustand j für Beobachtung o_t

Das mit dem HMM verbundene Wahrscheinlichkeitsmodell μ beinhaltet die Komponenten $< A, B >$. S wird in der Regel aus A oder B abgeleitet (siehe Abschnitt 11.4).

10.2 Wahrscheinlichkeit einer Beobachtung

Ein HMM lässt sich wie gesagt als Modell betrachten, das eine beobachtete Sequenz **erzeugt**. In der Spracherkennung mit Hilfe von HMM's ist jedem Sprachlaut (mindestens) ein HMM zugeordnet. Um ein neues Signal – vereinfachend sei angenommen, dass es sich um einen isolierten Laut handelt – klassifizieren zu können, wird für jedes HMM berechnet, mit welcher Wahrscheinlichkeit es dieses Signal erzeugen würde. Der Sprachlaut, der mit dem HMM der höchsten Wahrscheinlichkeit assoziiert ist, wird schließlich als Klassifikationsergebnis ausgegeben.

O sei eine observierte Sequenz der Länge T. Die Wahrscheinlichkeit, mit der O von einem HMM mit den Wahrscheinlichkeitsspezifikationen μ erzeugt wird ergibt sich für jede denkbare Zustandssequenz $X = (X_1, X_2, \ldots, X_T) \in Q^T$ der Länge (also zum Zeitpunkt) T wie folgt:

$$\begin{aligned} P(O|\mu) &= \sum_{X \in Q^T} P(O|X, \mu) P(X, \mu) \\ &= \sum_{X \in Q^T} (\prod_{t=1}^{T} a_{X_{t-1} X_t} b_{X_t o_t}) \end{aligned} \tag{10.1}$$

Es werden also für jede der bis zum Zeitpunkt T denkbarerweise durchlaufenen Zustandssequenzen Übergangs- und Emissionswahrscheinlichkeiten aufmultipliziert und die so erhaltenen Einzelpfadwahrscheinlichkeiten addiert. Ein effizientes Verfahren hierfür ist der **Forward-Backward-Algorithmus**, auf den hier nicht näher eingegangen werden soll. Meistens ist

nämlich weniger die Gesamtwahrscheinlichkeit als die höchstmögliche Wahrscheinlichkeit interessant, mit der die beobachteten Daten auf einem Pfad durchs HMM generiert werden können, sowie die damit verbundene Zustandssequenz. Der nächste Abschnitt geht kurz darauf ein, Details werden dann anhand von HMM-Anwendungen besprochen.

10.3 Finden der besten Zustandssequenz: Viterbi-Algorithmus

Der Viterbi-Algorithmus ist ein Verfahren der **Dynamischen Programmierung**, grob definiert als Suche des optimalen Pfades durch eine Tabelle durch sukzessive Ermittlung der Tabellenwerte. Die Tabelle, um die es sich hier handelt, ist eine sogenannte **Trellis**, ein Zustand-Zeitpunkt-Gitter: ein Knoten entspricht einem Zustand des Modells zu einem bestimmten Zeitpunkt. Ziel ist es, denjenigen Pfad durch die Trellis zu finden, der die höchste Wahrscheinlichkeit für die beobachtete Sequenz liefert. Dies wird dadurch erreicht, dass an jedem Knoten die maximal mögliche Wahrscheinlichkeit der bis dahin beobachteten Sequenz vermerkt wird zusammen mit dem vorangehenden Pfad, auf dem sich diese Wahrscheinlichkeit akkumuliert hat. Im Abschnitt 11.4.2 wird der Algorithmus anhand eines konkreten Beispiels eingehender vorgestellt.

10.4 Parameter-Schätzung

In diesem Abschnitt geht es darum, wie die μ-Parameter A und B eines HMM anhand von Trainingsdaten festgelegt werden können. Ziel ist, die Parameter so zu bestimmen, dass die Trainingsdaten $O = (o_1, \ldots, o_T)$ mit höchstmöglicher Likelihood $L(O|\mu)$ (siehe Gleichung 10.1) vom HMM erzeugt werden können. Hierfür gibt es keine **analytische** Lösung, also keine rechnerische Möglichkeit, das **globale Maximum** von $L(O|\mu)$ im Parameterraum zu finden. Ein numerisches **Hill-Climbing-Verfahren** stellt die Parameter aber so ein, dass ein **lokales Maximum** erreicht wird. Dieses Verfahren heißt **Baum-Welch-Algorithmus** und ist eine spezielle Ausprägung des **EM-Algorithmus**, den wir bereits in Abschnitt 6.3 zur Bestimmung von Interpolationsgewichten kennengelernt haben.

Bei diesem Verfahren wird das Ziel verfolgt, anhand der beobachteten Daten festzustellen, welche Zustandsübergänge und Symbolemissionen am häufigsten aufgetreten sind und durch eine Erhöhung von deren Wahrscheinlichkeiten das HMM weiter an die beobachteten Daten anzupassen, was sich in einer Erhöhung der Likelihood der Daten wiederspiegelt.

