

Maximum-Likelihood-Schätzung von Wortkategorien mit Verfahren der kombinatorischen Optimierung

Studienarbeit im Fach Informatik

vorgelegt
von

Franz Josef Och

geboren am 2. November 1971 in Ebermannstadt

Angefertigt am

Lehrstuhl für Mustererkennung (Informatik 5)
Institut für Mathematische Maschinen und Datenverarbeitung
Friedrich-Alexander-Universität Erlangen-Nürnberg.

Betreuer: E. G. Schukat-Talamazzini, V. Fischer

Beginn der Arbeit: 1. November 1994

Abgabe der Arbeit: 17. Juli 1995

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien des Lehrstuhls für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts in Punkt 2.3.

Erlangen, den

.....

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Entwicklung von Verfahren zur automatischen Bestimmung disjunkter Kategoriensysteme für statistische Sprachmodelle. Ausgehend vom Bayes-Klassifikator ergeben sich mit dem Maximum-Likelihood-Ansatz und einem Kreuzvalidierungs-Ansatz (leave-one-out) zwei Optimierungskriterien. Für eine effiziente Implementierung im Rahmen von iterativ-optimierenden Verfahren ist es notwendig eine effiziente Auswertung dieser Kriterien zu bestimmen.

Es wird die Klasse der *iterativ-optimierenden Verfahren* vorgestellt und deren konkrete Ausprägungen stochastische Relaxation (Hill Climbing), simuliertes Ausfrieren (Simulated Annealing), Sinflutalgorithmus (Great Deluge Algorithm), Schwellwertakzeptanz (Threshold Accepting) und Record-To-Record Travel beschrieben. Auf diesen Verfahren baut die *multidirektionale Suche mit Beschneidung* auf, welche durch parallele Verarbeitung mehrerer iterativ-optimierender Verfahren effizienter gute Lösungen berechnet. Die *iterierte Zustandsraum-Reduktion* stellt ein problemspezifisches Verfahren dar, welches die gleichen Anteile einer Menge von ‘guten Kategoriensystemen’ dazu verwendet, eine Zustandsraum-Reduktion vorzunehmen.

Die objektorientierte Implementierung erfolgt in C++. Es werden umfangreiche Auswertungen der Leistungsfähigkeit der Optimierungsverfahren und der Optimierungskriterien durchgeführt. Es zeigt sich, daß mit diesem Ansatz eine kleinere Perplexität als mit einem manuell erstellten Kategoriensystem erzielt wird.

Inhaltsverzeichnis

1	Einleitung	3
2	Das Optimierungsproblem	7
2.1	Sprachmodelle	7
2.2	Optimierungskriterien	10
2.2.1	Einfaches Maximum-Likelihood-Kriterium	12
2.2.2	Maximum-Likelihood-Kriterium mit Kreuzvalidierung	14
2.3	Formulierung des Optimierungsproblems	17
2.4	Zusammenfassung	19
3	Die Optimierungsverfahren	20
3.1	Iterativ-optimierende Verfahren	20
3.1.1	Stochastische Relaxation (Hill Climbing, HC)	22
3.1.2	Simuliertes Ausfrieren (Simulated Annealing, SA)	23
3.1.3	Schwellwertakzeptanz (Threshold Accepting, TA)	26
3.1.4	Record-To-Record Travel (RRT)	27
3.1.5	Sinifluralgorithmus (Great Deluge Algorithm, GDA)	28
3.2	Multidirektionale Suche mit Beschneidung	29
3.3	Iterierte Zustandsraum-Reduktion	31
3.4	Implementierung	36
3.5	Zusammenfassung	38
4	Implementierung des Optimierungsproblems	40
4.1	Auswertung der Optimierungskriterien	40
4.2	Initialisierung	45
4.3	Nachbarschaft	47
4.4	Endkriterium	50
4.5	Zusammenfassung	50
5	Vergleich der Optimierungsverfahren	52
5.1	Parameter-Einstellung	52
5.2	Vergleich der erreichten Kosten	57
5.3	Zusammenfassung	60

6	Ergebnisse auf verschiedenen Korpora	61
6.1	Testumgebung	61
6.2	Auswertung	62
7	Zusammenfassung und Ausblick	69
7.1	Zusammenfassung	69
7.2	Ausblick	70
A	Ergebnisse für das Intercity-Korpus	72
A.1	Ein Kategoriensystem	73
A.2	Ergebnisse einer Zustandsraum-Reduktion	76
B	Diagramme	79
C	Programm-Manual	91
	Literaturverzeichnis	95

Kapitel 1

Einleitung

Der spanische Jesuit Pedro Bermudo (1610–1648) teilte die Liste der Wörter in jeder Sprache in 44 Grundkategorien ein. Da sich diese Arbeit ebenfalls mit einer Einteilung der Wörter in Kategorien beschäftigt, ist es interessant, sein Ergebnis näher zu betrachten:

1. Elemente. 2. Himmlische Größen. 3. Geistige Größen. 4. Weltliche Größen. 5. Kirchliche Größen. 6. Kunstgriffe. 7. Instrumente. 8. Affekte. 9. Religion. 10. Sakramentale Konfession. 11. Gericht. 12. Armee. 13. Medizin. 14. Häßliche Tiere. 15. Vögel. 16. Reptilien und Fische. 18. Gerätschaften. 19. Speisen. 20. Getränke und andere Flüssigkeiten. 21. Kleider. 22. Seidengewebe. 23. Wollstoffe. 24. Segeltücher und andere Textilien. 25. Nautica und Aromen. 26. Metalle und Münzen. 27. Diverse Artefakte. 28. Steine. 29. Juwelen. 30. Bäume und Früchte. 31. Öffentliche Orte. 32. Maße und Gewichte. 33. Zahlen. 34. Zeit. 35–42. Nomina, Adjektive, Adverbien und so weiter. 43. Personen. 44. Wanderschaft.

In [Eco94] führt Umberto Eco diese Liste als Musterbeispiel für eine nicht logische Einteilung in Kategorien an, und behauptet, daß jede mögliche Liste von Kategorien inkongruent (also unlogisch) sein muß. Ob dies auch auf die in dieser Arbeit gefundenen Kategorien zutrifft, wird zu prüfen sein.

In dieser Arbeit sollen Wortkategorien mit dem Zweck eingesetzt werden, dem Computer das Verstehen gesprochener Sprache zu erleichtern. Diese Kategorien werden zur Bildung des sogenannten Sprachmodells eingesetzt. Ein statistisches Sprachmodell ordnet jeder Wortfolge eine Wahrscheinlichkeit zu und unterscheidet damit sinnlose von sinnvollen Wortfolgen. Mit dieser Unterscheidung kann ein spracherkennendes System fehlerfreier und schneller arbeiten. Ein Sprachmodell sollte beispielsweise *“das Frau”*, *“das Tür”* und *“das Hund”* qualitativ unterscheiden können von *“die Frau”*, *“die Tür”* und *“der Hund”*.

Es wäre sehr aufwendig alle sinnvollen Wortfolgen explizit zu speichern. Im Wortgraphen in Bild 1.1 (oben) bildet jede Wortfolge, die in Richtung der Pfeile gebildet wird, einen sinnvollen deutschen Satz. Es ist jede Kombination eines der Namen mit einem der Verben möglich. Zur Verringerung des Aufwandes können nun Kategorien eingesetzt wer-

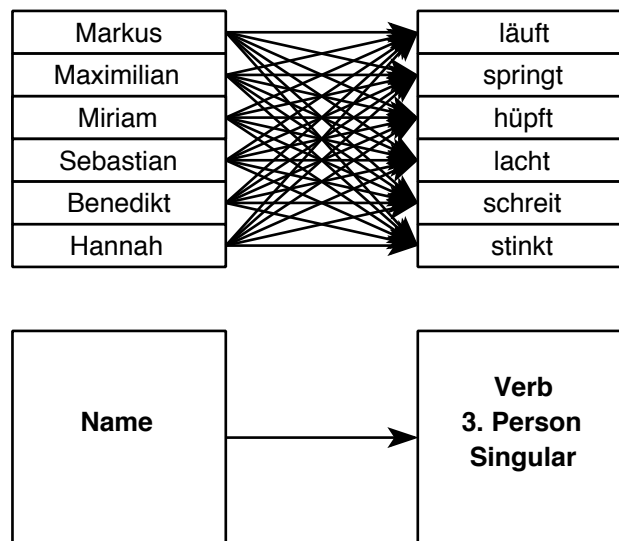


Bild 1.1: Verringerung des Aufwandes durch Bildung von Kategorien.

den (Bild 1.1 unten). Auf den ersten Blick fällt die große Einsparung an Pfeilen auf. In größeren praktischen Sprachmodellen ist diese Einsparung noch stärker.

Gute Kategorien (im Sinne der maschinellen Spracherkennung) sind dadurch gekennzeichnet, daß sie ‘ähnliche’ Wörter zusammenfassen. Häufig und mit Erfolg werden die linguistisch motivierten POS- (‘parts of speech’-) Kategorien eingesetzt, in die syntaktische Wortkategorien wie Wortart, Kasus, Numerus und Genus einfließen ([Sch94b]). Beispielsweise folgen dem Artikel “*das*” vermutlich Adjektive (“*das große*”) und sächliche Substantive (“*das Tier*”), jedoch keine weiteren Artikel (“*das der*”) oder feminine Substantive (“*das Frau*”).

Man kann sich jedoch auch vorstellen, daß semantische Kategorien eingesetzt werden können. Beispielsweise werden bestimmte Adjektive nicht zusammen mit bestimmten Substantiven verwendet, wie in “*aggressives Telefon*”. Man kann erwarten, daß auf ein Wort aus der semantischen Kategorie, in der “*aggressives*” enthalten ist, kein Substantiv folgen kann, das im allgemeinen nicht die Eigenschaft der Aggressivität besitzt. Würden nur syntaktische Kategorien berücksichtigt, so könnte man dies nicht handhaben.

Interessiert man sich für Kategorien, die für die maschinelle Spracherkennung geeignet sind, so helfen einem sowohl semantische als auch syntaktische Kategorien, deren praktische Bestimmung jedoch aufwendig und schwierig ist. Bei syntaktischen Kategorien geschieht dies meist durch den aufwendigen Vorgang des ‘*tagging*’. Dabei wird in einem großen Korpus an jedem Wort seine grammatische Kategorie von einem menschlichen Experten vermerkt. Die Erstellung semantischer Kategorien stellt sich noch ungleich schwieriger dar. Man wird erwarten, daß zwei Wörter in derselben semantischen Kategorie sind, wenn sie synonym verwendet werden können. Beispielsweise bedeuten “*hübsch*” und “*attraktiv*” in gewissen Grenzen dasselbe und sind austauschbar.

Ein umfassendes System zur Verarbeitung semantischer Kategorien müßte neben regionalen, domänenspezifischen und zeitlichen Unterschieden im Gebrauch der Wörter auch berücksichtigen, daß die menschliche Sprache *unscharf* ist. Dies wirkt sich insbesondere bei der Bildung von Kategorien aus, da hier das Problem besteht, daß die Synonyme eines Wortes nicht wieder Synonyme des ursprünglichen Wortes sein müssen. Beispielsweise ist jedes Wort in der folgenden Kette Synonym des vorhergehenden bzw. nachfolgenden Wortes, jedoch sind *„zuverlässig“* und *„unzuverlässig“* offensichtlich Antonyme ([Dew87]):

zuverlässig → *standhaft* → *hartnäckig* → *eigensinnig* → *unberechenbar* → *unzuverlässig*

In einer Untersuchung anhand des ‘New Collins Thesaurus’ wurden mehrere tausend solcher Ketten erzeugt ([Dew87]), und damit gezeigt, daß dies ein allgemeines Problem darstellt.

Sollte schließlich ein Kategoriensystem durch einen menschlichen Experten für eine begrenzte Domäne entwickelt worden sein, dann ist dies mit allerlei Schwierigkeiten verbunden. Es ist unwahrscheinlich, daß bei einem Vokabular von mehreren tausend Wörtern das Kategoriensystem fehlerfrei ist, also kein Wort vom menschlichen Experten durch Unachtsamkeit in die falsche syntaktische/semantische Kategorie eingeordnet wurde. Weiterhin ist es notwendig bei einer Erweiterung des Vokabulars oder der Ausdehnung auf eine neue Domäne die Kategoriensysteme anzupassen. Sollte der Autor des ursprünglichen Systems nicht mehr verfügbar sein, so muß ein anderer sich in das gewählte Kategoriensystem einarbeiten, um es sinnvoll erweitern zu können.

Der Aufwand zur Bestimmung und Pflege syntaktischer bzw. semantischer Kategorien kann vermieden werden, da derartige Kategorien — im Rahmen der Spracherkennung — nicht direkt von Interesse sind. Hier interessiert es, die Qualität des Spracherkenners zu verbessern, und die Tatsache, daß syntaktische/semantische Kategorien (erfolgreich) eingesetzt werden, deutet nur darauf hin, daß derartige Kategorien geeignet sind dieses Ziel zu verfolgen. Die Qualität des Spracherkenners läßt sich jedoch formal beschreiben und unter Anwendung von statistischen Methoden und geeigneten Vereinfachungen lassen sich formale Kriterien ermitteln, welche die Güte eines Kategoriensystems beschreiben. Diese Kriterien werden sich auch als praktisch (wenn auch nur näherungsweise) optimierbar erweisen. Wenn man die im Sinne des Optimierungskriteriums optimalen Kategorien als die ‘wahren Kategorien’ bezeichnet, dann könnte man auch sagen, daß die hier gefundenen Kategoriensysteme (im Gegensatz zu denen von Pedro Bermudo) ‘fast logisch’ und ‘fast kongruent’ sind und es nur eine Frage des Aufwandes ist, die ‘wahren Kategorien’ zu finden, die es im Sinne des Optimierungskriteriums durchaus gibt.

Die konkrete Motivation für diese Arbeit entstand im Rahmen des sprachverstehenden Dialogsystems EVAR, das an den Universitäten Erlangen-Nürnberg und Bielefeld entwickelt wird. Die Aufgabe von EVAR ist die Führung eines informationsabfragenden Dialogs über das deutsche InterCity-Zugsystem ([Eck92, Mas94]). Das System gliedert sich in die Module *Worterkennung*, *Syntax*, *Semantik*, *Pragmatik* und *Dialog*. In der Worterkennung wurde bisher ein manuell erstelltes Kategoriensystem verwendet ([Wit92]). Aus der Notwendigkeit dieses Kategoriensystem stetig zu pflegen und zu erweitern, entstand die Idee für diese Arbeit, nämlich die Erstellung eines automatischen Verfahrens zur Bestim-

mung von Wortkategorien.

Die Arbeit gliedert sich in sieben Kapitel. In Kapitel 2 wird das Optimierungsproblem vorgestellt. Ausgehend von der Fundamentalformel der automatischen Spracherkennung gelangt man zu formalen Optimierungskriterien. Die Auswertung dieser Kriterien erfolgt mittels allgemeiner und problemspezifischer Optimierungsverfahren, die in Kapitel 3 vorgestellt werden. Für eine gute Implementierung ist es erforderlich, sich besondere Gedanken über die Effizienz zu machen. Dies geschieht in Kapitel 4, in dem eine effiziente Auswertung der Kriterien vorgestellt wird und außerdem verschiedene Einstellungsmöglichkeiten für den Verlauf der Optimierung verglichen werden. In Kapitel 5 werden die Optimierungsverfahren anhand der Qualität der erreichten Lösungen verglichen. In Kapitel 6 werden Auswertungen an verschiedenen Korpora durchgeführt, anhand derer die Eigenschaften des Problems und der hier vorgenommenen Lösung sichtbar werden. Es folgt in Kapitel 7 eine Zusammenfassung von Vorgehensweise und Ergebnissen dieser Arbeit und Vorschläge für Erweiterungs- und Einsatzmöglichkeiten.

Kapitel 2

Das Optimierungsproblem

In diesem Kapitel soll das Optimierungsproblem — die Bildung optimaler Kategorien für Bigramm-Sprachmodelle — formalisiert werden.

Ausgehend vom Bayes-Klassifikator wird mit dem Maximum-Likelihood-Ansatz ein Optimierungskriterium für disjunkte Kategoriensysteme ermittelt. Dieses kennzeichnet die Qualität eines Kategoriensystems für das Trainingskorpus und kann deshalb nur mit Nebenbedingung auf unabhängige Korpora verallgemeinern. Es wird ein weiteres Optimierungskriterium mit Kreuzvalidierung (leave-one-out) ermittelt, welches das Auftreten von unbeobachteten Ereignissen simuliert.

2.1 Sprachmodelle

In der maschinellen Spracherkennung besteht das Ziel darin, aus akustischen Eingabedaten \mathbf{X} diejenige Wortfolge $\mathbf{w} = w_1 \dots w_m$ zu bestimmen, die gesprochen wurde. Um eine minimale Fehlerrate zu erhalten, muß man den Bayes-Klassifikator verwenden ([Sch94b, Ney92]). Dieser entscheidet sich für diejenige Wortfolge $\hat{\mathbf{w}}$, welche die a posteriori-Wahrscheinlichkeit maximiert:

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w}} P(\mathbf{w}|\mathbf{X}) = \operatorname{argmax}_{\mathbf{w}} \frac{P(\mathbf{w}) \cdot P(\mathbf{X}|\mathbf{w})}{P(\mathbf{X})} \quad (2.1)$$

Diese Formel wird auch als Fundamentalformel der Spracherkennung bezeichnet. Da über alle möglichen Wortfolgen maximiert wird, kann der konstante Term $P(\mathbf{X})$ vernachlässigt werden. Es müssen also zwei Wahrscheinlichkeiten modelliert werden: die akustische Wahrscheinlichkeit $P(\mathbf{X}|\mathbf{w})$ und die linguistische Wahrscheinlichkeit $P(\mathbf{w})$. Für erstere muß ein *akustisch-phonetisches Sprachmodell* entwickelt werden. Dies geschieht meist unter Verwendung von Hidden-Markov-Modellen ([Rab93, Sch94b]). Hier interessiert jedoch die linguistische Wahrscheinlichkeit $P(\mathbf{w})$. Es ist die Aufgabe des *linguistischen Sprachmodells* diese zu bestimmen. Ganz allgemein kann man die Wahrscheinlichkeit $P(\mathbf{w})$ für das Auftreten einer endlichen Wortfolge \mathbf{w} , im weiteren *Satz* genannt, als Produkt über bedingte

Wahrscheinlichkeiten formulieren:

$$P(\mathbf{w}) = P(w_1|\$) \cdot P(\$|\mathbf{w}) \cdot \prod_{i=2}^m P(w_i|w_1 \dots w_{i-1}) \quad (2.2)$$

Dabei wird das Sonderzeichen \$ zur Beschreibung der Satzanfangs- und Satzende-Wahrscheinlichkeiten eingeführt. Es entspricht $P(w_1|\$)$ der Wahrscheinlichkeit, daß ein Satz mit w_1 beginnt und $P(\$|\mathbf{w})$ der Wahrscheinlichkeit, daß die Wortfolge \mathbf{w} einen vollständigen Satz bildet. Es ist jedoch im allgemeinen nicht möglich, die Wahrscheinlichkeiten $P(w_i|w_1 \dots w_{i-1})$ praktisch zu bestimmen, da hierzu zu viele Parameter geschätzt werden müssen. Bei einer Vokabulargröße von $|\mathcal{W}|$ und einer auf m begrenzten Satzlänge sind $|\mathcal{W}|^m + |\mathcal{W}|^{m-1} + \dots + |\mathcal{W}|^1$ Parameter zu schätzen, woraus sich für ein Vokabular von 50 000 Wörtern und einer maximalen Satzlänge von 20 mehr als $9.5 \cdot 10^{93}$ zu schätzende Parameter ergeben. Eine Möglichkeit, die Zahl der Parameter zu reduzieren, besteht darin, Bigramm-Sprachmodelle einzusetzen. Hierbei wird die Auftrittswahrscheinlichkeit eines Wortes w_i in Abhängigkeit des vorhergehenden Wortes w_{i-1} approximiert:

$$P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-1}) \quad (2.3)$$

Man kann diese Approximation dadurch rechtfertigen, daß der Einfluß des letzten Wortes w_{i-1} auf w_i normalerweise viel größer ist, als der Einfluß früherer Wörter ([Sch94b]). Um die Satzanfangs- und Satzende-Wahrscheinlichkeiten im weiteren nicht explizit mitführen zu müssen, wird $w_0 = w_{m+1} = \$$ gesetzt. Die Wahrscheinlichkeit für das Auftreten eines Satzes berechnet sich nun durch:

$$P(\mathbf{w}) = P(w_1|\$) \cdot P(\$|w_m) \cdot \prod_{i=2}^m P(w_i|w_{i-1}) = \prod_{i=1}^{m+1} P(w_i|w_{i-1}) \quad (2.4)$$

Dabei entspricht $P(w_1|\$)$ der Wahrscheinlichkeit, daß ein Satz mit w_1 beginnt, und $P(\$|w_m)$ der Wahrscheinlichkeit, daß ein Satz mit dem Wort w_m endet.

Die Wahrscheinlichkeiten $P(w_i|w_{i-1})$ erhält man durch Abzählen der Bigramme in einem möglichst großen Korpus und die Approximation der Wahrscheinlichkeiten anhand der relativen Häufigkeiten. Man erhält jedoch wieder ein Schätzproblem ([Jel90]), da normalerweise noch zu viele Parameter einer zu geringen Zahl von Ereignissen gegenüberstehen. Bei einer Vokabulargröße von $|\mathcal{W}|$ gibt es $|\mathcal{W}|^2 - 1$ verschiedene Bigramme.¹ Dies ergibt bei einem Vokabular von 50 000 Wörtern etwa $2.5 \cdot 10^9$ zu schätzende Wahrscheinlichkeiten. Selbst in großen Korpora mit einigen Millionen Wörtern tritt jedoch nur ein Bruchteil dieser Bigramme auf.

Den hier abgeschätzten Aufwand kann man reduzieren, wenn man die Menge der Wörter in *Kategorien* strukturiert. Dabei unterscheidet man zwei prinzipiell verschiedene Typen von Kategoriensystemen:

1. **Disjunkte Kategoriensysteme:** Hierbei wird die Menge der Wörter in *Äquivalenzklassen* aufgeteilt. Dies wird auch durch die eingangs beschriebene Strukturierung des Vokabulars in linguistische Kategorien motiviert. Es werden dadurch die

¹Die -1 in der Zahl der Bigramme ergibt sich dadurch, daß das Bigramm $(\$, \$)$ nicht auftreten kann.

Statistiken aller Wörter einer Kategorie miteinander “verklebt”. Man muß nur noch die Bigramm-Wahrscheinlichkeiten zwischen Kategorien schätzen und kann die Zahl der zu bestimmenden Parameter reduzieren. Bei einer Kategorienzahl von $|\mathcal{C}|$ gibt es $|\mathcal{C}|^2 - 1$ verschiedene Kategorien-Bigramme. Dies ergibt bei 100 Kategorien nur etwa 10^4 Parameter.

Es wird nun die Wahrscheinlichkeit eines Satzes unter Verwendung disjunkter Kategoriensysteme formuliert. Es bezeichne $C(w)$ die Funktion, die einem Wort w seine Kategorie $c \in \mathcal{C}$ zuordnet. Dabei ist \mathcal{C} die gewählte Partition des Vokabulars \mathcal{W} , welche das Kategoriensystem bildet. Das Satzanfangs- und Satzendesymbol $\$$ soll dabei als spezielles Symbol erhalten bleiben, also nicht mit anderen Wörtern zusammen in einer Kategorie auftreten. Man approximiert also:

$$P(w_i|w_{i-1}) \approx P(C(w_i)|C(w_{i-1})) \cdot P(w_i|C(w_i)) \quad (2.5)$$

Dabei ist $P(c|c') = P(C(w_i)|C(w_{i-1}))$ die Wahrscheinlichkeit für Kategorien-Bigramme und $P(w|c) = P(w_i|C(w_i))$ die Wahrscheinlichkeit der Zugehörigkeit des Wortes w zur Kategorie c . Insgesamt müssen unter Berücksichtigung der Stochastizitätsbedingung $|\mathcal{C}|^2 - 1 - |\mathcal{C}|$ Bigramm-Wahrscheinlichkeiten und $|\mathcal{W}| - |\mathcal{C}|$ Wahrscheinlichkeiten der Zugehörigkeit eines Wortes zu einer Kategorie geschätzt werden. Man erhält nun für die Wahrscheinlichkeit eines Satzes durch Einsetzen von Gleichung (2.5) in Gleichung (2.4):

$$P(\mathbf{w}|\mathcal{C}) = \prod_{i=1}^{m+1} P(C(w_i)|C(w_{i-1})) \cdot P(w_i|C(w_i)) \quad (2.6)$$

2. **Nicht-disjunkte Kategoriensysteme:** Die Forderung der Einteilung des Vokabulars in feste Kategorien kann gelockert werden, indem man ermöglicht, daß ein Wort in mehr als einer Kategorie enthalten sein kann. Dies wird motiviert durch das Phänomen, daß ein Wort (z. B. ‘Essen’) verschiedenen linguistischen Kategorien zugeordnet werden kann (substantiviertes Verb, Eigennamen). Die Wahrscheinlichkeit eines Satzes ist dadurch natürlich erheblich aufwendiger zu berechnen, da nun über alle möglichen Folgen von Kategorien summiert werden muß:

$$P(\mathbf{w}|\mathcal{C}) = \sum_{\mathbf{c} \in \mathcal{C}^m} \left(\prod_{i=1}^{m+1} P(c_i|c_{i-1}) \cdot P(w_i|c_i) \right) \quad (2.7)$$

Es ergibt sich, daß ein nicht-disjunktes Kategoriensystem in dieser Form nichts anderes als ein diskretwertiges Markovmodell ist. Dabei entsprechen die Zustände den Kategorien und die Ausgabesymbole den Wörtern ([Sch94b], S. 210). Für dieses Sprachmodell sind nun erheblich mehr Parameter zu schätzen, da jedes Wort in jeder Kategorie mit gewisser Wahrscheinlichkeit sein kann. Für jedes Wort im Vokabular müssen $|\mathcal{C}| - 1$ Parameter geschätzt werden. Damit liefern nicht-disjunkte im

Gegensatz zu disjunkten Kategoriensystemen erheblich aufwendigere und schwieriger zu schätzende Sprachmodelle.

Für nicht-disjunkte Kategoriensysteme existiert ein Verfahren, welches diese automatisch bestimmen kann ([Kuh94a]). Dieses Verfahren basiert auf dem für diskretwertige Markovmodelle bekannten Baum-Welch-Algorithmus. Es stellt sich jedoch als sehr aufwendig heraus, da sich die volle Komplexität des Baum-Welch-Algorithmus ($\mathcal{O}(|\mathcal{C}|^2 \cdot |\mathcal{W}|)$) ergibt. Es hat sich auch herausgestellt, daß die Ergebnisse hinter denen eines manuell erstellten Kategoriensystems zurückbleiben.

Daher beschränkt sich diese Arbeit ausschließlich auf die Behandlung von disjunkten Kategoriensystemen, welche im folgenden einfach Kategoriensysteme genannt werden. Es sollen nun formale Kriterien für die Qualität derartiger Kategoriensystems entwickelt werden, mittels derer dann ein automatisches Verfahren gute Kategorien bilden kann.

2.2 Optimierungskriterien

Was ist ein ‘gutes Sprachmodell’ für die maschinelle Spracherkennung? Man wird ein Sprachmodell dann als ‘gut’ bezeichnen, wenn es die Fehlerrate des Spracherkenners minimiert. Eine direkte Minimierung dieser Größe ist jedoch sehr aufwendig ([Sch94b]). Ein häufig verwendeter Ersatz ist der *Maximum-Likelihood-Ansatz*, durch den ein Sprachmodell dann als ‘gut’ bezeichnet wird, wenn es einem repräsentativen Trainingskorpus eine hohe Wahrscheinlichkeit zuordnet. Somit muß das in diesem Sinne optimale Kategoriensystem \mathcal{C}' die Wahrscheinlichkeit aus Gleichung (2.6) maximal werden lassen:

$$\mathcal{C}' = \operatorname{argmax}_{\mathcal{C}} P(\mathbf{w}|\mathcal{C}) \quad (2.8)$$

Äquivalent hierzu ist die Minimierung der Perplexität ([Jar93b]).² Die Perplexität ([Jel82]) einer endlichen Wortfolge \mathbf{w} ist formal gegeben durch:

$$\mathcal{P}_x = P(\mathbf{w})^{-\frac{1}{m}} = 1 / \sqrt[m]{P(\mathbf{w})} \quad (2.9)$$

Die Perplexität kann anschaulich interpretiert werden als die mittlere Zahl der Worte, die eine Wortfolge fortsetzen können und stellt damit ein Maß für die Schwierigkeit dar, die einem Spracherkenner aus dem Sprachmodell erwächst. Eine Verringerung der Perplexität hat nach dieser Interpretation direkt eine Verringerung der Fehlerrate zur Folge. Durch Gleichung (2.9) ist sofort ersichtlich, daß eine Maximierung von $P(\mathbf{w}) = P(\mathbf{w}|\mathcal{C})$ einer Minimierung von \mathcal{P}_x entspricht.

Inwieweit kann man erwarten, daß der Maximum-Likelihood-Ansatz bzw. die Minimierung der Perplexität eine Verbesserung der Erkennungsleistung erwarten lassen kann? Im Sinne der Fundamentalformel der Spracherkennung (Gleichung (2.1)) erhält man die minimale Fehlerrate durch Maximierung der a posteriori-Wahrscheinlichkeit, also des Produktes

²Ein äquivalentes Optimierungskriterium erhält man durch Maximierung der mittleren wechselseitigen Information zwischen $C(w_i)$ und w_{i+1} ([Jel90]).

aus linguistischer und akustischer Wahrscheinlichkeit. Es ist also möglich, das linguistische und das akustische Sprachmodell unabhängig voneinander zu entwickeln. In der Praxis werden allerdings Vereinfachungen vorgenommen, die für eine wechselseitige Beeinflussung sorgen.

- Modelle für linguistische Wahrscheinlichkeit und akustische Wahrscheinlichkeit sind nicht korrekt: Diese Wahrscheinlichkeiten werden immer in vereinfachten Modellen bestimmt. Für die linguistische Wahrscheinlichkeit $P(\mathbf{w})$ wird hier ein Bigramm-Modell mit disjunkten Kategorien verwendet. Für die akustische Wahrscheinlichkeit $P(\mathbf{X}|\mathbf{w})$ werden meist Hidden-Markov-Modelle verwendet, die ebenfalls kein korrektes Modell darstellen. Dadurch können sich ungewollte Zusammenhänge zwischen linguistischer und akustischer Wahrscheinlichkeit ergeben und die zu maximierende a posteriori-Wahrscheinlichkeit aus der Fundamentalformel zerfällt nicht in zwei getrennt zu behandelnde Wahrscheinlichkeiten.
- Modellparameter werden nicht korrekt bestimmt: Nachdem die Modellstruktur festgelegt ist, gilt es die Parameter zu bestimmen. Die Modellparameter ergeben sich immer aus einer endlichen Stichprobe und können somit nicht die ‘wahren’ Wahrscheinlichkeiten wiedergeben, die nur asymptotisch für $m \rightarrow \infty$ angenähert werden können. Es erfolgt also eine Überanpassung der Modellparameter an die Lernstichprobe ([Pau90]). Dies führt beispielsweise dazu, daß in der Trainingsstichprobe normalerweise nicht alle möglichen Ereignisse auftreten und diesen *ungesehenen Ereignissen* — in Ermangelung besseren Wissens — eine konstante Wahrscheinlichkeit zugeordnet werden muß.

Man kann also nicht erwarten, daß der Maximum-Likelihood-Ansatz auch eine minimale Fehlerrate liefert, da in der Spracherkennung nichtkorrekte Wahrscheinlichkeits-Modelle mit nichtkorrekten Parameter-Einstellungen verwendet werden. Es ist also möglich, daß eine Verbesserung der Parameter des linguistischen Modells (im Sinne von Gleichung (2.8)) einer Verschlechterung der Erkennungsleistung entspricht.

Man könnte nun versuchen die Modelle zu verbessern und die Trainingsstichproben zu vergrößern, um sich den ‘wahren’ Wahrscheinlichkeiten anzunähern. Dies wird jedoch seine Grenzen haben, da komplexere Modelle auch mehr Parameter besitzen, die zu große Trainingsstichproben erfordern würden. Eine weitere Möglichkeit besteht darin, daß die mittlere wechselseitige Information zwischen Eingangssignal und gesprochener Wortfolge unter Variation der freien Parameter des akustischen und des linguistischen Modells optimiert wird, wovon man sich erhofft, daß dies bessere Ergebnisse als der Maximum-Likelihood-Ansatz aus Gleichung (2.8) liefert ([Bah86]).

Der Maximum-Likelihood-Ansatz für die linguistische Wahrscheinlichkeit ist also eine vereinfachende Annahme, damit aber allgemeiner, als eine Minimierung der Fehlerrate für einen Spracherkenner. Das so ermittelte Sprachmodell ist genauso in der maschinellen Texterkennung (OCR) oder bei der Syntaxprüfung eines Textverarbeitungssystems einsetzbar. In den folgenden Abschnitten wird es gelingen, mit dem Maximum-Likelihood-Ansatz zu

Optimierungskriterien zu gelangen, die sich als praktisch auswertbar erweisen, wohingegen es fraglich ist, ob sich aus einem Ansatz, welcher direkt die Fehlerrate minimiert, ebenfalls handhabbare Optimierungskriterien bestimmen lassen ([Sch94b]).

2.2.1 Einfaches Maximum-Likelihood-Kriterium

Es soll nun die Gleichung (2.6) so formuliert werden, daß sie effizient anhand eines Trainingskorpus ausgewertet und die Maximierung in Gleichung (2.8) durchgeführt werden kann ([Kne91]). Dabei soll nicht gefordert werden, daß das Trainingskorpus $w_1 \dots w_m$ nur aus einem Satz bestehen soll. Im Fall, daß die Sätze $w_{1,1} \dots w_{1,m_1}$ und $w_{2,1} \dots w_{2,m_2}$ das Trainingskorpus bilden, ergibt sich die Wahrscheinlichkeit für beide Sätze als Produkt der Wahrscheinlichkeiten für jeden einzelnen Satz:

$$\begin{aligned} P(w_{1,1} \dots w_{1,m_1}) \cdot P(w_{2,1} \dots w_{2,m_2}) &= P(w_{1,1}|\$) \cdot P(\$|w_{1,m_1}) \cdot \prod_{i=2}^m P(w_{1,i}|w_{1,i-1}) \\ &\quad \cdot P(w_{2,1}|\$) \cdot P(\$|w_{2,m_2}) \cdot \prod_{i=2}^m P(w_{2,i}|w_{2,i-1}) \end{aligned} \quad (2.10)$$

Einen identischen Ausdruck für die Wahrscheinlichkeit erhält man, wenn man die Sätze durch den Satzbegrenzer \$ konkatentiert: $w_1 \dots w_m = w_{1,1} \dots w_{1,m_1} \$ w_{2,1} \dots w_{2,m_2}$. Man kann also ohne Beschränkung der Allgemeinheit die Gleichung (2.6) maximieren. Man muß nur berücksichtigen, daß das Sonderzeichen \$ auch innerhalb des Korpus (mit der Funktion des Satzbegrenzers) vorkommen kann.

Ersetzt man in Gleichung (2.6) $P(c|c')$ und $P(w|c)$ durch die relativen Häufigkeiten (aus dem Trainingskorpus), so erhält man die Näherung:

$$\hat{P}(\mathbf{w}|C) = \prod_{i=1}^{m+1} \frac{n(C(w_{i-1}), C(w_i))}{n_1(C(w_{i-1}))} \cdot \frac{n_2(w_i)}{n_2(C(w_i))} \quad (2.11)$$

Dabei ist $n(c, c')$ die Häufigkeit des Kategorien-Bigramms (c, c') . Die Häufigkeiten $n_2(w)$, $n_1(c)$ und $n_2(c)$ beschreiben, wie oft das Wort w bzw. die Kategorie c an erster bzw. an zweiter Stelle in einem Bigramm auftritt. Da jedes Wort einschließlich des Satzbegrenzers \$ gleich oft an erster und an zweiter Stelle in einem Bigramm auftritt, gilt $n_1(w) = n_2(w)$ bzw. $n_1(c) = n_2(c)$ und man kann eine Zählfunktion $n(w) := n_1(w)$ bzw. $n(c) := n_1(c)$ einführen.³

Durch die Bildung des negativen Logarithmus wird aus dem Produkt (2.11) eine Summe und aufgrund der Vorzeichenumkehr ergibt sich, daß diese Gleichung minimiert werden muß. Mit $n(C(w_0)) = n(C(\$)) = n(C(w_{m+1}))$ gilt:

³Die Tatsache, daß man eine Zählfunktion $n(\cdot)$ einführen kann, ergibt sich daraus, daß die Bigramme alle einem fortlaufenden Korpus entnommen werden, der außerdem mit dem Sonderzeichen \$ umschlossen wird. Wenn dies nicht der Fall ist, so muß mit zwei Zählfunktionen $n_1(\cdot)$ und $n_2(\cdot)$ gerechnet werden.

$$-\log(\hat{P}(\mathbf{w}|\mathcal{C})) = -\sum_{i=1}^{m+1} \log\left(\frac{n(C(w_{i-1}), C(w_i))}{n(C(w_{i-1}))} \cdot \frac{n_2(w_i)}{n(C(w_i))}\right) \quad (2.12)$$

$$\begin{aligned} &= \sum_{i=1}^m \log(n(C(w_i))) + \sum_{i=2}^{m+1} \log(n(C(w_i))) \\ &\quad - \sum_{i=1}^{m+1} \log(n(C(w_{i-1}), C(w_i))) - \sum_{i=1}^{m+1} \log(n(w_i)) \end{aligned} \quad (2.13)$$

Da bei der Minimierung nur die Kategorienzugehörigkeiten variiert werden, kann der konstante Anteil $\sum_{i=1}^{m+1} \log(n(w_i))$ vernachlässigt werden und man erhält das Optimierungskriterium ML:

$$\text{ML}(\mathcal{C}) = 2 \sum_{i=1}^{m+1} \log(n(C(w_i))) - \sum_{i=1}^{m+1} \log(n(C(w_{i-1}), C(w_i))) \quad (2.14)$$

Es tritt für eine feste Kategorie c der Summand $\log(n(c))$ genau $n(c)$ mal auf. Entsprechend tritt $\log(n(c, c'))$ genau $n(c, c')$ mal auf. Man kann also die obige Summe umsortieren, über Kategorien anstatt über Wörter im Korpus summieren, und erhält die schließliche Form des Optimierungskriteriums ML zu:

$$\text{ML}(\mathcal{C}) = 2 \sum_{c \in \mathcal{C}} n(c) \log(n(c)) - \sum_{c, c' \in \mathcal{C}} n(c, c') \log(n(c, c')) \quad (2.15)$$

Man beachte, daß in dieser Formulierung des Optimierungskriteriums vom ursprünglichen Trainingskorpus nur noch die Statistiken über Häufigkeiten von Kategorien und Kategorien-Bigrammen übriggeblieben sind, die Komplexität der Auswertung somit also nicht von der Zahl der Wörter des Trainingskorpus abhängt.