Der Baum-Welch-Algorithmus ist in Abbildung 10.1 dargestellt. a_{ij} sei dabei die Übergangswahrscheinlichkeit von Zustand i in Zustand j, b_{io} sei die Emissionswahrscheinlichkeit zur Erzeugung von Beobachtung o in Zustand i.

$\#_*$ steht für Häufigkeit von $*$: $\#_i$ bezeichnet die Häufigkeit, mit der Zustand i besucht wird, $\#_{ij}$ die Übergangshäufigkeit von Zustand i nach Zustand j und $\#_{io}$ die Häufigkeit, mit der im Zustand i das Symbol o beobachtet wird.

definiere relevante Statistik für $a_{ij} := \#_i, \#_{ij}$
definiere relevante Statistik für $b_{io} := \#_i, \#_{io}$
initialisiere alle a_{ij}, b_{io}
iteriere (über k)

E-Schritt:

1. definiere folgende Hilfsvariablen:

$$\alpha_t(i) := P(o_1 \dots o_t, X_t = i|\mu) \tag{10.2}$$
$$\beta_t(i) := P(o_{t+1} \dots o_T|X_t = i, \mu) \tag{10.3}$$
$$\gamma_t(i) := P(X_t = i|o_1 \dots o_T, \mu) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \tag{10.4}$$

2. berechne die Erwartungswerte der relevanten Statistiken:

$$\#_i = \sum_t \gamma_t(i) \tag{10.5}$$
$$\#_{ij} = \sum_t P(X_t = i, X_{t+1} = j|o_1 \dots o_T, \mu) \tag{10.6}$$
$$= \sum_t \frac{\alpha_t(i)a_{ij}^{[k]}b_{jo_{t+1}}^{[k]}\beta_{t+1}(j)}{\sum_i \sum_j \alpha_t(i)a_{ij}^{[k]}b_{jo_{t+1}}^{[k]}\beta_{t+1}(j)} \tag{10.7}$$
$$\#_{io} = \sum_{t:o_t=o} \gamma_t(i) \tag{10.8}$$

M-Schritt:

$$a_{ij}^{[k+1]} = \frac{\#_{ij}}{\#_i} \tag{10.9}$$
$$b_{io}^{[k+1]} = \frac{\#_{io}}{\#_i} \tag{10.10}$$

terminiere, wenn $L(O|\mu^{[k+1]}) \approx L(O|\mu^{[k]})$

Abbildung 10.1: Schätzung von HMM-Parametern

Ziel ist es also, Werte für die Parameter μ, also für die Transitionswahrscheinlichkeiten a_{ij} und die Emissionswahrscheinlichkeiten b_{io} zu finden, die den beobachteten Daten $o_1, \dots, o_T$ eine höchstmögliche Likelihood zukommen lassen. Geschätzt werden diese Parameter nach anfänglicher Initialisierung anhand der erwarteten Transitions- und Emissionshäufigkeiten. $\#_i$ bezeichnet hierbei die Häufigkeit, mit der Zustand i besucht wird, $\#_{ij}$ die Transitionshäufigkeit von Zustand i nach Zustand j und $\#_{io}$ die Emissionshäufigkeit des Symbols o in Zustand i.

10.4.1 E-Schritt

Hier werden die für eine gegebene Belegung der a_{ij} und b_{io} erwarteten Häufigkeiten $\#_i$, $\#_{ij}$ und $\#_{io}$ ermittelt.

Für eine effiziente Berechnung dieser Größen bedarf es der Hilfsvariablen $\alpha_t(i)$, $\beta_t(i)$ und $\gamma_t(i)$.

Backward-Probability $\alpha_t(i)$ bezeichnet die Wahrscheinlichkeit, mit der sich das HMM bei gegebenen Parametern μ zum Zeitpunkt t in Zustand i befindet ($X_t = i$) und auf dem Pfad dorthin die beobachtete Symbolfolge $o_1 \dots o_t$ erzeugt hat (**Gleichung 10.2**).

Forward-Probability $\beta_t(i)$ bezeichnet die Wahrscheinlichkeit, mit der – gegeben das HMM befindet sich zum Zeitpunkt t in Zustand i – die noch ausstehende Symbolfolge $o_{t+1}, \dots, o_T$ generieren wird (**Gleichung 10.3**).

Kombination Anhand von $\alpha_t(i)$ und $\beta_t(i)$ lässt sich nun $\gamma_t(i)$ ermitteln, also die Wahrscheinlichkeit, mit der der Zustand i bei Erzeugung der Trainingsdaten $o_1, \dots, o_T$ zum Zeitpunkt t durchlaufen wird. Hierzu werden $\alpha_t(i)$ und $\beta_t(i)$ multipliziert und das Produkt auf die Forward-Backward-Produkte aller Zustände normiert (**Gleichung 10.4**).

Erwartete Zustandshäufigkeit Die erwartete Häufigkeit $\#_i$ ergibt sich durch Aufsummieren der γ-Wahrscheinlichkeiten über alle Zeitpunkte t (**Gleichung 10.5**).[2]

Erwartete Transitionshäufigkeit Für $\#_{ij}$ werden die Transitionswahrscheinlichkeiten $P(X_t = i, X_{t+1} = j | o_1 \dots o_T, \mu)$ von Zustand i nach j zum Zeitpunkt t gegeben Beobachtungssequenz $o_1 \dots o_T$ über alle Zeitpunkte aufsummiert (**Gleichung 10.7**). Die Transitionswahrscheinlichkeiten setzen sich zusammen aus der Backward-Probability von i, der Forward-Probability von j, dem derzeitigen Wert der (zeit- und beobachtungsunabhängigen) Übergangswahrscheinlichkeit aus A von i nach j, sowie dem derzeitigen Wert der Emissionswahrscheinlichkeit aus B für Beobachtung o_{t+1} in Zustand j.

Erwartete Emissionshäufigkeit Für die Emissionshäufigkeit $\#_{io}$ von Symbol o in Zustand i werden die γ_i-Wahrscheinlichkeiten derjenigen Zeitpunkte aufsummiert, zu denen o beobachtet wurde (**Gleichung 10.8**).

[2] vgl. die erwartete Häufigkeit beim Würfeln: wenn 12 mal gewürfelt wird ($T = 12$), dann ist die erwartete Häufigkeit des Würfelns einer 2 ($i = 2$) bei einer konstanten (also zu allen Zeitpunkten t gleichen) Wahrscheinlichkeit von $\frac{1}{6}$ gleich $12 \cdot \frac{1}{6} = 2$. Der Index t ermöglicht oben zusätzlich eine zeitlich variierende Wahrscheinlichkeit.

10.4.2 M-Schritt

Die Maximum-Likelihood-Schätzung der Übergangswahrscheinlichkeiten a_{ij} und der Emissionswahrscheinlichkeiten b_{io} anhand der im E-Schritt ermittelten erwarteten Häufigkeiten erfolgt gemäß der Formel 2.1 für bedingte Wahrscheinlichkeiten (**Gleichungen 10.9 und 10.10**).

10.4.3 Abbruchkriterium

Nach jedem Iterationsschritt wird geprüft, ob sich mit den neu geschätzten Modellparametern die Likelihood der beobachteten Daten ausreichend stark erhöht. Ist dies nicht der Fall, so ist ein lokales Maximum der Likelihood erreicht, und das Verfahren terminiert.

Kapitel 11

Part-of-Speech-Tagging

11.1 Aufgabenstellung

Unter Part-of-Speech-(POS)-Tagging versteht man, Wörter mit ihren Wortarten zu versehen. Aus zwei Gründen ist ein reiner **Lexikon-Lookup**, also ein Nachschlagen der Wortart in einem Lexikon unzureichend:

- **Out of Vocabulary-Fälle (OOV):** POS ist für Wörter, die nicht im Lexikon stehen, unbekannt
- **Ambiguitäten:** ein Worttype kann verschiedene POS-Labels tragen, Beispiele hierfür sind:
  - *Sucht*: NN vs. VVFIN[1]
  - *ab*: ADP vs. PTKVZ
  - *als*: KOKOM vs. KOUS
  - *am*: APPRART vs. PTKA
  - *Anfangs*: NN vs. ADV

Zu lösen sind diese Probleme durch Einbeziehen des **Wortkontexts** und POS-relevanter **Worteigenschaften** (z.B. String-Suffixe).

11.2 Tagsets

Ein Tagset beinhaltet die Wortarten, die von einem Tagger vergeben werden können. Das fürs Deutsche bekannteste Set ist das **Stuttgart-Tübingen Tagset (STTS)**, das 54 Wortarten umfasst. Zwei Beispiele für englische Sets sind das **Penn Tagset** (45 *Tags*) und das **C7 Tagset** (146 *Tags*).

[1] Die POS-Kürzel stammen aus dem STTS, vgl. Abschnitt 11.2

11.3 Tagger-Überblick

Beim POS-Tagging sind zu unterscheiden:

- Regelbasierte Verfahren: ENGTWOL (Voutilainen, 1995)
- Statistische Verfahren (z.B. Markov-Tagger): Jelinek (1985), Tree Tagger (Schmidt, 1995), Reichel (2005)
- Transformationsbasierte Verfahren: Brill (1995)

Auf die zweite Verfahrensklasse soll in diesem Seminar eingegangen werden.

11.4 Markov-Tagger

11.4.1 Grundform eines Markov-Taggers (Jelinek, 1985)

Das Tagging-Problem lässt sich im Rahmen eines **Noisy-Channel-Modells** (vgl. Abschnitt 3.8) formalisieren: es wird versucht, die einer am Ausgang des Kanals beobachteten Wortfolge W zugrundeliegende POS-Sequenz G, die den Input des Kanals darstellt, zu finden. G ist unbekannt und muss durch Maximierung von $P(G|W)$ als $\hat{G}$ rekonstruiert werden.

$$\hat{G} \quad = \quad \arg\max_G \Big[P(G|W)\Big] \tag{11.1}$$

$P(G|W)$ lässt sich nicht direkt schätzen, daher wird dieser Ausdruck zunächst mit dem Satz von Bayes (vgl. Abschnitt 2.3) umformuliert und das zur Maximierung irrelevante $P(W)$ entfernt:

$$\hat{G} = \arg\max_G \left[\frac{P(G)P(W|G)}{P(W)}\right] = \arg\max_G \Big[P(G)P(W|G)\Big] \tag{11.2}$$

Gleichung 11.2 lässt sich gemäß der Kettenregel (vgl. Abschnitt 2.2) so darstellen:

$$P(G)P(W|G) = \prod_{t=1}^{T} P(w_t|w_1g_1,\ldots,w_{t-1}g_{t-1}g_t)P(g_t|w_1g_1,\ldots,w_{t-1}g_{t-1}) \tag{11.3}$$

Mittels zweier vereinfachender Annahmen lässt sich diese Gleichung schließlich lösen:

- Die Wahrscheinlichkeit des Worts w_t hängt nur von seinem *Tag* g_t ab.