Es soll nun noch angegeben werden, wie sich das Kriterium verändert, wenn der Maximum-Likelihood-Ansatz ohne Berücksichtigung der Satzanfangs- und Satzende-Wahrscheinlichkeiten angewendet wird. In diesem Fall kann man das Trainingskorpus als Multimenge von Bigrammen betrachten (siehe auch Abschnitt 2.2.2). Es ist dann nicht mehr möglich eine einheitliche Zählfunktion $n(w)$ für $n_1(w)$ und $n_2(w)$ einzuführen, da $n_1(w) \neq n_2(w)$. Es ergibt sich:

$$\text{ML}^*(\mathcal{C}) = \sum_{c \in \mathcal{C}} n_1(c) \log(n_1(c)) + \sum_{c \in \mathcal{C}} n_2(c) \log(n_2(c)) - \sum_{c, c' \in \mathcal{C}} n(c, c') \log(n(c, c')) \quad (2.16)$$

Das Kriterium ML bzw. ML^* besitzt jedoch einen entscheidenden Nachteil. Das optimale Kategoriensystem im Sinne des Kriteriums ML besteht nämlich darin, jedem Wort genau eine Kategorie zuzuordnen. In diesem Fall ergibt sich:

$$\text{ML}_{c:=w} = 2 \sum_{w \in \mathcal{W}} n(w) \log(n(w)) - \sum_{w, w' \in \mathcal{W}} n(w, w') \log(n(w, w')) \quad (2.17)$$

In einem anderen Kategoriensystem müssen wenigstens zwei Wörter in einer Kategorie sein. Seien w_1 und w_2 zwei verschiedene Wörter, die in eine Kategorie zusammengefaßt werden. Dann müssen in Gleichung (2.17) die folgenden Ersetzungen vorgenommen werden:

$$\begin{aligned} n(w_1) \log(n(w_1)) + n(w_2) \log(n(w_2)) &\Rightarrow (n(w_1) + n(w_2)) \log(n(w_1) + n(w_2)) \\ n(\cdot, w_1) \log(n(\cdot, w_1)) + n(\cdot, w_2) \log(n(\cdot, w_2)) &\Rightarrow (n(\cdot, w_1) + n(\cdot, w_2)) \log(n(\cdot, w_1) + n(\cdot, w_2)) \\ n(w_1, \cdot) \log(n(w_1, \cdot)) + n(w_2, \cdot) \log(n(w_2, \cdot)) &\Rightarrow (n(w_1, \cdot) + n(w_2, \cdot)) \log(n(w_1, \cdot) + n(w_2, \cdot)) \end{aligned}$$

Es gilt jedoch allgemein $\forall m, n \geq 0$:

$$\begin{aligned} n \log(n) + m \log(m) &\leq n \log(\max(n, m)) + m \log(\max(n, m)) \\ &\leq (n + m) \log(\max(n, m)) \\ &\leq (n + m) \log(n + m) \end{aligned} \tag{2.18}$$

Wegen diesen allgemeingültigen Ungleichungen können die obigen Ersetzungen den Wert des Optimierungskriteriums nicht verkleinern.

Wenn jedoch jedes Wort genau einer Kategorie entspricht, ist die Bildung von Kategorien überflüssig. Im Falle eines umfassenden Korpus, bei dem jedes Bigramm der Sprache entsprechend seiner Wahrscheinlichkeit auftritt, ist dies in der Tat der Fall, da alle benötigten Wahrscheinlichkeiten aus dem Korpus geschätzt werden können.

Um also das Optimierungskriterium ML anwenden zu können, muß zu Beginn der Optimierung die Zahl der Kategorien festgelegt werden. Die ‘richtige’ Kategorienzahl kann dann mittels eines vom Trainingskorpus unabhängigen Testkorpus ermittelt werden. Hierzu muß man für verschiedene Kategorienzahlen im Sinne des Optimierungskriteriums gute Kategoriensysteme ermitteln. Diese werden dann anhand des Testkorpus verglichen und so die optimale Kategorienzahl bestimmt (siehe Kapitel 6).

2.2.2 Maximum-Likelihood-Kriterium mit Kreuzvalidierung

Das Problem des vorhergehenden Ansatzes liegt darin, daß das Trainingskorpus $\mathbf{w} = w_1 \dots w_m$ im allgemeinen nicht umfassend ist. Die überwiegende Zahl der Wort-Bigramme tritt nicht auf. Man muß also darauf achten, daß die gebildeten Kategorien allgemein sind, obwohl sie nur auf einem eingeschränkten Korpus gebildet wurden. Im folgenden soll ein Optimierungskriterium entwickelt werden, in dem dies berücksichtigt wird.

Im vorhergehenden Ansatz wurden die Wahrscheinlichkeiten für Kategorien-Unigramme und Kategorien-Bigramme anhand der relativen Häufigkeiten im Trainingskorpus geschätzt. Anhand dieser Wahrscheinlichkeiten wurde dann das Kategoriensystem am selben Trainingskorpus bewertet, wodurch insbesondere keine unbeobachteten Ereignisse auftraten. Anders ist dies jedoch bei einem unabhängigen Testkorpus. Hier treten unbeobachtete Ereignisse auf, denen eine von 0 verschiedene Wahrscheinlichkeit zugeordnet werden muß.

Mit dem Prinzip der rotierenden Kreuzvalidierung (leave-one-out) soll im weiteren ein alternatives Optimierungskriterium vorgestellt werden, welches obige Problematik berücksichtigt ([Kne93]). Hierbei besteht die Grundidee darin, das Korpus in N Teile aufzuteilen,

$N - 1$ Anteile zum Schätzen der Wahrscheinlichkeiten, und den verbleibenden Anteil zum Test zu verwenden. Der Testanteil wird immer variiert, so daß jeder Anteil einmal zum Test und $N - 1$ Male zum Schätzen verwendet wird.

Ein Testanteil bestehe jeweils aus einem einzelnen Bigramm (w_{i-1}, w_i) . Der Trainingsanteil T_i besteht also aus allen anderen Bigrammen. Man beachte, daß durch das Entfernen eines Bigramms aus dem Trainingskorpus der Fall simuliert wird, daß ein Bigramm im Testkorpus auftritt, aber nicht im Trainingskorpus. Da ein Korpus eine Folge von Wörtern und ein Bigramm ein statistisches Ereignis darstellt, muß man definieren, was unter dem Entfernen eines Bigrammes verstanden werden soll. Falsch wäre es, einem Bigramm die Folge der Wörter $w_{i-1}w_i$ zuzuordnen. Ein Entfernen dieser Wörter würde die umliegenden Bigramme (w_{i-2}, w_{i-1}) und (w_i, w_{i+1}) zerstören. Ebenso falsch wäre es, das Entfernen eines Bigrammes als Auftrennen des Trainingskorpus anzusehen. Dann würde das Korpus in $w_1 \cdots w_{i-1}$ und $w_i \cdots w_m$ zerschnitten und als Satzbegrenzer \$ eingefügt. Man kann jedoch das Korpus als Multimenge von statistischen Ereignissen — in unserem Falle also als Multimenge von Bigrammen — sehen:

$$w_1 \dots w_m \Leftrightarrow (w_1, w_2), \dots, (w_{m-1}, w_m) \Leftrightarrow \{(w_i, w_j) : n(w_i, w_j) | w_i, w_j \in \mathcal{W}\} \quad (2.19)$$

Dabei gibt $n(w_i, w_j)$ die Häufigkeit des Wort-Bigramms (w_i, w_j) an. Ein Entfernen eines einzelnen Bigramms (w_i, w_j) aus dem Trainingskorpus bewirkt also ein Dekrementieren von $n(w_i, w_j)$ um 1.

Die Wahrscheinlichkeit für das Test-Bigramm soll $P_{T_i}(w_i | w_{i-1})$ genannt werden. Durch die Schreibweise (aus [Kne93]) wird ausgedrückt, daß sich die Wahrscheinlichkeiten aus dem Trainingsanteil T_i ergeben. Da jedes Bigramm genau einmal im Testkorpus auftritt, ergibt sich die in Gleichung (2.4) angegebene Wahrscheinlichkeit für das Korpus zu:

$$P_{LO}(\mathbf{w}) = P_{T_1}(w_1 | \$) \cdot P_{T_{m+1}}(\$ | w_m) \cdot \prod_{i=2}^m P_{T_i}(w_i | w_{i-1}) \quad (2.20)$$

$$= \prod_{i=1}^{m+1} P_{T_i}(w_i | w_{i-1}) \quad (2.21)$$

Man beachte, daß der Unterschied zu Gleichung (2.4) darin besteht, daß die Wahrscheinlichkeit für ein Wort-Bigramm von einem Trainingskorpus T_i abhängt. Die Wahrscheinlichkeiten für Wort-Bigramme werden nun erneut durch die Wahrscheinlichkeiten für Kategorien-Bigramme angenähert:

$$P_{T_i}(w_i | w_{i-1}) \approx P_{T_i}(C(w_i) | C(w_{i-1})) \cdot P_{T_i}(w_i | C(w_i)) \quad (2.22)$$

$$= \frac{P_{T_i}(C(w_{i-1}), C(w_i))}{P_{T_i}(C(w_{i-1}))} \cdot \frac{P_{T_i}(w_i)}{P_{T_i}(C(w_i))} \quad (2.23)$$

Es müssen nun $P_{T_i}(c, c')$ und $P_{T_i}(c)$ geschätzt werden. Da man davon ausgehen kann, daß Kategorien-Unigramme im Trainingskorpus oft auftreten, kann man $P_{T_i}(c)$ durch relative Häufigkeiten $n_i(c, c')$, $n_{i,1}(c)$ und $n_{i,2}(c)$ schätzen. Der Index i bringt wieder zum Ausdruck, daß sich diese relativen Häufigkeiten ohne das Bigramm (w_{i-1}, w_i) ergeben. Im Falle der

Kategorien-Bigramme muß man nicht-beobachtete Bigramme schätzen. Dazu verwenden wir die *absolute discounting*-Methode aus [Ney93], bei der die Häufigkeiten mit einer Konstante $\rho < 1$ reduziert werden und das damit erhaltene Wahrscheinlichkeitspotential über alle unbeobachteten Bigramme aufgeteilt wird. In [Ney93] wurde vorgeschlagen $\rho = 0.75$ zu setzen. Man erhält:

$$P_{T_i}(c_{i-1}, c_i) = \begin{cases} \frac{n_i(c_{i-1}, c_i) - \rho}{n_i} & \text{falls } n_i(c_{i-1}, c_i) > 0 \\ \frac{\eta_{+,i}\rho}{\eta_{0,i}n_i} & \text{falls } n_i(c_{i-1}, c_i) = 0 \end{cases} \quad (2.24)$$

Es wurde $C(w_i)$ durch c_i abgekürzt, $\eta_{0,i}$ ist die Anzahl der unbeobachteten Bigramme, und $\eta_{+,i}$ die Anzahl der beobachteten Bigramme. Die Wahrscheinlichkeit $\frac{\eta_{+,i}\rho}{\eta_{0,i}n_i}$ für unbeobachtete Bigramme ergibt sich aus der Normierungsbedingung $\sum_i P_{T_i}(c_{i-1}, c_i) = 1$. Es ergibt sich also eine Gleichung (2.11) entsprechende Wahrscheinlichkeit für das Trainingskorpus:

$$\begin{aligned} \hat{P}_{LO}(\mathbf{w}|\mathcal{C}) &= \prod_{i=1}^{m+1} \frac{n_{i,2}(w_i) \cdot n_i}{n_{i,1}(c_{i-1})n_{i,2}(c_i)} \\ &\cdot \left(\frac{n_i(c_{i-1}, c_i) - \rho}{n_i} [n_i(c_{i-1}, c_i) > 0] + \frac{\eta_{+,i}\rho}{\eta_{0,i}n_i} [n_i(c_{i-1}, c_i) = 0] \right) \end{aligned} \quad (2.25)$$

Dabei wurde die Schreibweise von Iverson (aus [Gra94]) zum Einbringen von Bedingungen in Gleichungen verwendet. Der Term $[P]$ soll genau dann 1 sein, wenn die Bedingung P erfüllt ist und genau dann 0 sein, wenn die Bedingung P nicht erfüllt ist.

Wie beim einfachen Maximum-Likelihood-Ansatz erhält man durch Bildung des negativen Logarithmus, Umsortieren und Vernachlässigen konstanter Terme das Optimierungskriterium LO:

$$\begin{aligned} \text{LO}(\mathcal{C}) &= \sum_{i=1}^{m+1} \log(n_{i,1}(c_{i-1})) + \sum_{i=1}^{m+1} \log(n_{i,2}(c_i)) \\ &- \log \left((n_i(c_{i-1}, c_i) - \rho) [n_i(c_{i-1}, c_i) > 0] + \frac{\eta_{+,i}\rho}{\eta_{0,i}} [n_i(c_{i-1}, c_i) = 0] \right) \end{aligned} \quad (2.26)$$

Durch folgende Überlegungen läßt sich die Abhängigkeit von T_i eliminieren.

- Es gilt $n_i(c_{i-1}, c_i) = n(c_{i-1}, c_i) - 1$, da das Bigramm (w_{i-1}, w_i) genau einmal herausgenommen wird.
- Für Kategorien-Unigramme gilt entsprechend $n_{i,1}(c_{i-1}) = n_1(c_{i-1}) - 1$ und $n_{i,2}(c_i) = n_2(c_i) - 1$. Da jedoch wieder $n_1(c) = n_2(c)$ gilt, kann eine Zählfunktion $n(c) := n_1(c)$ eingeführt werden und es gilt: $n_{i,1}(c) = n_{i,2}(c) = n(c) - 1$.
- Im Falle $n_i(c_{i-1}, c_i) = 0$ gilt $\eta_{+,i} = \eta_+ - 1$ und $\eta_{0,i} = \eta_0 + 1$, da ein einmal vorkommendes Bigramm herausgenommen wird.

Man erhält:

$$\begin{aligned} \text{LO}(\mathcal{C}) = & 2 \sum_{i=1}^{m+1} \log(n(c_i) - 1) - \log\left(\frac{(\eta_+ - 1)\rho}{(\eta_0 + 1)}\right) \sum_{i=1}^{m+1} [n(c_{i-1}, c_i) = 1] \\ & - \sum_{i=1}^{m+1} \log(n(c_{i-1}, c_i) - 1 - \rho) [n(c_{i-1}, c_i) > 1] \end{aligned} \quad (2.27)$$

Zum Umsortieren der Summe wie beim einfachen Maximum-Likelihood-Ansatz benötigt man folgende Überlegungen:

- Der Term $\log(n(c_i) - 1)$ tritt für eine feste Kategorie c genau $n(c)$ mal auf.
- Die Bedingung $n(c_{i-1}, c_i) > 1$ ist für feste Kategorien c und c' genau $n(c, c')$ mal erfüllt, wenn $n(c, c') > 1$ ist.
- Die Bedingung $n(c_{i-1}, c_i) = 1$ ist η_1 -mal erfüllt, wobei η_1 die Zahl der einmal auftretenden Kategorien-Bigramme bezeichnet.

Es ergibt sich die endgültige Form des Optimierungskriteriums LO zu ([Kne93]):

$$\begin{aligned} \text{LO}(\mathcal{C}) = & 2 \sum_{c \in \mathcal{C}} n(c) \log(n(c) - 1) - \eta_1 \log\left(\frac{\eta_+ - 1}{\eta_0 + 1} \rho\right) \\ & - \sum_{c, c' \in \mathcal{C}} [n(c, c') > 1] n(c, c') \log(n(c, c') - 1 - \rho) \end{aligned} \quad (2.28)$$

Zusammen mit dem Optimierungskriterium ML (Gleichung (2.15)) besitzt man nun zwei Optimierungskriterien für disjunkte Kategoriensysteme. Das Kriterium ML verwendet einen einfachen Maximum-Likelihood-Ansatz, während das Kriterium LO zusätzlich eine Kreuzvalidierungskomponente besitzt. Es stellt sich nun die Frage, welches der beiden Kriterien bessere Ergebnisse liefert und ob es (wie in [Kne93]) möglich ist, mit LO die ‘optimale’ Kategorienzahl zu bestimmen. Dies wird in Kapitel 6 behandelt.

Im folgenden soll das Optimierungsproblem in einen allgemeinen Rahmen gestellt werden.

2.3 Formulierung des Optimierungsproblems

Ein *kombinatorisches Optimierungsproblem* ist ein Minimierungs-(Maximierungs-)problem einer Kosten-(Bewertungs-)funktion auf einer diskreten Menge. Im folgenden soll ohne Beschränkung der Allgemeinheit von Minimierung gesprochen werden.

Formal läßt sich ein kombinatorisches Optimierungsproblem COP als Tupel darstellen ([Fis94]):

$$COP = (Z, \phi, Z_{min}) \quad (2.29)$$

Die Variablen besitzen die folgende Bedeutung:

- Z ist die diskrete Zustandsmenge.
- ϕ ist die Kostenfunktion: $\phi : Z \rightarrow \mathbb{R}$. Sie ordnet jedem Element der Zustandsmenge seine Kosten zu.
- Z_{min} ist die Lösungsmenge des Problems, also die Menge der Zustände mit minimalen Kosten

$$Z_{min} = \{z \in Z \mid \forall z' \in Z : \phi(z) \leq \phi(z')\} \quad (2.30)$$

Man muß jedoch aus Aufwandsgründen oft darauf verzichten Z_{min} (oder auch nur ein Element daraus) zu bestimmen ([Fis94]). Dies ist insbesondere bei NP-harten Problemen der Fall, wie beispielsweise dem Problem des Handlungsreisenden oder dem Stundenplanproblem, für die keine effizienten Lösungsalgorithmen bekannt sind. Man begnügt sich dann mit Lösungen, deren Kosten ‘nahe’ den optimalen Lösungen liegen.

In dem hier behandelten Problem ist die diskrete Zustandsmenge Z gegeben durch die Menge der Partitionen des Vokabulars \mathcal{W} des Trainingskorpus:

$$Z = \{\mathcal{C} \mid \mathcal{C} = \{c_1, c_2, \dots, c_{|\mathcal{C}|}\} \wedge \bigcup_{i=1}^{|\mathcal{C}|} c_i = \mathcal{W} \wedge \forall (1 \leq i < j \leq |\mathcal{C}|) : c_i \cap c_j = \emptyset\} \quad (2.31)$$

Die Kardinalität der Menge Z wächst schnell mit der Größe des Vokabulars. Die Zahl der Partitionen einer n -elementigen Menge in k nichtleere Mengen ist gegeben durch $S(n, k)$, den Stirling-Zahlen der zweiten Art ([Gra94]). Schon für $k = 2$, also die Aufteilung in zwei Kategorien, wächst die mögliche Zahl der Kategoriensysteme exponentiell mit der Größe des Vokabulars: $S(n, 2) = 2^{n-1} - 1$.

Die Kostenfunktion ϕ ist bestimmt durch die Optimierungskriterien ML und LO. Durch Z und ϕ ist auch Z_{min} bestimmt. Da Z endlich ist, ist es sehr einfach möglich, einen Algorithmus zur Bestimmung von Z_{min} anzugeben. Es müssen einfach alle Partitionen aufgezählt werden, ihre Kosten ermittelt und die Partitionen mit kleinsten Kosten ausgewählt werden. Aufgrund der Größe des Zustandsraums, die im allgemeinen exponentiell mit der Vokabulargröße wächst, ist ein vollständiges Durchsuchen der Zustandsmenge bei großem Vokabular nicht mehr möglich.

Die Elemente der Lösungsmenge Z_{min} werden auch als globale Minima bezeichnet. Um den Begriff des *lokalen Minimums* zu definieren muß zuerst der Begriff der *Nachbarschaft* definiert werden. Die Nachbarschaft \sim_N ist eine Relation auf der Menge der Zustände. Durch die Definition einer Nachbarschaftsrelation wird der Zustandsraum strukturiert. Jeder Zustand besitzt eine Menge von Nachbarzuständen $N(z) \neq \emptyset$:

$$N(z) = \{z' \mid z \sim_N z'\} \quad (2.32)$$

Im hier behandelten Optimierungsproblem sollen zwei Zustände als benachbart gelten, wenn sie sich in der Kategorienzugehörigkeit genau eines Wortes unterscheiden. Dies ist

wohl die natürlichste Form der Nachbarschaft. Ein lokales Minimum z_{lokmin} ist nun ein Zustand, dessen direkte Nachbarzustände keine kleineren Kosten besitzen. Die Menge der lokalen Minima Z_{lokmin} wird formal definiert durch:

$$Z_{lokmin} := \{z \in Z \mid \forall z' \in N(z) : \phi(z) \leq \phi(z')\} \quad (2.33)$$

Ein lokales Minimum ist damit ein Kategoriensystem, dessen Kosten durch Transport eines einzelnen Wortes nicht verringert werden können.