- Die Wahrscheinlichkeit des *Tags* g_t hängt von einer begrenzten *Tag*-Vorgeschichte $g\text{-history}_t$ ab (Markov-Annahme).

$\hat{G}$ ergibt sich nun wie folgt:

$$\hat{G} = \arg \max_{g_1,\ldots,g_T}, \Big[\prod_{t=1}^{T} P(g_t|g\text{-history}_t) P(w_t|g_t) \Big] \tag{11.4}$$

11.4.2 Wahrscheinlichste *Tag*-Sequenz mittels Viterbi

Gleichung 11.4 lässt sich mit Hilfe des in Kapitel 10 angesprochenen Viterbi-Algorithmus lösen:

Zunächst wird eine **Trellis**, ein Zustand-Zeitpunkt-Gitter aufgebaut. Ein Knoten der Trellis entspricht dabei einem POS-*Tag* zu einem bestimmten Zeitpunkt, also zum Index des jeweiligen Worts im zu taggenden Text.[2]

Ziel ist es, für die beobachtete Wortfolge denjenigen Pfad durch die Trellis zu finden, in dem das Produkt aus Emissions- und Übergangswahrscheinlichkeiten maximiert wird. Damit wird auch $\Big[P(G)P(W|G)\Big]$ und letztlich $\Big[P(G|W)\Big]$ maximiert. Die mit der Maximierung verbundene *Tag*-Sequenz $g_1, \ldots, g_T$ bildet dann das Tagging-Resultat.

In jedem Knoten $k_j(t)$ der Trellis für *Tag* j und Zeitpunkt t wird folgendes notiert:

- die Wahrscheinlichkeit $\delta_j(t)$ des bis hierhin wahrscheinlichsten Pfads,
- der Vorgängerknoten auf diesem Pfad.

Die Ermittlung der $\delta_j(t)$'s funktioniert folgendermaßen:

- **Initialisierung:**[3]

$$\delta_j(1) = b_{jo_1}$$

- **Induktion:**[4]

$$\delta_j(t) = \max_i \left[\delta_i(t-1) a_{ij} b_{jo_t}\right]$$

Initialisierung Bei der Initialisierung zum Zeitpunkt $t = 1$ werden die δ's für jeden POS j gleich den bedingten Wahrscheinlichkeiten P(erstes Wort im zu taggenden Text | POS j) gesetzt. Diese werden ausgedrückt als Emissionswahrscheinlichkeiten b_{jo_1} (vgl. Notation in Kapitel 10).

[2] **Indizes:** über Zustände (*Tags*): $1 \leq i, j \leq N$; über Zeitpunkte: $1 \leq t \leq T$ bei einem POS-Inventar der Größe N und einem zu taggenden Text der Länge T

[3] Zur Ermittlung von $\delta_j(1)$ kann zusätzlich zur Emissionswahrscheinlichkeit noch die Transitionswahrscheinlichkeit von Satzgrenze g zu *Tag* j herangezogen werden, da ein Textbeginn ja auch einen Satzbeginn darstellt: $\delta_j(1) = a_{gj} \cdot b_{jo_1}$.

[4] **Induktion (informell):** Fortführung eines für n gültigen Sachverhalts mit $n + 1$.

Induktion Im Zuge der Induktion wird für jeden Knoten $k_j(t)$ der Trellis für POS j zum Zeitpunkt t die Wahrscheinlichkeit $\delta_j(t)$ des wahrscheinlichsten Pfads hin zu diesem Knoten ermittelt. Diese ergibt sich durch Multiplikation der Emissionswahrscheinlichkeit b_{jo_t} des t-ten Worts im zu taggenden Text gegeben POS j mit dem Maximum unter den Produkten

> (*Wahrscheinlichkeiten der Pfade zu den Knoten zum Zeitpunkt* $t-1$) x (*Übergangswahrscheinlichkeiten* a_{ij} *zwischen den Knoten der Ebene* $t-1$ *und* $k_j(t)$).

Der Vorgänger von $k_j(t)$ im wahrscheinlichsten Pfad wird ebenfalls am Knoten vermerkt, damit der Pfad dann nach Zeitpunkt T, also nach dem letzten Wort, ausgelesen werden kann.

11.4.3 Tree-Tagger (Schmidt, 1995)

Der Tree-Tagger schätzt die $P(w_t|g_t)$ mittels eines Lexikons, das Vollformen und Suffixe enthält und die $P(g_t|\text{g-history}_t)$ mittels eines Entscheidungsbaum, der für *g-history* einen Vektor mit den Wahrscheinlichkeiten der POS-*Tags* ausgibt. *g-history* ist dabei als Pfad im Baum repräsentiert. Das OOV-Problem wird auf diese Weise reduziert.

11.4.4 Integration von Worteigenschaften (Reichel, 2005)

Als zusätzliche Wissensquelle neben dem Wortkontext (der POS-Vorgeschichte) lassen sich wie beim Tree-Tagger schon gesehen auch Worteigenschaften nutzen, die POS-Informationen bereithalten. In Sprachen wie dem Deutschen und Englischen sind dies vor allem Suffix-Morpheme, Flexionsendungen und finale Kompositumglieder (*Gelegen**heit**/NN*, *Umgehungs**straße**/NN*, *part**ly**/ ADV*). Die Nutzung dieser Einheiten reduziert das **OOV-Problem**.

Entgegen dem Tree-Tagger-Ansatz kann auch der Versuch unternommen werden, solche Einheiten **ohne linguistisches Wissen** zu ermitteln, was keine linguistische Vorarbeit erfordert und den Tagger sprachunabhängig macht. In den folgenden beiden Abschnitten soll dargestellt werden, wie ein Markov-Tagger zur Integration dieser zusätzlichen Informationen erweitert werden kann und wie diese Worteigenschaften ohne linguistisches Wissen gewonnen werden können.

Verallgemeinerung des Markov-Taggers

Die Verallgemeinerung geschieht mittels **Linearer Interpolation** mehrerer Wahrscheinlichkeitsmodelle (vgl. Kapitel 6).

- Interpolation von N-Gramm-Modellen für die POS-Vorgeschichte: ersetze $P(g_t|\text{g-history}_t)$ durch $\sum_j u_j P(g_t|\text{g-history}_{tj})$

- Interpolation von Modellen für unterschiedliche Wortrepräsentationen: ersetze $P(w_t|g_t)$ durch $\frac{P(w_t)}{P(g_t)} \sum_k v_k P(g_t|\text{w-representation}_{tk})$

Gleichung 11.4 lautet dann in dieser verallgemeinerten Form:

$$\hat{T} = \arg \max_{g_1,\ldots,g_T} \Big[\prod_{t=1}^{T} \frac{1}{P(g_t)} \sum_j u_j P(g_t|\text{g-history}_{tj}) \sum_k v_k P(g_t|\text{w-representation}_{tk}) \Big] \quad (11.5)$$

Wortrepräsentation

Wörter werden hier repräsentiert als Liste automatisch gewonnener String-Suffixe. Die Gewinnung linguistisch relevanter String-Suffixe kann beispielsweise mittels der **Weighted Backward Successor Variety (SV)** erfolgen. Die **SV** eines Strings steht für die Anzahl verschiedener Characters, die ihm in einem Lexikon folgen können. **Backward SV** bedeutet, dass die SV von gespiegelten Strings ermittelt wird (da in unserem Fall ja die Suffixe interessieren). Die SV's werden in Abhängigkeit der mittleren SV an der entsprechenden String-Position gewichtet, um Positions-Effekte zu eliminieren (die zu Beginn hohe SV sinkt kontinuierlich beim Durchlaufen des Strings ab). Lokale SV-Gipfel werden schließlich als Morphemgrenzen betrachtet (vgl. Peak and Plateau Algorithmus von Nascimento et al., 1998).

Implementieren lässt sich das notwendige Lexikon gespiegelter Wörter als **Trie**. Darunter versteht man einen Baum zur Speicherung von Strings, der für jedes gemeinsame Präfix einen Knoten enthält. Die SV an einem Knoten entspricht der Anzahl der abgehenden Transitionen.

Abbildung 11.4.4 zeigt einen solchen Lexikon-Trie mit den gespiegelten Einträgen *Einigung, Kreuzigung* und *Eignung*. Die SV-Maxima an den Knoten 3 und 5 entsprechen den Grenzen der Morpheme *ung* and *ig*.

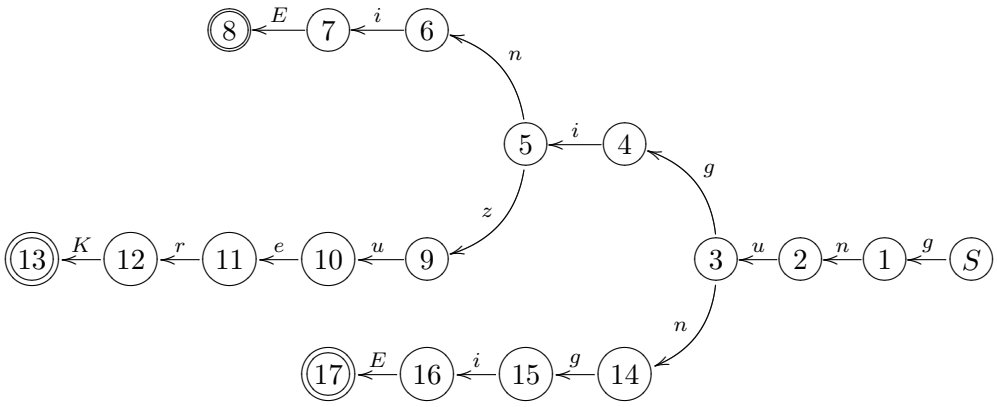

Abbildung 11.1: Lexikon-Trie mit SV-Maxima als Morphemgrenzentsprechungen

11.5 Evaluierung

Gemessen wird bei der Evaluierung die Performanz des Taggers für ein Testcorpus, also der Anteil der korrekt getaggten Tokens.