2.4 Zusammenfassung

Es wurde ausgehend vom Bayes-Klassifikator mit dem Maximum-Likelihood-Ansatz das formale Kriterium ML für die Güte disjunkter Kategoriensysteme entwickelt. Die bezüglich dieses Kriteriums optimalen Kategoriensysteme maximieren die Ableitungswahrscheinlichkeit für das Trainingskorpus. Dies führt insbesondere dazu, daß die Kategorienzahl vorgegeben werden muß. Es wurde ein alternatives Kriterium LO entwickelt, welches durch Kreuzvalidierung versucht zu ‘verallgemeinern’.

Beide Optimierungskriterien benötigen in ihrer schließlichen Form (Gleichungen (2.15) und (2.28)) nur Bigramm- und Unigramm-Statistiken. Alle weiteren Informationen aus dem Korpus haben keinen Einfluß auf die Auswertung der Optimierungskriterien. Hieraus ist ersichtlich, daß die Komplexität der Auswertung nicht von der Größe des Korpus abhängt, sondern höchstens von der Größe der Bigramm-Statistik, welche wiederum abhängig ist von der Größe des Vokabulars. In Abschnitt 4.1 wird dargestellt, wie eine möglichst effiziente Auswertung der Optimierungskriterien in der Nachbarschaft eines Zustandes durchgeführt werden kann.

Das konkrete Optimierungsproblem wurde schließlich noch formal dargestellt durch Definition von Zustandsraum, Kostenfunktion und Nachbarschaft. Der sehr große Zustandsraum macht ein vollständiges Durchsuchen unmöglich. In Kapitel 3 werden allgemeine Verfahren zur Lösung kombinatorischer Optimierungsprobleme vorgestellt.

Kapitel 3

Die Optimierungsverfahren

In diesem Abschnitt sollen die Optimierungsverfahren beschrieben werden, die zur Lösung des im vorhergehenden Kapitel formulierten Optimierungsproblems verwendet werden. Die Darstellung erfolgt dabei soweit als möglich allgemein und unabhängig vom Optimierungsproblem. Es wird die Klasse der *iterativ-optimierenden Verfahren* vorgestellt. Auf diesen Verfahren baut die *multidirektionale Suche mit Beschneidung* auf, welche durch parallele Verarbeitung mehrerer iterativ-optimierender Verfahren effizienter gute Lösungen berechnet. Die *iterierte Zustandsraum-Reduktion* stellt ein problemspezifisches Verfahren dar, welches die gleichen Anteile einer Menge von ‘guten Kategoriensystemen’ dazu verwendet, eine Zustandsraum-Reduktion vorzunehmen.

Es wird schließlich eine objektorientierte Implementierung motiviert, welche insbesondere eine logische Trennung der Optimierungsverfahren vom Optimierungsproblem vorsieht.

3.1 Iterativ-optimierende Verfahren

Mit dem Stichwort *iterativ-optimierende Verfahren* sollen die Algorithmen beschrieben werden, die einen Zustand wiederholt durch kleine Modifikationen verändern. Darunter fallen die im weiteren erläuterten Algorithmen stochastische Relaxation (Hill Climbing), simuliertes Ausfrieren (Simulated Annealing), Sinflutalgorithmus (Great Deluge Algorithm), Schwellwertakzeptanz (Threshold Accepting) und Record-To-Record Travel.

All diese Algorithmen lassen sich durch Verfeinerungen des Algorithmenschemas für den iterativ-optimierenden Algorithmus mit globaler Auswertung (Bild 3.1) beschreiben.

Dabei wird ein initialer Zustand erzeugt (**initialize**) und bewertet (**value**). Der anfängliche Zustand der Optimierungsverfahren T_0 — bei den hier behandelten Verfahren auch Temperatur genannt — wird bestimmt (**INIT**). In einer Schleife werden laufend neue Zustände in einer Nachbarschaft erzeugt (**generate**), bewertet (**value**) und wenn ein algorithmenspezifisches Annahmekriterium (**ACCEPT**) erfüllt ist, wird der erzeugte Zustand akzeptiert. Außerdem wird stetig eine Veränderung am Zustand T_{t-1} des Optimierungsverfahrens vorgenommen (**COOL**), die normalerweise als Abkühlung bezeichnet wird. Dies

$t := 0; T_0 := \mathbf{INIT}()$
$z_c := \mathbf{initialize}(); \phi_c := \mathbf{value}(z_c)$
WHILE $\mathbf{END}() = \mathbf{TRUE}$
$t := t + 1$
$z_n := \mathbf{generate}(z_c); \phi_n := \mathbf{value}(z_n); \Delta\phi = \phi_n - \phi_c$
$T_t := \mathbf{COOL}(T_{t-1}, \phi_c)$
IF $\mathbf{ACCEPT}(\phi_c, \Delta\phi) = \mathbf{TRUE}$
THEN $z_c := z_n; \phi_c := \phi_n$

Bild 3.1: Iterativ-optimierender Algorithmus — Globale Auswertung.

wird bei der Erläuterung von simuliertem Ausfrieren (Abschnitt 3.1.2) noch genauer dargestellt. Die Schleife wird beendet, wenn ein bestimmtes Endkriterium (**END**) erfüllt ist. Ähnliche Darstellungen dieses Verfahrens findet man in ([Fis94]).

Im obigen Algorithmus werden mittels **value** die Kosten eines Zustandes berechnet. Bei komplexen Optimierungsproblemen kann dies eine sehr aufwendige Operation sein. Oft können die Kosten jedoch *inkrementell* berechnet werden, das bedeutet, die Änderung der Kosten eines Zustandes kann anhand der Zustandsänderung berechnet werden. Wenn dies effizienter als eine vollständige Neuauswertung geschehen kann, so ist das folgende Algorithmenschema vorzuziehen.

$t := 0; T_0 := \mathbf{INIT}()$
$z_c := \mathbf{initialize}(); \phi_c := \mathbf{value}(z_c)$
WHILE $\mathbf{END}() = \mathbf{TRUE}$
$t := t + 1$
$\Delta z := \mathbf{change}(z_c); \Delta\phi := \mathbf{valueChange}(z_c, \Delta z)$
$T_t := \mathbf{COOL}(T_{t-1}, \phi_c)$
IF $\mathbf{ACCEPT}(\phi_c, \Delta\phi) = \mathbf{TRUE}$
THEN $z_c := \mathbf{doChange}(z_c, \Delta z); \phi_c := \phi_c + \Delta\phi$

Bild 3.2: Iterativ-optimierender Algorithmus — Lokale Auswertung.

Hierbei wird wie vorher ein initialer Zustand erzeugt und in einer Schleife iterativ verändert. Es wird jedoch nicht mehr ein neuer Zustand mittels **generate** erzeugt und bewertet, sondern es wird eine Zustandsänderung mittels **change** erzeugt und die Kostenänderung ermittelt (**valueChange**). Erst wenn die Zustandsänderung wirklich akzeptiert wird, erfolgt eine Änderung des aktuellen Zustandes.

Es sind nun folgende wichtige Eigenschaften des Algorithmus in Bild 3.2 zu bemerken:

- Die globale Kostenberechnung **value** wird nur ein einziges Mal aufgerufen.
- Es bietet sich eine objektorientierte Implementierung an. Die vom konkreten Optimierungsverfahren abhängigen Funktionen **INIT**, **ACCEPT**, **COOL** und **END** können

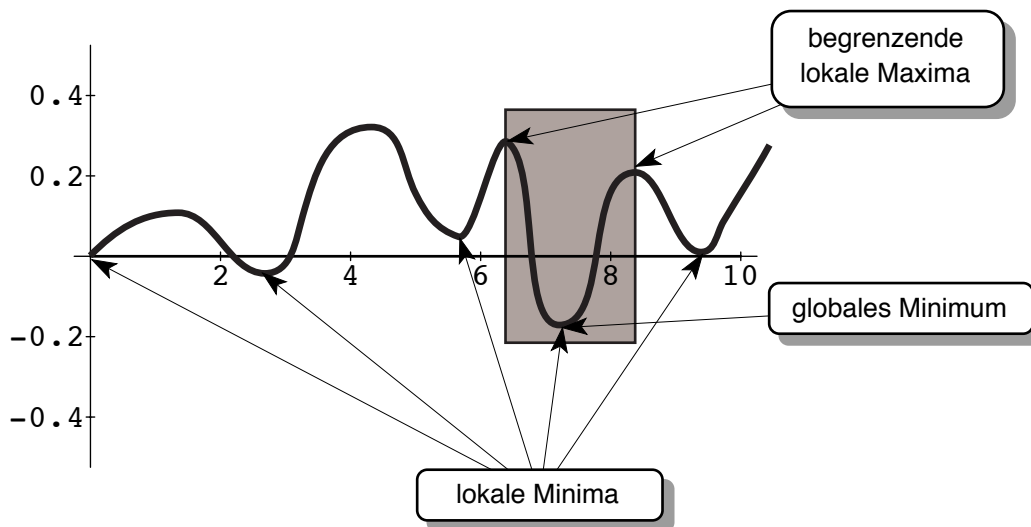


Bild 3.3: Erreichbarkeit globaler Minima bei stochastischer Relaxation (HC).

getrennt werden von den problemabhängigen Anteilen **initialize**, **value**, **change**, **valueChange** und **doChange**. Dies wird näher in Abschnitt 3.4 betrachtet.

Im folgenden werden fünf iterativ-optimierende Verfahren vorgestellt. Durch die konzeptionelle Trennung von Optimierungsverfahren und Optimierungsproblem kann dieses Kapitel ganz allgemein gehalten werden, ohne Bezug auf das in dieser Arbeit interessierende Problem.

3.1.1 Stochastische Relaxation (Hill Climbing, HC)

Bei stochastischer Relaxation (Hill Climbing, HC) wird eine Änderung nur dann akzeptiert, wenn sie verbessernd wirkt. Dies führt dazu, daß dieser Algorithmus sehr schnell (im Vergleich zu den später vorgestellten Verfahren) in einem lokalen Minimum konvergiert und dieses nicht mehr verlassen kann. Da bei komplexen Optimierungsproblemen normalerweise viele lokale Minima existieren, ist es vom Startzustand abhängig, ob ein bestimmtes Minimum erreicht werden kann. Beispielsweise wird das globale Minimum der Funktion in Bild 3.3 nur dann erreicht, wenn der Startzustand innerhalb des gekennzeichneten Bereichs liegt.

Das Akzeptanzkriterium lautet:

$$\text{ACCEPT}_{HC} = \begin{cases} \text{TRUE} & \text{für } \Delta\phi < 0 \\ \text{FALSE} & \text{sonst} \end{cases} \quad (3.1)$$

Die stochastische Relaxation besitzt keine Parameter und ist damit ein sehr einfaches Verfahren. Alle weiteren Optimierungsverfahren benötigen Parameter, deren Bestimmung meist aufwendig ist und Informationen über das Optimierungsproblem erfordert.

3.1.2 Simuliertes Ausfrieren (Simulated Annealing, SA)

Der Nachteil der stochastischen Relaxation liegt darin, daß ein lokales Minimum nicht mehr verlassen wird, da die Kosten nur verkleinert werden dürfen. Bei simuliertem Ausfrieren und seinen Abwandlungen Schwellwertakzeptanz, Record-To-Record Travel und Simulated Annealing ist es möglich, daß eine Verschlechterung akzeptiert wird. Damit besitzen diese Verfahren im Prinzip die Möglichkeit, ein einmal erreichtes lokales Minimum wieder zu verlassen.

Simuliertes Ausfrieren ([Kir83, Laa87, Nur93, Fis94]) geht zurück auf Untersuchungen von Metropolis über die Simulation des Übergangs eines Festkörpers ins thermische Gleichgewicht ([Met53]). In [Kir83] wurde dann ein darauf basierendes Verfahren zur Lösung kombinatorischer Optimierungsprobleme entwickelt. Beim Ausfrieren wird ein Material stark erhitzt und dann so langsam abgekühlt, daß die Atome sich in einem kristallinen Gitter strukturieren können. Bei zu schneller Abkühlung würden Unregelmäßigkeiten entstehen. Es ist nun das Ziel, einen energiereichen Anfangszustand (hohe Temperatur) in einen möglichst energiearmen Grundzustand (niedrige Temperatur) zu überführen. Dabei erlauben die physikalischen Gesetze, daß einzelne Teilchen im Laufe der Abkühlung kurzfristig eine höhere Energie erlangen. Dadurch können lokale Minima wieder verlassen werden.

Eine Übertragung der physikalischen Ergebnisse auf Optimierungsprobleme führt zu einem Akzeptanzkriterium, bei dem eine Zunahme der Kosten mit exponentiell abnehmender Wahrscheinlichkeit angenommen werden. Die Annahmewahrscheinlichkeit einer Kostenänderung $\Delta\phi$ ergibt sich zu:

$$P(\Delta\phi, T) = \begin{cases} 1 & \Delta\phi \leq 0 \\ \exp(-\frac{\Delta\phi}{T}) & \text{sonst} \end{cases} \quad (3.2)$$

Dies wird auch als *Metropolis-Kriterium* bezeichnet. Der Parameter T entspricht der Temperatur im physikalischen Prozeß des Ausfrierens und wird im Laufe der Optimierung schrittweise verringert. Die Art und Weise der Abkühlung wird durch den sogenannten *Abkühlplan* festgelegt. Mit logarithmischer Abkühlung ist es möglich, das Erreichen eines globalen Minimums der Kostenfunktion zu garantieren. Die dazu nötigen Rechenzeiten sind jedoch zu groß. Es soll hier eine exponentiell abkühlende Strategie angewendet werden, wie sie häufig vorgeschlagen wird. Dabei wird in jedem Abkühlungsschritt die aktuelle Temperatur T_t mit einem Faktor $\alpha < 1$ multipliziert ([Kir83]). Hier soll ein Abkühlungsschritt nach S Iterationsschritten erfolgen.

$$T_t = T_0 \cdot \alpha^{\lfloor t/S \rfloor} \quad (3.3)$$

Das Verfahren besitzt demnach drei Parameter:

- T_0 : Startwert der Temperatur
- S : Schrittzahl bis zur Reduktion der Temperatur
- α : Abkühlungsfaktor für Temperatur

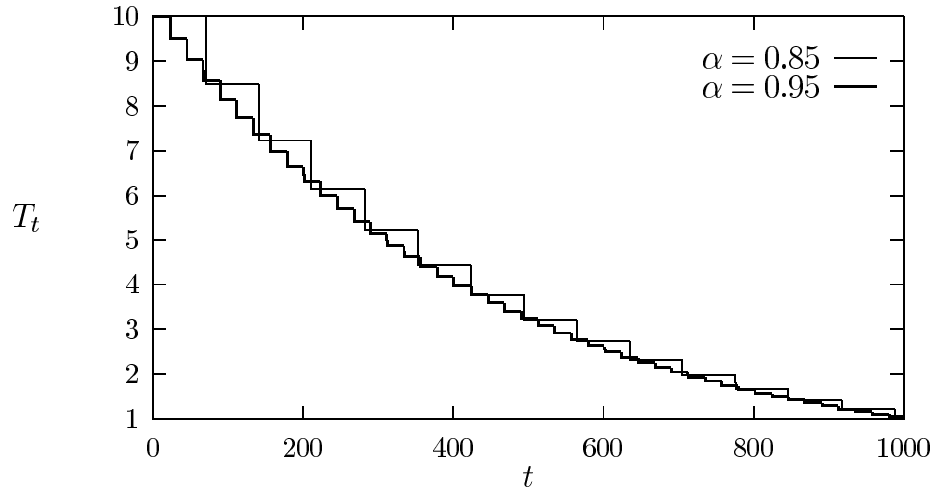


Bild 3.4: Abkühlungskurven für simuliertes Ausfrieren mit gleicher Anfangs- und Endtemperatur, jedoch mit $\alpha = 0.95$ und $\alpha = 0.85$.

Das Akzeptanzkriterium lautet:

$$\mathbf{ACCEPT}_{SA} = \begin{cases} TRUE & \text{für } \Delta\phi \leq 0 \\ TRUE & \text{für } \Delta\phi > 0 \wedge q \leq P(\Delta\phi, T) \\ FALSE & \text{sonst} \end{cases} \quad (3.4)$$

Dabei ist q eine gleichverteilte Zufallszahl aus dem Intervall $[0, 1]$.

Der Bestimmung des Startwertes der Temperatur T_0 kommt besondere Bedeutung zu. Ein erheblich zu hoher Startwert führt zu einer nutzlosen anfänglichen Abkühlphase. Eine zu kleine Starttemperatur führt zu einer zu geringen Annahmewahrscheinlichkeit schlechterer Zustände und damit einem ähnlichen Verhalten wie stochastische Relaxation. Es wurde vorgeschlagen den Startwert so zu wählen, daß zu Beginn fast alle Zustandsübergänge erlaubt sind [Laa87]. Damit ergibt sich das folgende einfache Verfahren (von [Joh86], zitiert aus [Fis94]) zur Bestimmung des Startwertes. Es werden n zufällige Zustandsübergänge mit Kostenänderung $\Delta\phi_i$ beobachtet. Darunter seien m_0 Verschlechterungen. Von diesen bildet man den mittleren Kostenzuwachs:

$$\overline{\Delta\phi}_0^{(+)} = \frac{1}{m_0} \sum_{i=1}^m [\Delta\phi_i > 0] \Delta\phi_i \quad (3.5)$$

Zur Bestimmung von T_0 wird eine *anfängliche Annahmerate* $\gamma_0 \approx 1$ vorgegeben und durch Umformung von $\gamma_0 = \exp(-\overline{\Delta\phi}_0^{(+)} / T_0)$ erhält man:

$$T_0 = \frac{\overline{\Delta\phi}_0^{(+)}}{\ln(\gamma_0^{-1})} \quad (3.6)$$

Es müssen nun noch die Schrittzahl S und der Abkühlungsfaktor α bestimmt werden. Die Parameter S und α sind im Rahmen der notwendigen Genauigkeit voneinander abhängig. Dies bedeutet, daß bei einer Vergrößerung der Schrittzahl S der Abkühlungsfaktor α reduziert werden muß, um einen im Prinzip gleichen Abkühlungsplan zu erhalten, der nur eine höhere Abstufung der Temperatur aufweist (siehe Bild 3.4). Solange die Abstufung nicht zu groß ist, hat dies jedoch keinen Einfluß auf die Qualität des Ergebnisses, da dies robust gegen kleinere Veränderungen von α reagiert ([Fis94]). Der Grund für die Einführung einer Schrittzahl ungleich 1 liegt darin, daß bei einer großen Zahl von Iterationsschritten der Abkühlungsfaktor α im Rahmen der Rechengenauigkeit nicht mehr von 1 unterschieden werden kann. Es kann einer der beiden Parameter, willkürlich festgelegt werden, z. B. $\alpha = 0.95$.