11.5.1 Referenzen

Die Evaluierung kann sich auf zwei unterschiedliche Referenzen beziehen:

- **Gold-Standard:** die Referenz bilden manuell gesetzte *Tags*. Relativiert wird die Güte dieser Referenz dadurch, dass das **Inter-Tagger-Agreement** in der Regel weniger als 100 % beträgt, was zeigt, dass es auch unter Experten Uneinigkeit über die Wahl der richtigen Wortart gibt.
- **Baseline:** die Referenz ist durch den Output eines Baseline-Taggers gegeben, beispielsweise eines Unigramm-Taggers, der keine POS-Vorgeschichte nutzt.

11.5.2 Vergleichende Evaluierung

Möchte man die Performanz verschiedener Tagger miteinander vergleichen, empfiehlt es sich, Mittelwerte und Varianzen über mehrere Stichproben zu ermitteln (vgl. Kapitel 8). Um Tagger bei beliebiger Tagset-Größe gegenüberstellen zu können, muss der Anteil korrekt getaggter Wörter in Form der **Kappa-Statistik** korrigiert werden.[5]

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{11.6}$$

$P(A)$ bezeichnet hierbei den Anteil der korrekt klassifizierten Wörter und $P(E)$ den zu erwartender Anteil zufällig korrekt klassifizierter Wörter $\frac{1}{|\text{Tagset}|}$.

[5]Bei einem Tagset der Größe 1 ist die Performanz des Taggers, der nur diesen einen *Tag* vorhersagt, 100 %.

Kapitel 12

Text-Klassifikation

12.1 Aufgabenstellung

Dieses Kapitel behandelt knapp die Vorgehensweise bei der Erstellung von Textklassifikationssystemen. Ziel ist es, einen beliebigen Text einer von mehreren vorgegebenen Kategorien zuordnen zu können (z.B. *Kochrezept* vs. *Heimwerkerinformation*; oder *Werk von Thomas Mann* vs. *Heinrich Mann*; oder *terroristische* vs. *nicht-terroristische E-Mail*).

Solche Systeme erleichtern die Suche bestimmter Dokumententypen in großen Textressourcen ungemein.

In den nächsten Abschnitten wird ein mögliches Vorgehen zur Erstellung solcher Systeme vorgestellt. Es handelt sich um ein überwachtes Lernverfahren (vgl. Abschnitt 7.2).

12.2 Bayes'sche Klassifikatoren

Als Trainingsmaterial dient ein Dokument-Korpus, in dem jedes der enthaltenen Dokumente mit dem entsprechenden Kategorie-Namen gelabelt ist. Für jeden Dokumenttyp t lässt sich ein Sprachmodell P_t trainieren, das die Wortfolgewahrscheinlichkeiten in Texten des Typs t beinhaltet.

Die Klassifikationsfunktion eines neuen Dokuments d lautet nun beim Bayes'schen Klassifikationsverfahren wie folgt:

$$\hat{t} \quad = \quad \arg\max_t \left[P(t|d)\right] \tag{12.1}$$

Gesucht wird also derjenige Dokumenttyp t, der am wahrscheinlichsten zu wählen ist, wenn Dokument d vorliegt. Anhand der Trainingsdaten lässt sich $P(t|d)$ nicht direkt berechnen, dafür aber die Wahrscheinlichkeit $P(d|t)$ anhand der gegebenen Sprachmodelle P_t. Es bedarf also einer bayes'schen Umformulierung der Gleichung 12.1 wie in Kapitel 2 erläutert:

$$
\begin{aligned}
\hat{t} &= \arg\max_t [P(t|d)] \\
&= \arg\max_t \Big[\frac{P(d|t) \cdot P(t)}{P(d)}\Big] \qquad (12.2) \\
&= \arg\max_t [P(d|t) \cdot P(t)] \qquad (12.3)
\end{aligned}
$$

$P(d|t)$ ist die Wortfolgewahrscheinlichkeit von d im Sprachmodell, das anhand von Dokumenten des Typs t trainiert worden ist. $P(t)$ gibt die Wahrscheinlichkeit an, mit der ein Dokument des Typs t überhaupt auftritt. Klassifikationsergebnis ist derjenige Dokumenttyp t, der das Produkt dieser beiden Terme maximiert.