Die nun noch zu bestimmende Schrittzahl S hängt natürlich von der notwendigen Zahl der Iterationen bis zum Erreichen eines lokalen Minimums und von der Geschwindigkeit der Abkühlung ab. Große Probleme werden eine erheblich größere Zahl von Iterationsschritten benötigen als kleine Probleme. Es soll hierzu der Parameter ν_{SA} eingeführt werden, der die Geschwindigkeit der Abkühlung relativ zur Zahl der Iterationsschritte von stochastischer Relaxation festlegt. Damit bestimmt sich die Zahl der Iterationsschritte I_{SA} bis zum Ende der Abkühlung zu:

$$I_{SA} = \nu_{SA} \cdot I_{HC} \quad (3.7)$$

Ein hohes ν_{SA} führt also zu einer langen Abkühlphase I_{SA} . Wenn man nun eine *schließliche Annahmerate* γ_{end} (entsprechend γ_0) fordert, die nach I_{SA} Iterationsschritten erreicht wird, dann ergibt sich eine Endtemperatur T_{end} :¹

$$T_{end} = \frac{\overline{\Delta\phi_0}^{(+)}}{\ln(\gamma_{end}^{-1})} \quad (3.8)$$

Die Schrittzahl S ergibt sich durch Umformung von $T_{end} = T_0 \cdot \alpha^{I_{SA}/S}$:

$$S = I_{SA} \cdot \frac{\log(\alpha)}{\log(T_{end}/T_0)} \quad (3.9)$$

Damit sind die ursprünglichen drei Parameter T_0 , S und α durch neue Parameter γ_0 , γ_{end} und ν_{SA} bestimmt. Auf den ersten Blick ist das Problem genauso schwierig geblieben, da auch die neuen Parameter festgelegt werden müssen. Diese Parameter liefern jedoch einen der Problemgröße angepaßten Abkühlplan. Ein Parametersatz der ursprünglichen Parameter führt auf einen festen Abkühlplan, der für eine bestimmte Problemgröße gute Ergebnisse liefert, jedoch nicht für erheblich größere oder kleinere Probleme geeignet ist. In Abschnitt 5.1 werden γ_0 , γ_{end} und ν_{SA} anhand vieler Testläufe (für das in Abschnitt

¹Eigentlich dürfte in dieser Gleichung (3.8) nicht der anfängliche (zu Beginn der Optimierung) Kostenzuwachs $\overline{\Delta\phi_0}^{(+)}$ verwendet werden, sondern es müßte der schließliche (am Ende der Optimierung) Kostenzuwachs verwendet werden. Solange jedoch ein grober linearer Zusammenhang zwischen dem anfänglichen und schließlichen Kostenzuwachs besteht, ist dies nicht problematisch, da dies durch den Parameter γ_{end} ausgeglichen werden kann.

2.3 beschriebene Optimierungsproblem) eingestellt. Die Bestimmung der Parameter ist ein zeitaufwendiger Vorgang und kann deshalb nur für kleine Probleme vorgenommen werden. Ob diese Parameter dann für erheblich größere Probleme ebenfalls gute Ergebnisse liefern, kann nur beschränkt nachgeprüft werden. Dieser Punkt wird in Abschnitt 5.1 noch näher betrachtet.

3.1.3 Schwellwertakzeptanz (Threshold Accepting, TA)

Schwellwertakzeptanz ([Due90]; [Nur93], [Fis94]) stellt eine Variante des simulierten Ausfrierens dar, bei dem nicht ein probabilistisches Akzeptanzkriterium verwendet wird, sondern es wird eine Zunahme der Kosten nur bis zu einem Schwellwert zugelassen. Dieses Verfahren wurde in [Due90] vorgestellt und lieferte beim Problem des Handlungsreisenden bessere Ergebnisse als simuliertes Ausfrieren.

Der Algorithmus besitzt einen Parameter T , der angibt, wie sehr die aktuellen Kosten vergrößert werden dürfen. Im Gegensatz zum simulierten Ausfrieren besitzt Schwellwertakzeptanz ein deterministisches Annahmekriterium. T wird im Laufe der Optimierung stetig verkleinert und kann wie bei simuliertem Ausfrieren als Temperatur interpretiert werden. Der Abkühlplan ist wieder von besonderer Bedeutung. Es soll hier T bei jedem Iterationsschritt um ΔT verringert werden. Man beachte hierbei den Unterschied zu der exponentiellen Abkühlung bei simuliertem Ausfrieren (Gleichung (3.3)).

$$T_t = \begin{cases} T_{t-1} - \Delta T & \text{für } T_{t-1} - \Delta T > 0 \\ 0 & \text{sonst} \end{cases} \quad \forall t > 0 \quad (3.10)$$

Der Algorithmus besitzt also zwei Parameter:

- T_0 : Startwert von T
- ΔT : Verringerung von T

Das Akzeptanzkriterium lautet:

$$\mathbf{ACCEPT}_{TA} = \begin{cases} TRUE & \text{für } \Delta\phi < T \\ FALSE & \text{sonst} \end{cases} \quad (3.11)$$

Es sollen nun wie bei simuliertem Ausfrieren die anfängliche Temperatur anhand einer festgelegten anfänglichen Annahmerate bestimmt werden. Dies erreicht man, indem man das γ_{TA} -Quantil der empirisch bestimmten Verteilungsfunktion der anfänglichen Verschlechterungen $F_{(\Delta\phi)(+)}(x)$ ([Fis94]) bestimmt (siehe Bild 3.5). Welche anfängliche Annahmerate γ_{TA} geeignet ist, eine gute Lösung zu gewährleisten, ist a priori nicht offensichtlich und hängt von der Verteilungsfunktion ab. Deshalb sollte die anfängliche Annahmerate problemabhängig bestimmt werden.

Der Parameter ΔT soll wieder aus einer vor Beginn der Optimierung festgelegten Zahl von Iterationen I_{TA} bestimmt werden. Diese soll als Vielfaches der mittleren Zahl der Iterationen von stochastischer Relaxation bestimmt werden:

$$I_{TA} = \nu_{TA} \cdot I_{HC} \quad (3.12)$$

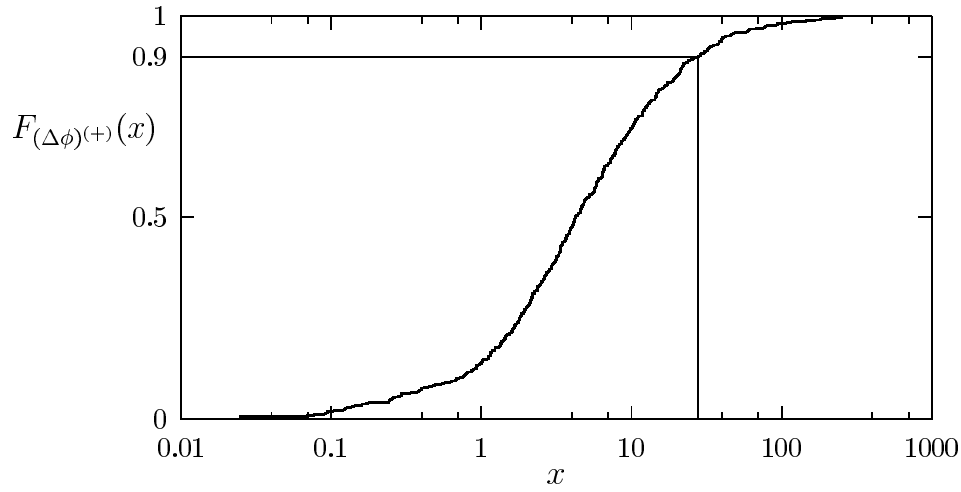


Bild 3.5: Beispiel einer Verteilungsfunktion der anfänglichen Verschlechterungen und Bestimmung der Starttemperatur aus der anfänglichen Annahmerate ($\gamma_{TA} = 0.9$).

Der Parameter ΔT ergibt sich dann zu:

$$\Delta T = \frac{T_0}{\nu_{TA} \cdot I_{HC}} \quad (3.13)$$

Der Algorithmus verhält sich also nach I_{TA} Iterationen exakt wie stochastische Relaxation und konvergiert dann in einem lokalen Minimum.

Die Parameter T_0 und ΔT werden hier anhand von ν_{TA} und γ_{TA} bestimmt. In Abschnitt 5.1 wird die Einstellung dieser Parameter vorgenommen.

3.1.4 Record-To-Record Travel (RRT)

Bei Record-To-Record Travel ([Due93]; [Nur93], [Fis94]) werden im Laufe der Optimierung ständig die geringsten Kosten ϕ_r (Rekord) vermerkt und keine Kosten akzeptiert, die eine Verschlechterung T überschreiten. Es existiert also eine erlaubte Abweichung von den geringsten bisher erreichten Kosten, während bei Schwellwertakzeptanz eine erlaubte Abweichung von den aktuellen Kosten existiert. Prinzipiell können bei Schwellwertakzeptanz beliebig schlechte Kosten über eine Folge von einzelnen Verschlechterungen erreicht werden. Bei Record-To-Record Travel ist dies nicht möglich.

In [Due93] wurde vorgeschlagen, die Abweichung vom Rekord während der Optimierung konstant zu halten (keine Abkühlung). Dies würde dazu führen, daß der Algorithmus normalerweise nicht in einem lokalen Minimum konvergiert, da auch am Ende der Optimierung diese Abweichung erlaubt ist. In dieser Arbeit wurden bessere Ergebnisse erzielt,

wenn T — wie die Temperatur bei Schwellwertakzeptanz — um ein ΔT reduziert wird. Die Temperatur T_t zum Zeitpunkt t ergibt sich durch:

$$T_t = \begin{cases} T_{t-1} - \Delta T & \text{für } T_{t-1} - \Delta T > 0 \\ 0 & \text{sonst} \end{cases} \quad \forall t > 0 \quad (3.14)$$

Der Algorithmus besitzt also zwei Parameter:

- T_0 : anfänglich erlaubte Abweichung vom Rekord
- ΔT : Verringerung von T

Das Akzeptanzkriterium lautet:

$$\mathbf{ACCEPT}_{RRT} = \begin{cases} TRUE & \text{für } \phi_c + \Delta\phi < \phi_r + T \\ FALSE & \text{sonst} \end{cases} \quad (3.15)$$

Die anfängliche erlaubte Abweichung T_0 soll ebenso wie bei Schwellwertakzeptanz anhand der Verteilungsfunktion der anfänglichen Verschlechterungen bestimmt werden. Dabei wird ΔT wieder aus der vorgegebenen Zahl der Iterationen I_{RRT} (wie bei Schwellwertakzeptanz) bestimmt zu:

$$\Delta T = \frac{T_0}{I_{RRT}} \quad (3.16)$$

3.1.5 Sinflutalgorithmus (Great Deluge Algorithm, GDA)

Beim Sinflutalgorithmus ([Due93]; [Nur93], [Fis94]) sind alle Kosten bis zu einem Grenzwert erlaubt. Dieser Wert wird stetig verringert. Man kann sich bei Maximierungsproblemen den Grenzwert als stetig ansteigenden Wasserspiegel im Gebirge der Zustandsbewertungen vorstellen. Ein Zustand wird akzeptiert, wenn seine Kosten über dem aktuellen Wasserspiegel liegen.

Für ein Minimierungsproblem existiert also ein oberer Grenzwert T für die erlaubten Kosten. Dieser wird stetig um einen Wert ΔT verringert, der konstant gewählt werden kann. Ein effizienterer Algorithmus ergibt sich jedoch wenn $T = T - \alpha(T - \phi_c)$, $\alpha \ll 1$ gewählt wird. Dadurch erfolgt eine stärkere Verkleinerung von T , wenn der Abstand zu den Kosten höher ist. Der Wasserspiegel zum Zeitpunkt t ergibt sich zu:

$$T_t = T_{t-1} - \alpha(T_{t-1} - \phi_c) \quad \forall t > 0 \quad (3.17)$$

Der Algorithmus besitzt also nur zwei Parameter:

- T_0 : anfänglicher Wasserspiegel
- α : Parameter für Verringerung des Wasserspiegels

Das Akzeptanzkriterium lautet:

$$\mathbf{ACCEPT}_{GDA} = \begin{cases} TRUE & \text{für } \phi_c + \Delta\phi < T \\ FALSE & \text{sonst} \end{cases} \quad (3.18)$$

Für den Parameter α wird in [Due93] der Wert 0.002 vorgeschlagen. Je größer α gewählt wird, desto schneller nähert sich T den Kosten ϕ_c an und der Algorithmus verhält sich wie stochastische Relaxation. Ein kleines α sorgt für langsame Abkühlung und bessere Ergebnisse.

Für die Wahl des Startwertes T_0 lohnt es sich zu bemerken, daß wenig Gefahr besteht, diesen zu groß zu wählen, da sich T exponentiell schnell den aktuellen Kosten annähert. Wenn also die maximalen Kosten berechenbar oder abschätzbar sind, dann kann man diese wählen.

3.2 Multidirektionale Suche mit Beschneidung

Das Durchführen von n Optimierungsläufen mit Auswahl der besten Lösung stellt eine sehr einfache Möglichkeit der Parallelisierung dar. Es wird jedoch der n -fache Rechenaufwand eines Optimierungslaufes benötigt. Im folgenden soll ein heuristisches Verfahren vorgestellt werden, das dazu dient, den Rechenaufwand zu reduzieren.

Die im letzten Abschnitt vorgestellten iterativ-optimierenden Verfahren sind bis auf simuliertes Ausfrieren deterministisch. Allerdings ist es möglich, durch probabilistische Initialisierung bzw. probabilistische Nachbarschaftsauswahl einen probabilistischen Algorithmus zu erhalten, der für dasselbe Problem immer wieder unterschiedliche Lösungen liefern kann. Es interessiert normalerweise nur die Lösung mit den geringsten Kosten. Es ist also sinnvoll, mehrere probabilistische Algorithmen parallel laufen zu lassen und die beste Lösung auszuwählen. Dies wird auch als *multidirektionale Suche* bezeichnet ([Fis94]). Diese Form der Parallelisierung erweist sich als sehr einfach, da nur am Ende der Optimierung eine Notwendigkeit zur Kommunikation besteht: Es muß die beste Lösung bestimmt werden.

Im folgenden soll der Ablauf eines iterativ-optimierenden Verfahrens im Rahmen einer parallelen Suche *Lauf* genannt werden.

Die Grundidee der *multidirektionalen Suche mit Beschneidung* besteht darin, daß schon nach einem Bruchteil der vollständigen Iterationszeit bestimmte Läufe als nicht mehr erfolgversprechend eingestuft werden können. Ein Lauf gilt als nicht erfolgversprechend, wenn angenommen werden kann, daß die schließlich erreichten Kosten im Vergleich zu den anderen Läufen nicht gut sind. Es ist natürlich sehr einfach, bestimmte Läufe als nicht erfolgversprechend einzustufen, wenn die Endergebnisse der einzelnen Läufe bekannt sind. Erheblich schwieriger ist es, ‘vorherzusagen’ welche Läufe schlechte und welche gute Ergebnisse erzielen werden. Dies soll hier geschehen, indem angenommen wird, daß die Läufe, die nach einer gewissen Iterationszeit schlechte Ergebnisse erzielen auch schlechte Endergebnisse erzielen.

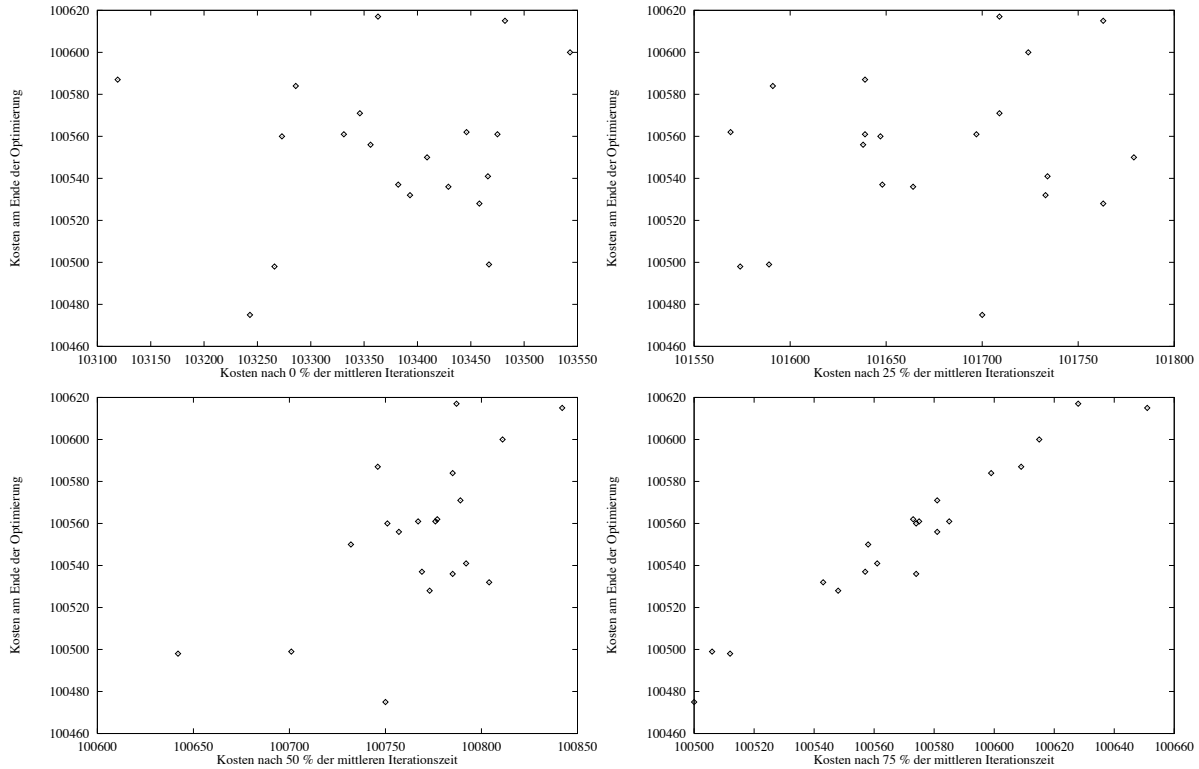


Bild 3.6: Gegenüberstellung der Kosten nach einem Anteil der gesamten Iterationszeit mit den schließlich erreichten Kosten für 20 Läufe.

Diese Vorgehensweise stellt natürlich nur eine Heuristik dar, die nicht unbedingt auf jedes Optimierungsproblem angewendet werden kann. Geprüft werden kann die Anwendbarkeit dieser Heuristik durch Gegenüberstellen der Kosten nach einem Anteil der Iterationszeit mit den Kosten am Ende der Optimierung für eine Menge von Läufen. Wenn sich ein näherungsweise linearer Zusammenhang (siehe Bild 3.6, rechts unten) ergibt, dann ist schon nach einem Anteil der Iterationszeit weitgehend sicher, daß der zu diesem Zeitpunkt beste Lauf auch das beste Endergebnis erzielt. Es ist allerdings auch möglich (siehe Bild 3.6, links oben), daß kein linearer Zusammenhang entsteht. Je nach Stärke der Korrelation kann man davon ausgehen, daß nach einem Anteil der Iterationszeit eine bestimmte Zahl von Läufen nicht mehr weiterverfolgt werden muß. Durch solche Diagramme kann man also prüfen, ob eine multidirektionale Suche mit Beschneidung möglich ist.

Je früher die Beschneidung erfolgen kann, desto größer ist natürlich die gesamte Rechenzeiterparnis. Wenn für eine große Zahl von Läufen nach der Hälfte der nötigen Iterationen nur noch der zu diesem Zeitpunkt beste Lauf weiterverfolgt wird, so werden fast 50% der Rechenzeit eingespart. Die *Beschneidungstabelle* legt fest, wieviel Prozent b_i der Läufe nach einem Anteil r_i der gesamten Iterationszeit beendet sein sollen.

Ein Beispiel für eine Beschneidungstabelle:

i	1	2	3	4
r_i	0.2	0.3	0.4	0.5
b_i	0.3	0.6	0.9	1.0

Es werden also alle Läufe bis zu 20% der insgesamt notwendigen Iterationszeit verfolgt und dann 30% beendet. Die restlichen 70% besten Läufe werden bis zu 30% der Iterationszeit weiterverfolgt und dann 60% der (anfänglichen) Läufe beendet und die 40% besten Läufe weiterverfolgt. Wenigstens soll jedoch immer ein Lauf weiterverfolgt werden.

Aus der Beschneidungstabelle läßt sich auch die eingesparte Rechenzeit abschätzen. Dabei soll angenommen werden, daß die Rechenzeit gleichmäßig auf alle Iterationsschritte verteilt ist. Wenn n Läufe normalerweise eine Rechenzeit von n Einheiten benötigen, dann ergibt sich für die Rechenzeit der parallelen Suche mit Beschneidung (mit $r_0 = 0$, $b_0 = 0$, $r_n = 1.0$):

$$\sum_{i=1}^n (r_i - r_{i-1}) \cdot \lceil n \cdot (1 - b_{i-1}) \rceil \quad (3.19)$$

Dabei ist $\lceil x \rceil$ die kleinste ganze Zahl, welche größer oder gleich x ist. Obige Beschneidungstabelle benötigt also die folgende Rechenzeit bei n Läufen:

$$0.2 \cdot n + 0.1 \cdot \lceil n \cdot 0.7 \rceil + 0.1 \cdot \lceil n \cdot 0.4 \rceil + 0.1 \cdot \lceil n \cdot 0.1 \rceil + 0.5 \cdot 1 \quad (3.20)$$

Bei 10 Läufen würde die normale Rechenzeit 10 Einheiten betragen. Bei paralleler Suche mit obiger Beschneidungstabelle benötigt man nur 3.7 Einheiten. Man erhält also eine Rechenzeiterparnis von 63%.

Bei hoher Beschneidungsrate nach kurzer Zahl der Iterationen ist es wahrscheinlicher, daß der optimale Lauf beendet wird. Allerdings können mit gleichem Rechenzeiteinsatz mehr Läufe gestartet werden. Im anderen Fall bei geringem Anteil der Beschneidung können nicht so viele Läufe gestartet werden. Der minimale Lauf wird allerdings weniger wahrscheinlich beendet. Die Bestimmung der Beschneidungstabelle soll hier anhand obiger Diagramme und durch Vergleich verschiedener Testläufe ermittelt werden.

3.3 Iterierte Zustandsraum-Reduktion

Es soll nun noch ein problemspezifisches Lösungsverfahren vorgestellt werden, welches auf der Idee beruht, ein schwieriges Problem mit großem Zustandsraum mehrfach zu verkleinern, und somit ein leichteres Problem zu erhalten.

Zur Erläuterung des Verfahrens ist es nötig, die Beschaffenheit des Zustandsraumes zu rekapitulieren (siehe Abschnitt 2.3). Es gilt eine Kostenfunktion auf der Menge der $|\mathcal{C}|$ -elementigen Kategoriensysteme eines Vokabulars zu optimieren. Als lokales Minimum werden diejenigen Kategoriensysteme bezeichnet, die im Rahmen der gewählten Nachbarschaft nicht mehr lokal verbessert werden können. Die Menge der lokalen Minima wird

mit Z_{lokmin} bezeichnet. Die vorgestellten iterativ-optimierenden Verfahren konvergieren normalerweise (nach ausreichender Abkühlung) in einem lokalen Minimum.

Die *iterierten Zustandsraum-Reduktion* beruht auf der Idee, daß gleiche Anteile in guten Kategoriensystemen in gewissem Sinne zusammengehörig sind und zu einer Einheit zusammengefaßt werden können. Dadurch wird der Zustandsraum verkleinert. In diesem verkleinerten Zustandsraum können effizienter gute Kategoriensysteme berechnet werden.

Im folgenden wird erläutert, wie eine Zustandsraum-Reduktion durchgeführt wird (siehe hierzu auch Bild 3.7). Es sei $\{\mathcal{C}_1, \dots, \mathcal{C}_n\} \subseteq Z_{lokmin}$ eine Menge von lokal minimalen Kategoriensystemen. Es sollen nun die in allen Kategoriensystemen $\mathcal{C}_1, \dots, \mathcal{C}_n$ gleichen Anteile bestimmt werden. Gesucht sind also diejenigen Mengen von Wörtern (im folgenden *Wortcluster* genannt), die in allen \mathcal{C}_i in der selben Kategorie sind. Dies entspricht der Bildung des Infimums² der Kategoriensysteme $\mathcal{C}_1, \dots, \mathcal{C}_n$:

$$\text{INF} = \inf(\mathcal{C}_1, \dots, \mathcal{C}_n) = \mathcal{C}_1 \sqcap \dots \sqcap \mathcal{C}_n \quad (3.22)$$

Die Elemente von INF sind die gesuchten Wortcluster. Es kann nun ein neues Optimierungsproblem über die Wortcluster gebildet werden, indem die Nachbarschaft geändert wird: Zwei Zustände gelten nun als benachbart, wenn sie sich nur in der Kategorienzugehörigkeit eines Wortclusters unterscheiden. In Abschnitt 2.3 war der Nachbarschaftsbegriff darauf beschränkt, daß zwei Kategoriensysteme dann benachbart sind, wenn sie sich in der Kategorienzugehörigkeit genau eines Wortes unterscheiden. Durch INF wird also eine neue Nachbarschaft definiert. Wenn wenigstens zwei Wörter zu einem Wortcluster zusammengefaßt werden können, dann gilt $|\text{INF}| < |\mathcal{W}|$. Während der ursprüngliche Zustandsraum $S(|\mathcal{W}|, |\mathcal{C}|)$ Elemente besaß (siehe Abschnitt 2.3), hat der neue Zustandsraum (bei $|\mathcal{C}| \ll |\mathcal{W}|$) dann nur $S(|\text{INF}|, |\mathcal{C}|) < S(|\mathcal{W}|, |\mathcal{C}|)$ Elemente³. In diesem kleineren Zustandsraum können effizienter lokale Minima berechnet werden. Es wird sich herausstellen, daß diese Kategoriensysteme von hoher Qualität sind. Man kann dies dadurch erklären, daß der Zustandsraum auf ein Teilgebiet eingeschränkt wurde, in dem geringe Kosten zu erwarten sind.

Der prinzipielle Ablauf der iterierten Zustandsraum-Reduktion ist im Struktogramm in Bild 3.8 dargestellt. Dabei ist $\text{OPT}(\mathcal{C}_i, \text{INF})$ das Ergebnis des gewählten probabilistischen Optimierungsverfahrens angewendet auf das Kategoriensystem \mathcal{C} mit der durch INF bestimmten Nachbarschaft. Es ist $\text{OPT}(\cdot, \text{INF})$ ein neu berechnetes Kategoriensystem mit der durch INF bestimmten Nachbarschaft. Wenn die Optimierung mit der ‘normalen’ Wort-Nachbarschaft aus Abschnitt 2.3 vollzogen wird, dann wird dies mit $\text{OPT}(\mathcal{C}_i, \mathcal{W})$ und $\text{OPT}(\cdot, \mathcal{W})$ bezeichnet.

²Das Infimum eines Kategoriensystems (= Partition) ist die gröbste Partition, die feiner als alle an der Infimum-Bildung beteiligten Kategoriensysteme (Partitionen) ist (siehe auch [Mül87]). Formal ergibt sich das Infimum zweier Partitionen:

$$\mathcal{C} \sqcap \mathcal{C}' = \inf(\mathcal{C}, \mathcal{C}') = \{c \cap c' \mid c \in \mathcal{C} \wedge c' \in \mathcal{C}' \wedge c \cap c' \neq \emptyset\} \quad (3.21)$$

Entsprechend ergibt sich das Infimum von n Kategoriensystemen (Partitionen).

³Für Rechenregeln mit Stirling-Zahlen siehe [Gra94].

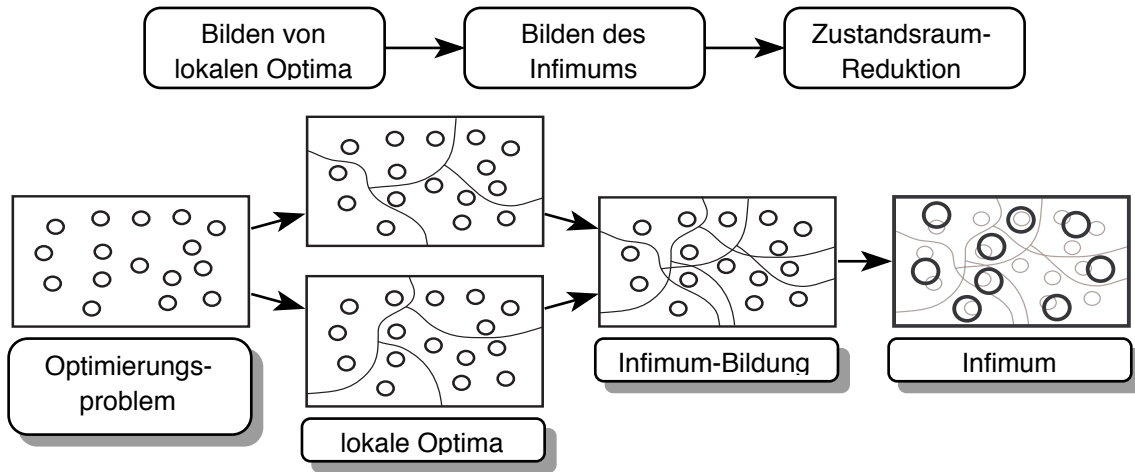


Bild 3.7: Veranschaulichung der Zustandsraum-Reduktion.

Für das Optimierungsproblem werden durch ein probabilistisches iterativ-optimierendes Verfahren n_0 Kategoriensysteme $\mathcal{C}^{(0)}$ berechnet. Hierfür kann eines der in Abschnitt 3.1 vorgestellten Verfahren verwendet werden. Welches Verfahren am besten geeignet ist, weil es besonders gute Kategoriensysteme berechnet, wird in Kapitel 5 ermittelt. In einer Schleife wird dann wiederholt eine Zustandsraum-Reduktion vorgenommen. Die Schleife wird beendet, wenn die Zahl der Wortcluster der Zahl der Kategorien entspricht. Wenn nämlich $|\text{INF}^{(t)}| = |\mathcal{C}|$ gilt, dann besteht der Zustandsraum aus einem einzigen Element, welches somit die Lösung ist. In der Schleife werden die besten Kategoriensysteme $\mathcal{C}_{b_1}^{(t)}, \dots, \mathcal{C}_{b_m}^{(t)}$ ausgewählt und von diesen das Infimum $\text{INF}^{(t)}$ gebildet. Dabei wird darauf geachtet, daß nur so viele Elemente am Infimum beteiligt werden, daß ein echt kleinerer Zustandsraum entsteht. Es ist damit ein neues Optimierungsproblem entstanden, in welchem zwei Zustände als benachbart gelten, wenn sie sich in der Klassenzugehörigkeit eines einzigen Wortclusters aus $\text{INF}^{(t)}$ unterscheiden. Die schon berechneten Kategoriensysteme $\mathcal{C}_{b_i}^{(t)}$ sind im Rahmen der neuen Nachbarschaft möglicherweise nicht mehr lokal optimal und es wird versucht sie zu verbessern. Es wird auch eine Menge von neuen Kategoriensystemen erzeugt. Von diesen Kategoriensystemen werden in der Schleife nun wieder die besten n ausgewählt, eine Zustandsraum-Reduktion durchgeführt, usw. Nach Beendigung der Schleife wird das beste Kategoriensystem \mathcal{C}_b (aus $\mathcal{C}^{(t)}$) ausgewählt. Dieses ist möglicherweise im Rahmen der Wort-Nachbarschaft nicht mehr lokal optimal, und es wird versucht, dieses noch weiter zu verbessern. Das Ergebnis ist dann das dadurch entstehende Kategoriensystem \mathcal{C}_{best} .

Das Verfahren besitzt neben der Auswahl des probabilistischen Optimierungsverfahrens noch mehrere Freiheitsgrade. Es muß die Zahl n der an der Zustandsraum-Reduktion beteiligten Kategoriensysteme festgelegt werden. Wenn n groß gewählt wird, dann werden nur wenige Wortcluster mit mehr als einem Wort entstehen. Wenn n klein gewählt wird, dann werden große Wortcluster entstehen. Außerdem muß noch die Zahl der jeweils neu zu bestimmenden Kategoriensysteme $n_t - n$ bestimmt werden. Es soll vorgeschlagen

$t := 0$	
Bildung von n_0 Kategoriensystemen durch ein probabilistisches Optimierungsverfahren: $\mathcal{C}^{(0)} := \{\mathcal{C}_1^{(0)} := \text{OPT}(\cdot, \mathcal{W}), \dots, \mathcal{C}_{n_0}^{(0)} := \text{OPT}(\cdot, \mathcal{W})\}$	
	$m := n + 1$
	$m := m - 1$
	Auswahl der besten m Kategoriensysteme $\mathcal{C}_{b_1}^{(t)}, \dots, \mathcal{C}_{b_m}^{(t)}$ aus $\mathcal{C}^{(t)}$
	Bildung des Infimums $\text{INF}^{(t)}$: $\text{INF}^{(t)} := \mathcal{C}_{b_1}^{(t)} \sqcap \dots \sqcap \mathcal{C}_{b_m}^{(t)}$
UNTIL $(t = 0 \wedge \mathcal{W} > \text{INF}^{(t)}) \vee (t > 0 \wedge \text{INF}^{(t-1)} > \text{INF}^{(t)})$	
Optimierung der bisherigen Kategoriensystemen mit der durch $\text{INF}^{(t)}$ bestimmten Nachbarschaft: $\forall i : 1 \leq i \leq m : \mathcal{C}_i^{(t+1)} := \text{OPT}(\mathcal{C}_{b_i}^{(t)}, \text{INF}^{(t)})$	
Bildung von $n_{t+1} - m$ neuen Kategoriensystemen: $\forall i : m + 1 \leq i \leq n_{t+1} : \mathcal{C}_i^{(t+1)} := \text{OPT}(\cdot, \text{INF}^{(t)})$	
$\mathcal{C}^{(t+1)} := \{\mathcal{C}_1^{(t+1)}, \dots, \mathcal{C}_{n_{t+1}}^{(t+1)}\}$	
$t := t + 1$	
UNTIL $ \text{INF}^{(t)} = \mathcal{C} $	
Auswahl des besten Kategoriensystems \mathcal{C}_b aus $\mathcal{C}^{(t)}$	
Optimierung von \mathcal{C}_b mit Wort-Nachbarschaft: $\mathcal{C}_{best} := \text{OPT}(\mathcal{C}_b, \mathcal{W})$	
Ergebnis: \mathcal{C}_{best}	

Bild 3.8: Iterierte Zustandsraum-Reduktion.

werden, daß für jeden Schleifendurchlauf der gleiche Rechenaufwand zur Verfügung gestellt wird, wodurch sich dann n_t automatisch ergibt. Da die Problemgröße im Laufe der Zustandsraum-Reduktion monoton abnimmt, wird der Rechenaufwand für einen Optimierungslauf immer geringer und n_t wird mit steigendem t größer.

Eine einzelne Zustandsraum-Reduktion findet die *stabilen Anteile* von Kategoriensystemen. Man kann sie also als empirisches Verfahren zum Testen der Stabilität der mit einem iterativ-optimierenden Verfahren bestimmten Kategoriensysteme ansehen. Die iterierte Zustandsraum-Reduktion kann auch als hierarchische Clusteranalyse ([Muc92]) angesehen werden. Bei jeder Iteration entsteht eine gröbere Partition der Wörter. Im Anhang A sind die stabilen Anteile eines Korpus abgedruckt.

Die Idee zu diesem Verfahren entstand durch Überlegungen zu genetischen Algorithmen ([Hol75, Boo89]), welche häufig und mit Erfolg für kombinatorische Optimierung eingesetzt werden. Diese Verfahren basieren auf den aus der Evolution bekannten Prinzipien der *Selektion*, der *Rekombination* und der *Mutation* ([Fis94]). Die Suche nach einem sinnvollen Rekombinations-Operator führte auf die hier beschriebene Infimum-Bildung. Nach jedem

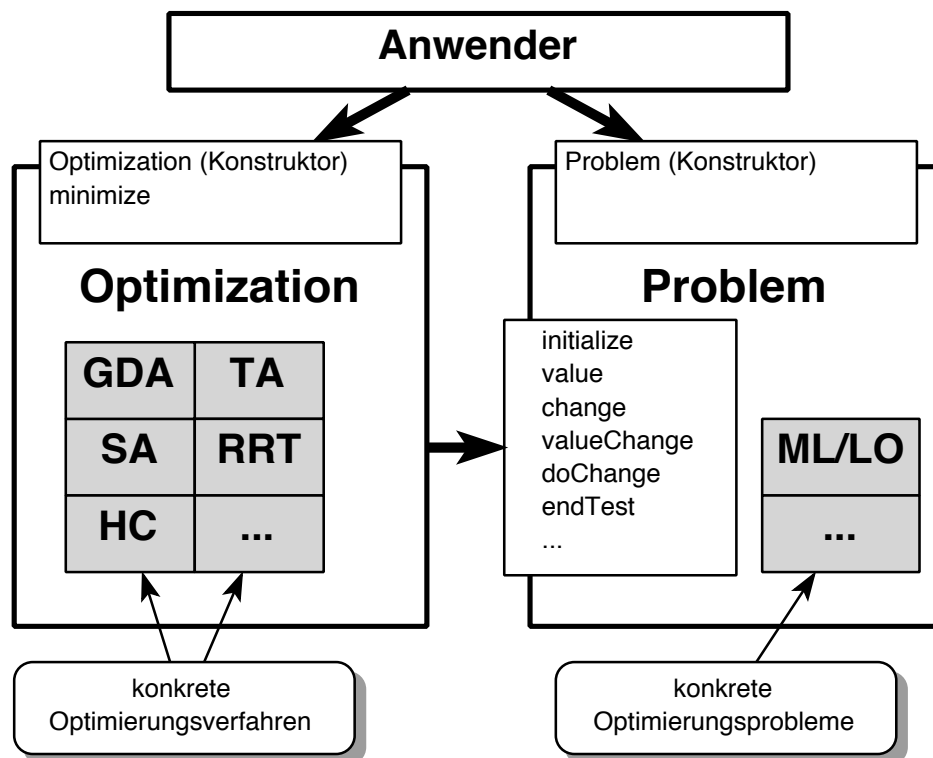


Bild 3.9: Prinzipielle Struktur der Implementierung mit Trennung von Optimierungsverfahren (Optimization) und Optimierungsproblem (Problem).

Iterationsschritt (im Algorithmus 3.8) findet eine Selektion der besten Kategoriensysteme statt. Es wird allerdings kein einfacher Mutations-Operator verwendet, sondern es werden unter Verwendung iterativ-optimierender Verfahren neue Kategoriensysteme erzeugt.

3.4 Implementierung

In diesem Abschnitt wird die objektorientierte Implementierung der iterativ-optimierenden Verfahren und der multidirektionalen Suche mit Beschneidung vorgestellt. Die Darstellung beschränkt sich dabei auf allgemeine Prinzipien. Die Implementierung erfolgte in C++ unter Verwendung der Klassenbibliothek NIHCL ([Gor90]).

Für die vorgestellten Optimierungsverfahren (bis auf iterierte Zustandsraum-Reduktion) bietet sich eine objektorientierte Implementierung an. Dies kann man schon aus der Darstellung der Verfahren in den Abschnitten 3.1 und 3.2 entnehmen, die weitgehend ohne Bezugnahme auf das konkrete Optimierungsproblem erfolgen konnte.