Zur Verdeutlichung betrachten wir die Kategorien K (für *Kochrezepte*) und H (für *Heimwerkertexte*). Anhand unseres Trainingsmaterials haben wir zwei Sprachmodelle P_K und P_H trainiert, ersteres auf Grundlage von Rezepten, letzteres auf Grundlage von Heimwerkerinformationen. Liegt nun ein einzuordnender Text d vor, in dem viel von *Kartoffeln, Gewürzen* und *Garzeiten* die Rede ist, dann wird die enthaltene Wortfolge in P_K eine höhere Wahrscheinlichkeit haben als in P_H; $P(d|K)$ ist also höher als $P(d|H)$. Die höhere Wahrscheinlichkeit ist einfach auf die höhere Auftretenshäufigkeit der entsprechenden Wörter im K-gelabelten Trainingsmaterial zurückzuführen. Der Text d wird damit als Kochrezept klassifiziert werden.

Kapitel 13

Grammar Induction (Draft)

Unter Grammar Induction lässt sich allgemein das automatische Auffinden von Strukturen in Sprache verstehen. Es geht es darum, Sprachdaten nach gewählten Optimalitätskriterien zu segmentieren, beispielsweise in Morpheme-inheiten oder syntaktische Konstituenten. Hierzu gibt es diverse Ansätze, von denen hier zwei kurz vorgestellt werden sollen.

13.1 Minimum Description Length (MDL)

Hier wird Strukturierung der Daten als Komprimierungsproblem betrachtet. Es wird nach einer optimalen Komprimierung der Daten unter Ausnutzung der den Daten inhärenten Struktur gesucht. Die Güte eines Modells hängt ab von:

1. dem Kompressionsgrad der Daten D
2. der Größe des Modells M

Gesucht wird nach dem Modell M zur Sprachstrukturierung, das folgenden Ausdruck erfüllt:

$$\hat{M} = \arg\min_{M} [\mathrm{dl}(D|M) + \mathrm{dl}(M)], \tag{13.1}$$

wobei unter *dl* die *description length* zu verstehen ist. Ihre Messung hängt vom jeweiligen Verfahren ab. Hinter der Suche nach einem möglichst kleinen Modell (also nach einem niedrigen Wert für $\mathrm{dl}(M)$) steckt die Überlegung, das Modelle mit wachsender Größe zu stark an die Trainingsdaten adaptieren. Je mehr Parameter ein Modell besitzt, desto eher wird es nicht nur allgemein systematische sondern auch zufällige Zusammenhänge in den Trainingsdaten mitberücksichtigen. Solche Modelle können schlecht auf ungesehene Daten generalisieren, sind also im schlimmsten Fall für andere Sprachdaten als die, auf die sie trainiert wurden, unbrauchbar. Die bevorzugte Wahl kleinerer Modelle wird als *Occam's Razor* bezeichnet.

Beispiel In einer ersten Annäherung im Rahmen der automatischen morphologischen Strukturierung kann dl(M) gleich der Größe eines endlichen Automaten[1] zur morphologischen Segmentierung von Texten gesetzt werden ("Größe" meint Anzahl seiner Zustände), und dl($D|M$) gleich der Größe des Morphemvokabulars, also der Anzahl der Morphemtypes, die sich durch die Segmentierung des Texts durch M ergeben. Gesucht wird hier also ein möglichst kompakter Automat, der ein möglichst kleines Vokabular erzeugt. Eine Buchstabensegmentierung würde eine sehr kleine dl($D|M$) bewirken, nämlich die Anzahl der unterschiedlichen Buchstaben. Dafür wäre aber der Automat zur Segmentierung der Daten und damit dl(M) sehr groß. Bei einer Wortsegmentierung verhielte es sich genau andersherum. Die optimale Segmentierung liegt also irgendwo dazwischen.

Informationstheoretische Überlegungen Üblicherweise wird das **Bit** als Einheit der *description length* herangezogen. Gesucht wird dann nach einem Modell, dass einen Text so segmentiert, dass die Anzahl der Bits zu seiner Codierung minimiert wird. Über den Zusammenhang zwischen Wahrscheinlichkeit und Entropie, wie er in Kapitel 3 vorgestellt wurde, wissen wir, dass eine Erhöhung der Wahrscheinlichkeit eines Textes zu einer Abnahme der zu seiner Codierung nötigen Bitzahl führt. Die Segmentierung sollte also den Text so zerlegen, dass die Segmente im Durchschnitt eine möglichst hohe Wahrscheinlichkeit haben. Dies erreicht man durch Erzeugung einer möglichst geringen Anzahl von Segment-Types (vgl. obiges Beispiel) mit jeweils hoher Auftretenshäufigkeit.

Im nächsten Abschnitt wird ein konkretes Verfahren mit MDL-Bezug vorgestellt.

13.2 Sequitur

Sequitur (Nevill-Manning&Witten, 1997) komprimiert eine gegebene Zeichenfolge S, indem sie aus ihr eine hierarchische Struktur ableitet (genauer: eine kontextfreie Grammatik[2]), die S komplett beschreibt. Dies geschieht durch rekursives Ersetzen sich wiederholender Digramme (Teilstrings der Länge 2) durch non-terminale Symbole. Rekursiv meint, dass auch non-terminale Symbole als Teil von Digrammen wiederum zu weiteren non-terminalen Symbolen zusammengefasst werden können.