Ein Grundpfeiler der Implementierung stellt die Trennung von Optimierungsproblem und Optimierungsverfahren dar. Es ist das Ziel, daß bei der Implementierung eines neuen Problems keine Änderung am Code der Verfahren vorgenommen werden muß. Jedes neue Problem muß nur ein bestimmtes Methodenprotokoll implementieren, damit eine Opti-

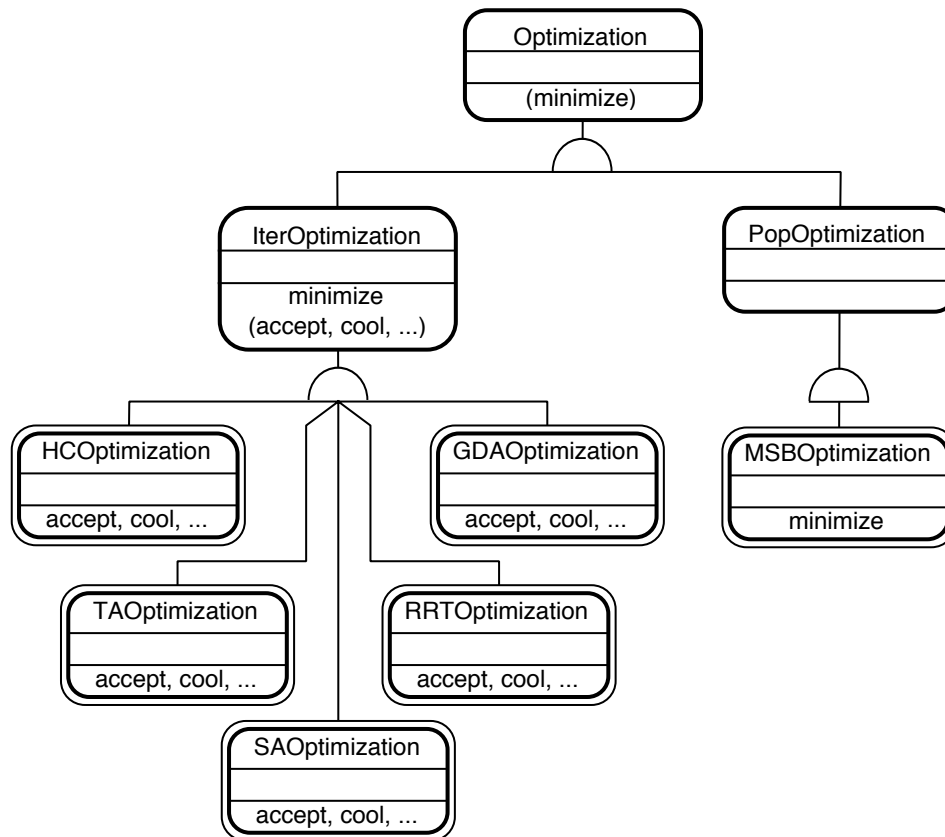


Bild 3.10: Klassenhierarchie für die iterativ-optimierenden Verfahren und für multidirektionale Suche mit Beschneidung (rein virtuelle Funktionen sind in Klammern angegeben).

rung vorgenommen werden kann. Es ergibt sich die prinzipielle Struktur in Bild 3.9. Dabei stellen **Optimization** und **Problem** die abstrakten Basisklassen für alle Optimierungsverfahren und Optimierungsprobleme dar.

Die Klasse **Optimization** ist eine abstrakte Basisklasse. Es können also keine Instanzen erzeugt werden. Sie spezifiziert das Protokoll für den Anwender. Neben dem in C++ obligatorischen Konstruktor besteht das Protokoll aus der Methode `minimize`. Durch Deklaration dieser Methoden als *reine virtuelle Funktionen*⁴ ist sichergestellt, daß alle konkreten abgeleiteten Klassen von **Optimization** diese implementieren. Der Benutzer der Optimierungsverfahren kann also im allgemeinen unter Verwendung des Konstruktors und der Methode `minimize` eine Minimierung durchführen.

Die Klasse **Problem** ist die abstrakte Basisklasse für Optimierungsprobleme. Welche Methoden von einem konkreten Optimierungsproblem implementiert werden müssen, ergibt sich weitgehend aus dem Diagramm 3.2 für den iterativ-optimierenden Algorithmus mit

⁴Eine reine virtuelle Funktion ist in C++ eine Funktion, die von abgeleiteten Klassen nicht nur redefiniert werden kann (virtuelle Funktion), sondern redefiniert werden muß ([Str92]).

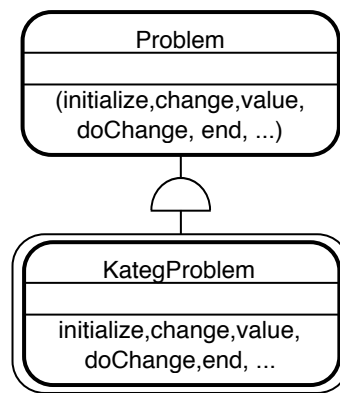


Bild 3.11: Klassenhierarchie für Optimierungsprobleme (rein virtuelle Funktionen sind in Klammern angegeben).

lokaler Auswertung. Diese Methoden werden in **Problem** wieder als reine virtuelle Funktionen deklariert und müssen somit von jeder (konkreten) abgeleiteten Klasse implementiert werden.

Eine abgeleitete (konkrete) Klasse von **Optimization** — also ein konkretes Optimierungsverfahren — muß sich genau dieses Protokolls bedienen. Dadurch ist es möglich, daß zur Lösung eines Optimierungsproblems alle allgemeinen Optimierungsverfahren eingesetzt werden können und umgekehrt zum Testen eines Verfahrens alle implementierten Optimierungsprobleme verwendet werden können.

In Bild 3.10 ist die implementierte Klassenhierarchie für die Optimierungsverfahren dargestellt. Die Klasse **Optimization** ist von der NIHCL-Klasse **Object** abgeleitet und ist somit in die NIHCL-Klassenbibliothek eingebunden. Es wurden die zwei (abstrakten) Klassen **IterOptimization** und **PopOptimization** von **Optimization** abgeleitet.

Die abstrakte Klasse **IterOptimization** ist die Basisklasse für iterativ-optimierende Verfahren und implementiert damit die Methode **minimize** entsprechend der Beschreibung in Bild 3.2. Die Methoden **accept**, **cool**, **init** und **end** werden als reine virtuelle Methoden deklariert. Damit muß jede der abgeleiteten Klassen **HCOptimization**, **SAOptimization**, **TAOptimization**, **RRTOptimization** und **GDAOptimization** diese Methoden implementieren.

Die abstrakte Klasse **PopOptimization** ist die Basisklasse für Optimierungsverfahren, die eine Menge (Population) von Optimierungsproblemen verwenden. Sie beinhaltet Methoden und Datenstrukturen zur Verwaltung einer Menge von Instanzen von **Problem**. Man könnte sich vorstellen, daß genetische Algorithmen von dieser Klasse abgeleitet werden. Hier erbt jedoch nur die Klasse **MSBOptimization**, in der die multidirektionale Suche mit Beschneidung implementiert ist.

In Bild 3.11 ist die Klassenhierarchie für die Optimierungsprobleme dargestellt. In **Problem** werden die Methoden spezifiziert, die von jedem konkreten Optimierungsproblem implementiert werden müssen. Die einzige Klasse, die momentan davon abgeleitet

ist, ist `KategProblem`. Diese beinhaltet das in dieser Arbeit behandelte Optimierungsproblem. Neue Optimierungsprobleme, wie das Problem des Handlungsreisenden, müßten entsprechend als abgeleitete Klassen von `Problem` implementiert werden.

3.5 Zusammenfassung

Es wurden verschiedene Optimierungsverfahren vorgestellt. Der ähnliche Ablauf der einzelnen iterativ-optimierenden Verfahren ermöglichte die Formulierung eines allgemeinen Algorithmenschemas. Die fünf konkreten Optimierungsverfahren stochastische Relaxation, simuliertes Ausfrieren, Sinflutalgorithmus, Schwellwertakzeptanz und Record-To-Record Travel unterscheiden sich hauptsächlich in Abkühlplan und Annahmekriterium.

Stochastische Relaxation ist das einzige Verfahren ohne Abkühlung. Für die anderen Verfahren müssen Parameter festgelegt werden, welche den Abkühlplan bestimmen. Es wurden die ursprünglichen Parameter bei GDA, SA, TA und RRT durch solche ersetzt, die eher unabhängig von der Größe des Optimierungsproblems sind. Dies geschah in der Hoffnung, daß eine einmalige gute Einstellung dieser Parameter für ein Optimierungsproblem dann unabhängig von der Problemgröße verwendet werden kann. In Abschnitt 5.1 werden diese Parameter bestimmt.

Mit multidirektionaler Suche mit Beschneidung wurde eine Heuristik vorgestellt, die auf alle iterativ-optimierenden Verfahren anwendbar ist. Mit diesem Verfahren soll die Effizienz von simultaner Suche im Zustandsraum gesteigert werden. Es beruht auf der Annahme, daß schon frühzeitig bestimmte Optimierungsläufe als nicht erfolgversprechend eingestuft werden können.

Mit der iterierten Zustandsraum-Reduktion wurde ein problemspezifisches Optimierungsverfahren vorgestellt. Dabei werden die gleichen Anteile aus verschiedenen Kategoriensystemen dazu verwendet, eine Zustandsraum-Reduktion vorzunehmen. In dem verkleinerten Zustandsraum lassen sich nun effizienter neue gute Kategoriensysteme berechnen, die dann zu einer erneuten Zustandsraum-Reduktion eingesetzt werden können.

Die iterativ-optimierenden Verfahren und die multidirektionale Suche mit Beschneidung sind allgemeine Optimierungsverfahren, die auf viele Optimierungsprobleme angewendet werden können. Unter anderem aus diesem Grunde ist es möglich, eine objektorientierte Implementierung vorzunehmen. Der objektorientierte Entwurf sieht eine strikte Trennung von Optimierungsverfahren und Optimierungsproblem vor. Dadurch ist es möglich, daß Optimierungsverfahren und Optimierungsprobleme unabhängig voneinander entwickelt, und beliebig kombiniert werden können.

Nach Kapitel 2 (Optimierungsproblem) und Kapitel 3 (Optimierungsverfahren) erfolgt im Kapitel 4 deren Verknüpfung. Anschließend werden in Kapitel 5 die Optimierungsverfahren hinsichtlich Rechenaufwand und Lösungsqualität verglichen.

Kapitel 4

Implementierung des Optimierungsproblems

In diesem Kapitel sollen die Grundzüge der Implementierung des Optimierungsproblems vorgestellt und verschiedene Variationsmöglichkeiten verglichen werden. Die Optimierungskriterien ML und LO aus Kapitel 2 werden so formuliert, daß sie inkrementell ausgewertet werden können, und sich für eine effiziente Implementierung im Rahmen von iterativ-optimierenden Verfahren eignen. Verschiedene deterministische und probabilistische Möglichkeiten für die Initialisierung und die Auswahl des Folgezustandes werden verglichen.

Es gibt sehr viele verschiedene Variationsmöglichkeiten für die Lösung eines Optimierungsproblems: Initialisierung, Nachbarschaft, Optimierungsverfahren, Parameter. Um die beste Einstellung zu finden, müßte man theoretisch alle Kombinationen miteinander vergleichen. Die Zahl der Kombinationen ist jedoch zu groß und es muß eine Auswahl getroffen werden. Die getroffene Auswahl ist anhand vieler Versuche und Testläufe entstanden, die keinesfalls alle in dieser Arbeit vorgestellt werden können, aber dem Autor die entscheidende Hilfestellung für die vorgenommenen Einstellungen gegeben haben.

4.1 Auswertung der Optimierungskriterien

Beim Ablauf eines Optimierungsverfahrens müssen die Optimierungskriterien ML (Gleichung (2.15)) oder LO (Gleichung (2.28)) sehr häufig ausgewertet werden. Dies muß also sehr effizient geschehen. Bei einer **vollständigen Berechnung** benötigt man die Häufigkeiten für Kategorien-Unigramme $n(c)$ und die Häufigkeiten für Kategorien-Bigramme $n(c, c')$. Diese kann man durch Abzählen im Korpus mit der Komplexität $\mathcal{O}(m)$ berechnen, wobei m die Zahl der Wörter im Korpus bezeichne. Die Auswertung der Doppelsummen (2.15) und (2.28) besitzt dann noch die Komplexität $\mathcal{O}(|\mathcal{C}|^2)$.

Es ist jedoch möglich eine erheblich effizientere **inkrementelle Berechnung** vorzunehmen. Wenn der Wert eines Kriteriums für eine bestimmte Partition bekannt ist und sich die Partition nur geringfügig ändert, dann stellt sich heraus, daß die meisten Terme bei beiden Optimierungskriterien konstant bleiben und somit nicht mehrfach berechnet werden

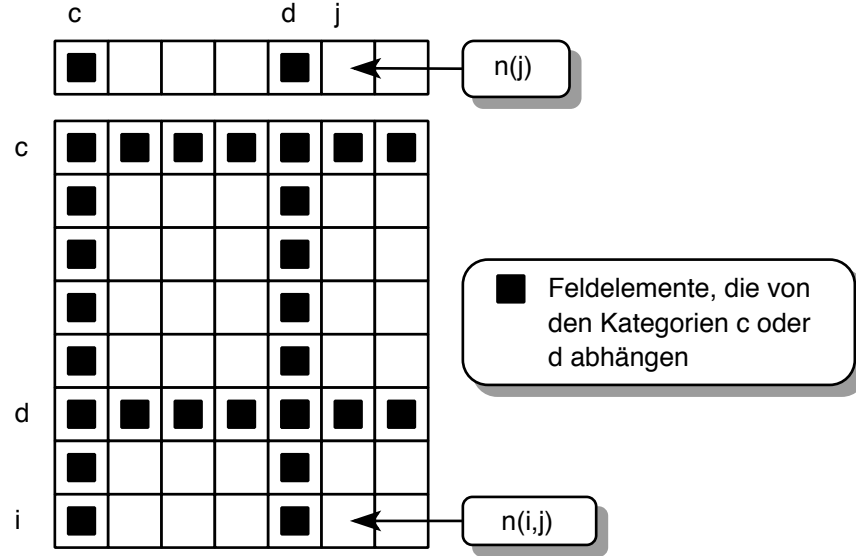


Bild 4.1: Visualisierung der inkrementellen Auswertung des Optimierungskriteriums ML.

müssen. Die Doppelsummen in den Optimierungskriterien summieren über ein zweidimensionales Feld der Ausdehnung $|\mathcal{C}| \cdot |\mathcal{C}|$. Das Element in Zeile c und Spalte d entspricht $n(c, d)$. Die Partitionen \mathcal{C} und \mathcal{C}' unterscheiden sich nur in den zwei Kategorien $c, d \in \mathcal{C}$ bzw. $c', d' \in \mathcal{C}'$. In diesem Falle unterscheiden sich die zugehörigen Häufigkeitsarrays nur in den Termen, die von den beiden Kategorien c und d abhängen (siehe Bild 4.1).

Es gilt also für die Differenz $ML' - ML$:

$$\Delta ML := ML' - ML = ML(c', d') - ML(c, d) \quad (4.1)$$

Dabei soll $ML(c, d)$ den Anteil von ML bezeichnen, der von den Kategorien c und d abhängig ist. Dieser Anteil entspricht offenbar den vier gekennzeichneten Streifen in Bild 4.1. Es ergibt sich mit der Indexmenge $I_{c,d} = \{(i, j) | i = c \vee i = d \vee j = c \vee j = d\}$, welche den abhängigen Anteil indiziert:

$$ML(c, d) = 2n(c) \log(n(c)) + 2n(d) \log(n(d)) - \left(\sum_{(i,j) \in I_{c,d}} n(i, j) \log(n(i, j)) \right) \quad (4.2)$$

Durch Einsetzen von $ML(c, d)$ und $ML(c', d')$ in Gleichung (4.1) erhält man:

$$\Delta ML = 2(n(c') \log(n(c')) - n(c) \log(n(c))) + 2(n(d') \log(n(d')) - n(d) \log(n(d))) - \left(\sum_{(i,j) \in I_{c,d}} n(i', j') \log(n(i', j')) - n(i, j) \log(n(i, j)) \right) \quad (4.3)$$

Der Aufwand zur Berechnung dieser Differenz ist erheblich geringer als der Aufwand $\mathcal{O}(|\mathcal{C}|^2)$ zur vollständigen Auswertung. Unter Verwendung von $n(i, j)$, $n(c)$ und $n(d)$ kann $\text{ML}(c, d)$ mit der Komplexität $\mathcal{O}(2 + |I_{c,d}|) = \mathcal{O}(2 + 4|C| - 4) = \mathcal{O}(|C|)$ berechnet werden. Entsprechendes gilt für $\text{ML}(c', d')$.

Für das Kriterium LO ergibt sich mit der Vereinbarung $\forall x \leq 0 \log(x) = 0$:

$$\begin{aligned} \Delta\text{LO} = & 2(n(c') \log(n(c') - 1) - n(c) \log(n(c) - 1)) \\ & + 2(n(d') \log(n(d') - 1) - n(d) \log(n(d) - 1)) \\ & - \left(\sum_{(i,j) \in I_{c,d}} n(i', j') \log(n(i', j') - 1 - \rho) - n(i, j) \log(n(i, j) - 1 - \rho) \right) \\ & - \left(\eta'_1 \log\left(\frac{\eta'_+ - 1}{\eta'_0 + 1} \rho\right) + \eta_1 \log\left(\frac{\eta_+ - 1}{\eta_0 + 1} \rho\right) \right) \end{aligned} \quad (4.4)$$

Man benötigt wie zur Berechnung von ΔML die Häufigkeiten $n(c', d')$ und $n(c')$. Zusätzlich müssen jedoch η_0, η_1, η_+ und $\eta'_0, \eta'_1, \eta'_+$ ermittelt werden. Diese erhält man durch einfaches Abzählen in den Häufigkeitsstatistiken mit derselben Komplexität wie ΔLO . Beispielsweise erhält man η'_0 durch:

$$\eta'_0 = \eta_0 + \sum_{(i,j) \in I_{c,d}} ([n(i', j') = 0] - [n(i, j) = 0]) \quad (4.5)$$

Die Auswertung von ΔLO besitzt also ebenfalls die Komplexität $\mathcal{O}(|\mathcal{C}|)$, wenn die Häufigkeiten $n(\cdot)$ bekannt sind.

Bisher wurde ganz allgemein von einer auf zwei Kategorien beschränkten Änderung gesprochen. Es soll nun der Transport eines Wortes w von seiner Kategorie $C(w)$ in eine andere Kategorie genauer untersucht werden. Dabei müssen die Häufigkeiten $n(i', j')$ und $n(i')$ möglichst effizient aus den Häufigkeiten $n(i, j)$ und $n(i)$ ermittelt werden. Im folgenden sei mit $L(w)$ die Menge der Wörter bezeichnet, die links von w im Trainingskorpus stehen. Entsprechend sei $R(w)$ die Menge der Wörter, die rechts von w im Trainingskorpus stehen. Formal:

$$L(w) = \{w' | n(w', w) > 0\} \quad (4.6)$$

$$R(w) = \{w' | n(w, w') > 0\} \quad (4.7)$$

Der Transport eines Wortes w von Kategorie c nach Kategorie d führt zu veränderten Kategorien $c' = c \setminus \{w\}$ und $d' = d \cup \{w\}$. Es ergibt sich:

$$\begin{aligned} n(c', i') &= n(c \setminus \{w\}, i') &= n(c, i') - n(w, i') \text{ für } i' \neq c' \wedge i' \neq d' \\ n(c', c') &= n(c \setminus \{w\}, c \setminus \{w\}) &= n(c, c) - n(c, w) - n(w, c) + n(w, w) \\ n(c', d') &= n(c \setminus \{w\}, d \cup \{w\}) &= n(c, d) + n(c, w) - n(w, d) - n(w, w) \\ n(c') &= n(c \setminus \{w\}) &= n(c) - n(w) \end{aligned} \quad (4.8)$$

Entsprechendes gilt für $n(d', i')$, $n(i', c')$ und $n(i', d')$. Es werden die Häufigkeiten $n(w, i)$ und $n(i, w)$ benötigt. Die Berechnung von $n(w, i)$ bzw. $n(i, w)$ für alle Kategorien i benötigt