Zwei Prinzipien leiten den Algorithmus bei der Strukturierung der Daten:

[1] *Endliche Automaten* sind Netzwerke aus einer endlichen Anzahl von miteinander verbundenen Zuständen. Textsegmentierung funktioniert hier durch Übergang von einem Zustand in einen anderen unter Einlesen eines Teilstrings des Texts. Landet man nach Einlesen des letzten Symbols in einem sog. *finalen* Zustand, dann war die Segmentierung erfolgreich.

[2] *Kontextfreie Grammatiken* enthalten nur Regeln der Form '$A \longrightarrow *$' – iGgs. zu '$K_l A K_r \longrightarrow *$', d.h. A wird unabhängig vom Kontext K zu $*$.

1. **Digrammeindeutigkeit:** Jedes Digramm darf höchstens einmal vorkommen. Ansonsten werden alle seine Vorkommen durch ein neues non-terminales Symbol ersetzt.

2. **Symbolnützlichkeit:** Jedes non-terminale Symbol muss mindestens zweimal verwendet werden.

(1) sorgt für eine Komprimierung der Daten, (2) für eine Komprimierung des Modells. Strenggenommen ist Sequitur kein reines MDL-Verfahren, da hier trotz Komprimierung nicht die Minimierung des Ausdrucks in Gleichung 13.1 angestrebt wird. Dafür ist die Komplexität dieses Algorithmusses sehr gering bei $O(n)$.

Beispiel

1. Die Zeichenkette $S = abcdabcdefef$ wird buchstabenweise von links nach rechts eingelesen.

2. Bis *abcda* fällt keine Ersetzungsoperation an.

3. Bei *abcdab* wird das erste doppelt auftretende Digramm *ab* durch ein non-terminales Symbol *A* ersetzt: *abcdab* wird zu *AcdA* (**Digrammeindeutigkeit**).

4. Nach Einlesen des folgenden *c* tritt *Ac* doppelt auf: *AcdAc* wird zu *BdB*.

5. *BdBd* wird zu *EE*.

6. *EEefef* wird zu *EEFF*.

7. Jedes der auftretenden non-terminalen Symbole kommt mindestens zweimal vor. Damit ist hier das Prinzip der **Symbolnützlichkeit** nicht verletzt.

Die mit dieser Komprimierung einhergehende Kontextfreie Grammatik sieht folgendermaßen aus:

$$\begin{aligned} S &\longrightarrow EEFF \\ E &\longrightarrow abcd \\ F &\longrightarrow ef \end{aligned}$$

Literaturverzeichnis

[1] T. Andernach. A machine learning approach to the classification and prediction of dialogue utterances. In *NeMLaP-2*, Ankara, 1996.

[2] M.R. Brent, S.K. Murthy, and A. Lundberg. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proc. of the Fifth International Workshop on Artificial Intelligence and Statistics*, 1995.

[3] E. Charniak. *Statistical Language Learning.* MIT Press, Cambridge, Massachusetts, 1993.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society*, 39(1):1–21, 1977.

[5] W.A. Gale and G. Sampson. Good-turing frequency estimation without tears. *J. Quantitative Linguistics*, 2(3):217–237, 1995.

[6] J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27, 2001.

[7] P.D. Grünwald. *The Minimum Description Length Principle.* MIT press, Cambridge, MA, 2007.

[8] Z. Harris. *Methods in Structural Linguistics.* University of Chicago Press, Chicago, 1951.

[9] M.A. Hearst. Texttiling: A quantitative approach to discourse segmentation. Technical report, UCB:S2K-93-24, 1993.

[10] F. Jelinek. Markov source modeling of text generation. In *The Impact of Processing Techniques on Communications*, NATO ASI series. Dordrecht: M. Nijhoff, 1985.

[11] D. Jurafsky and J.H. Martin. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice-Hall, New Jersey, 2000.

[12] G. Leech, R. Garside, and M. Bryant. Claws4: The tagging of the British National Corpus. In *COLING-94*, pages 622–628, Kyoto, 1994.

[13] C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT, Cambridge, Massachusetts, 2001.

[14] M.P. Marcus, B. Santorini, and M.A. Marcikiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1995.

[15] M.A. Nascimento and A.C.R. da Cunha. An experiment stemming nontraditional text. In *SPIRE'98 Proceedings*, Santa Cruz de La Sierra, Bolivia, 1998.

[16] C.G. Nevill-Manning and I.H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *J. Artificial Intelligence Research*, 7:67–82, 1997.

[17] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

[18] U.D. Reichel. Improving data driven part-of-speech tagging by morphologic knowledge induction. In *Proc. AST Workshop*, Maribor, 2005.

[19] A. Schiller and S. Teufel. *Guidelines für das Tagging deutscher Textcorpora*, 1995.

[20] H. Schmid. Improvements in part-of-speech tagging with an application to german. In *EACL, SIGDAT*. ., Dublin, 1995.

[21] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[22] A. Voutilainen. A syntax-based part of speech analyser. In *Proc. of the Seventh Conference of the European Chapter of the Association for Computation al Linguistics*, pages 157–164, Dublin, 1995. Association for Computational Linguistics.