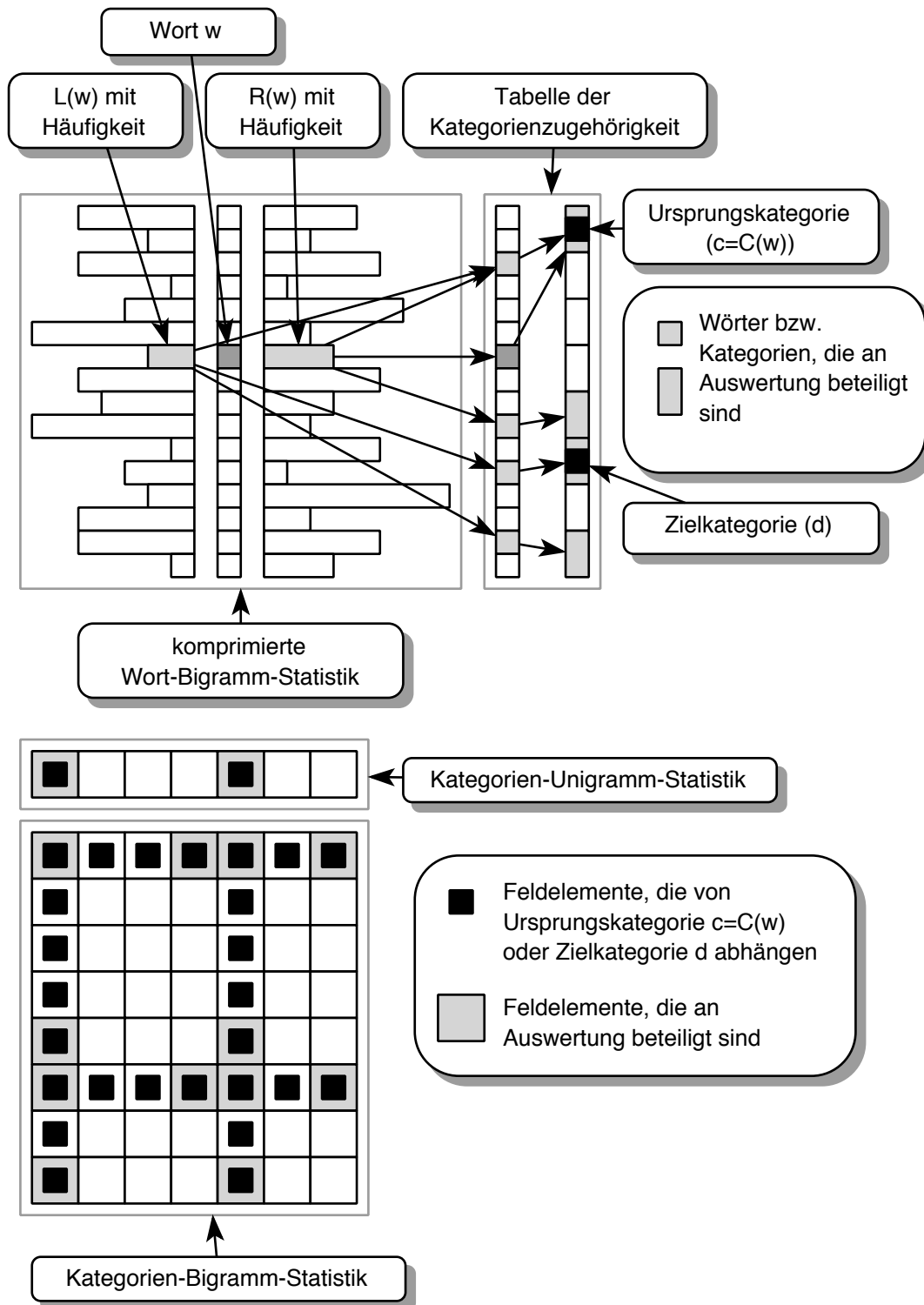


Bild 4.2: Transport eines Wortes zwischen zwei Kategorien.

bei effizienter Berechnung eine Zahl von Operationen, die proportional zur Kardinalität von $L(w)$ bzw. $R(w)$ ist. Die Berechnung von ΔML und ΔLO beim Transport eines Wortes besitzt also höchstens die Komplexität $\mathcal{O}(|\mathcal{C}| + |R(w)| + |L(w)|)$. Man kann die Auswertung jedoch noch effizienter gestalten. Wenn nämlich $n(w, i) = 0$ ist, dann gilt $n(c', i') = n(c, i)$ und der entsprechende Summand in den Gleichungen (4.3) bzw. (4.4) fällt weg. Es bleiben nur $|R(w)| + |L(w)|$ Summanden übrig und die Auswertung der Differenzformeln bei Transport eines Wortes besitzt im Mittel die Komplexität $\mathcal{O}(\overline{|R(w)| + |L(w)|})$. Dabei ist $\overline{|R(w)| + |L(w)|}$ der mittlere Wert von $|R(w)| + |L(w)|$ und somit eine Konstante des verwendeten Korpus. Bei allen im Rahmen dieser Arbeit verwendeten Korpora gilt $\overline{|L(w)|} = \overline{|R(w)|} < 10$. In [Kne93] wurde (ohne Begründung) eine Zeitkomplexität von $|\mathcal{C}|$ angegeben. Die hier vorgenommene Auswertung ist somit bei größeren Kategorienzahlen effizienter.

Die notwendigen und hinreichenden Anteile der Bigramm-Statistiken zur Berechnung der Kostendifferenz sind in Bild 4.2 dargestellt. Oben sieht man die komprimierte Wort-Bigramm-Statistik, in der die Häufigkeiten $n(w, w')$ der Bigramme (w, w') gespeichert sind. Bei einem Vokabular von $|\mathcal{W}|$ Wörtern könnte man hierzu einfach ein zweidimensionales Feld der Größe $|\mathcal{W}|^2$ verwenden. Dieses wäre jedoch nur spärlich mit Elementen $\neq 0$ besetzt. In der hier verwendeten Datenstruktur werden die Zeilen und Spalten des Feldes ohne Nullelemente gespeichert. Zu einem Wort w werden also alle seine Vorgänger $L(w)$ und seine Nachfolger $R(w)$ jeweils mit Häufigkeit gespeichert. Dadurch erzielt man eine hohe Speicherreduktion und einen effizienten Zugriff auf die interessierenden Häufigkeiten.

Es existiert eine Tabelle, die jedem Wort w seine Kategorien-Zugehörigkeit $C(w)$ zuordnet. Aus der Wort-Bigramm-Statistik $n(w, w')$ und der Kategorien-Zugehörigkeit $C(w)$ ergeben sich die Werte der Kategorien-Bigramm-Statistik $n(c, c')$, die als zweidimensionales Feld der Größe $|\mathcal{C}|^2$ abgelegt ist. Der Transport eines Wortes w von der Kategorie $c = C(w)$ in die Kategorie d geht nun folgendermaßen vonstatten. Alle Wörter mit denen das Wort w in einem Bigramm zusammen vorkommt (dies ist $L(w) \cup R(w)$) sind nun an der Auswertung beteiligt. Bildet man die Menge der Kategorien dieser Wörter, so erhält man die beteiligten Kategorien. Nur für diese Kategorien muß $n(c', d')$ bzw. $n(c')$ berechnet werden. Es ergibt sich also eine Teilmenge von beteiligten Feldelementen (c, d) , für die $n(c', d')$ berechnet werden muß. Die eigentliche Berechnung geschieht dann entsprechend der Formeln (4.3) bzw. (4.4) und (4.8).

Der Speicherplatzbedarf wird im wesentlichen durch die komprimierte Wort-Bigramm-Statistik und die Kategorien-Bigramm-Statistik bestimmt. Die Kategorien-Bigramm-Statistik besitzt $|\mathcal{C}|^2$ Einträge (jeweils 4 Byte). Für das Wort w gibt es in der komprimierten Wort-Bigramm-Statistik $|R(w)| + |L(w)|$ Einträge, also gibt es im Mittel $\overline{|R(w)| + |L(w)|}$ Einträge. Pro Eintrag muß die Häufigkeit (4 Byte) und ein Verweis auf das Wort (4 Byte) gespeichert werden. Es ergibt sich also näherungsweise (ohne den notwendigen Bedarf für die Speicherverwaltung) der Speicherplatzbedarf für die Wort-Bigramm-Statistik zu $|\mathcal{W}| \cdot \overline{|L(w)| + |R(w)|} \cdot 8$ Byte.

4.2 Initialisierung

Die iterativ-optimierenden Verfahren benötigen ein initiales Kategoriensystem, von dem aus die Optimierung beginnt. Oft wählt man den Startzustand eines Optimierungsverfahrens zufällig ([Gre87]). Es ist jedoch denkbar, daß bei einer geschickten Initialisierung bessere Lösungen oder effizienter gute Lösungen gefunden werden. Folgende Initialisierungen sollen verglichen werden:

1. **I-ALL-IN-ONE**: Alle Wörter in eine Kategorie. Diese Initialisierung resultiert in einem maximalen Wert für ML bzw. LO, da für jedes Wort w nur die Wahrscheinlichkeit der Kategorienzugehörigkeit $P(w|C(w))$ verwendet wird. Diese Initialisierung wird in [Jar93a] verwendet.
2. **I-FREQ**: Die häufigsten Wörter in jeweils eine Kategorie für sich; alle anderen Wörter in eine Kategorie ([Kne91, Kne93]). Dies geht von der Idee aus, daß die häufigsten Wörter am meisten das endgültige Kategoriensystem bestimmen und am ehesten in verschiedenen Kategorien liegen müssen.
3. **I-RAN**: Zufällige Aufteilung auf alle Kategorien: Dadurch erreicht man, daß selbst deterministische Optimierungsverfahren mehrmals aufgerufen werden können und immer ein anderes Ergebnis liefern.
4. **I-LW-RW**: Hier sollen die häufigsten Wörter in jeweils eine Kategorie für sich und außerdem die Mengen $L(w)$ bzw. $R(w)$ für die häufigsten Wörter zu einzelnen Kategorien zusammengefaßt werden. Dabei wird sichergestellt, daß die Disjunktheit der Kategorien erhalten bleibt, indem eine einmal festgelegte Kategorie für ein Wort nicht wieder verändert wird. Dies sollte geringe anfängliche Kosten liefern, da man annehmen kann, daß alle Wörter die vor oder nach einem häufigen Wort auftreten ähnliche Bigramm-Statistik haben. Es hat sich herausgestellt, daß eine Zuordnung der Hälfte der Kategorien für die häufigsten Wörter besonders geringe anfängliche Kosten liefert.
5. **I-OTHER**: Verwendung des Ergebnisses eines anderen Optimierungsverfahrens

Um nicht an Allgemeinheit zu verlieren, soll im Gegensatz zu [Kne93] kein weiteres linguistisches Wissen zur Initialisierung verwendet werden. Es werden nun die Möglichkeiten 1 bis 4 in Qualität des Endergebnisses und Effizienz verglichen. Dabei soll davon ausgegangen werden, daß dieser Vergleich unabhängig von dem konkreten iterativ-optimierenden Verfahren durchgeführt werden kann. Die folgenden Auswertungen sind mit stochastischer Relaxation auf dem Intercity-Korpus entstanden. Die Testumgebung mit den verwendeten Korpora wird näher in Abschnitt 6.1 beschrieben.

Die verschiedenen Initialisierungen unterscheiden sich insbesondere in der anfänglichen Perplexität. In Bild 4.3 wird die Perplexität der Initialisierungen für verschiedene Kategorienzahlen verglichen. Wie erwartet liefert die Initialisierung I-ALL-IN-ONE die schlechteste anfängliche Perplexität. Wenn alle Wörter in einer Kategorie sind, so entspricht dies einem

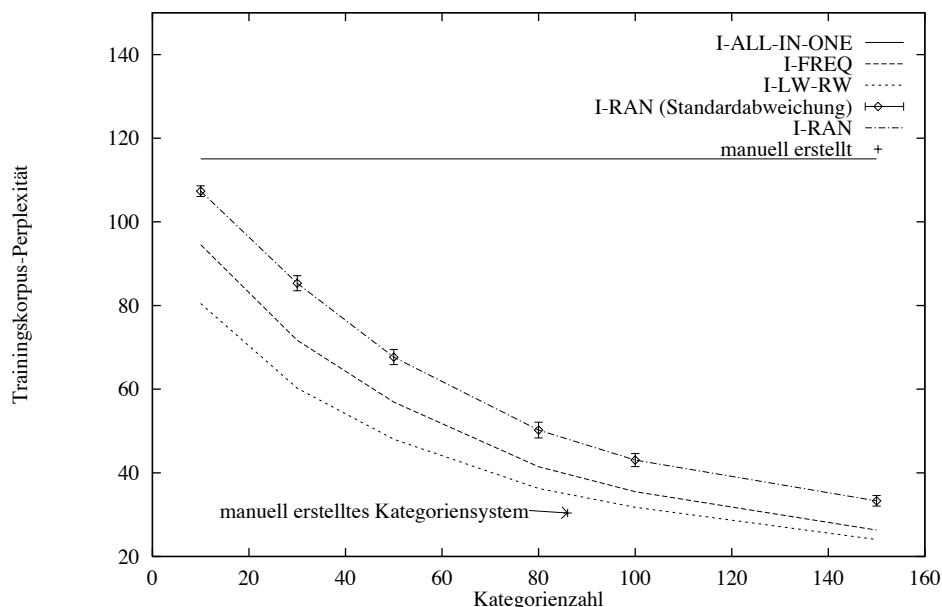


Bild 4.3: Vergleich der Trainingskorpus-Perplexität verschiedener Initialisierungen.

Sprachmodell, welches nur die Unigramm-Wahrscheinlichkeiten verwendet. Es ergibt sich die konstante Perplexität des Unigramm-Sprachmodells. Eine erheblich geringere Perplexität liefert die zufällige Initialisierung I-RAN. Die häufigsten Wörter in eine Kategorie für sich zu transportieren (I-FREQ) verbessert die Perplexität um weitere 15%. Die Initialisierung I-LW-RW verbessert dies noch mal um etwa 10% und liefert damit die besten initialen Kategoriensysteme. Als Vergleichswert ist noch die Trainingskorpus-Perplexität des manuell erstellten Kategoriensystems eingetragen. Ob die anfänglichen Kostenunterschiede einen Einfluß auf die Qualität des Endergebnisses besitzen, soll nun überprüft werden.

In Bild 4.4 sind die Endkosten der verschiedenen Initialisierungen verglichen. Jeder Balken gibt die Standardabweichung um den gekennzeichneten Mittelwert an. Es wurden jeweils 20 Testläufe mit probabilistischer Nachbarschaftsauswahl (siehe Abschnitt 4.3) durchgeführt. Dabei wurden jeweils Kategoriensysteme mit 80 Kategorien berechnet, da diese Kategorienzahl für das Intercity-Korpus eine besonders gute Testkorpus-Perplexität liefert (siehe Kapitel 6). Man sieht, daß die Initialisierung I-LW-RW geringfügig bessere Endergebnisse als die anderen Initialisierungen liefert. Ein ähnliches Ergebnis ergibt sich beim Verbmobil-Korpus (siehe Bild B.1 im Anhang B). Es konnte keine Beeinflussung der Laufzeiten durch die einzelnen Initialisierungen (bei probabilistischer Nachbarschaftsauswahl) festgestellt werden.

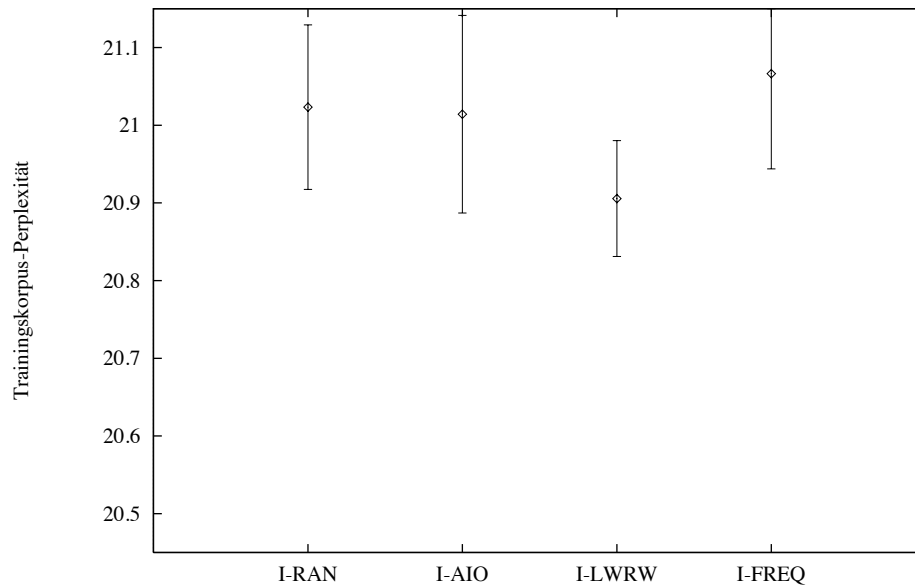


Bild 4.4: Vergleich der erreichten Perplexität verschiedener Initialisierungen (Intercity-Korpus, 80 Kategorien, probabilistische Nachbarschaftsauswahl (W-RAN-RAN)).

4.3 Nachbarschaft

Die Funktion **change** war im Rahmen des Algorithmenschemas für iterativ-optimierende Verfahren mit inkrementeller Auswertung dafür zuständig, eine Zustandsänderung zu erzeugen. Für die hier gewählte Nachbarschaft (siehe Abschnitt 2.3) muß also ein Wort w und eine Kategorie $c \neq C(w)$ gewählt werden, in die das Wort transportiert wird. Die Implementierung der Funktion **change** besitzt damit noch eine Reihe von Freiheitsgraden.

Es sollen folgende Möglichkeiten für die Auswahl des Wortes w und der Kategorie c verglichen werden:

- **W-RAN-RAN**: zufällige Auswahl des Wortes und der Kategorie ([Jar93a])
- **W-DET-DET**: deterministische Auswahl des Wortes und der Kategorie: Dabei sollen die Wörter nach absteigender Häufigkeit und die Kategorie in fester Reihenfolge ausgewählt werden. Durch den früheren Transport häufiger Wörter wird die Zahl der notwendigen Funktionsauswertungen geringfügig reduziert, da die häufigen Wörter am meisten das endgültige Kategoriensystem bestimmen, und durch das frühzeitige Transportieren in die ‘richtige’ Kategorie weniger Transporte durchgeführt werden müssen. Dies wurde in ([Kne93]) festgestellt und in dieser Arbeit nachvollzogen.

Durch diese deterministische Nachbarschaftsauswahl kann einfach überprüft werden, ob ein lokales Optimum erreicht wurde. Wenn nämlich bei einem Problem mit $|\mathcal{C}|$ Kategorien und $|\mathcal{W}|$ Wörtern während $|\mathcal{C}| \cdot |\mathcal{W}|$ Iterationsschritte keine Verbesserung

auftritt, dann existiert keine Möglichkeit durch Transport eines Wortes die Kostenfunktion zu verbessern (siehe auch Abschnitt 4.4).

- **W-RAN-BEST:** zufällige Auswahl des Wortes und Auswahl der für das gewählte Wort optimalen Kategorie: Bei dieser Auswahl wird also überprüft, wie sich die Kosten beim Transport des Wortes w in die einzelnen Kategorien $c \neq C(w)$ ändern und es wird diejenige Kategorie ausgewählt, welche die geringsten Kosten liefert.
- **W-DET-BEST:** deterministische Auswahl des Wortes und Auswahl der für das gewählte Wort optimalen Kategorie ([Kne91, Kne93])

Es sollen also zwei deterministische und zwei probabilistische Verfahren verglichen werden. Eine zufällige Komponente in der Nachbarschaftsauswahl hat — wie zufällige Initialisierung — den Effekt, daß selbst deterministische Optimierungsverfahren (wie HC, TA, RRT, GDA) mehrmals aufgerufen werden können und unterschiedliche Ergebnisse liefern. Von diesen Ergebnissen kann dann das Beste als endgültiges Kategoriensystem gewählt werden. Der Vergleich wird anhand von Testläufen mit stochastischer Relaxation durchgeführt. Es ist allerdings nicht offensichtlich, wie die iterativ-optimierenden Verfahren mit Abkühlung (SA, TA, RRT und GDA) auf die Nachbarschaftsauswahlen W-RAN-BEST und W-DET-BEST reagieren. Verschlechterungen treten nämlich prinzipiell nur dann auf, wenn das gewählte Wort w am besten in Kategorie $C(w)$ paßt. Die in einem solchen Fall zurückgelieferte Kategorie c ist dann diejenige, die am zweitbesten paßt. Dies wird später beim Vergleich der Optimierungsverfahren (Kapitel 5) berücksichtigt.

In Bild 4.5 (Bild B.2) wurden nun die erreichten Kosten für verschiedene Wortauswahlen und verschiedene Kategorieauswahlen verglichen. Die besten Ergebnisse werden mit deterministischer Wort- und Kategorieauswahl (W-DET-DET) erzielt. Am schlechtesten schneidet die zufällige Auswahl von Wort und Kategorie (W-RAN-RAN) ab. Die Überlegenheit der deterministischen Auswahl läßt sich dadurch erklären, daß aufgrund des Endkriteriums (siehe Abschnitt 4.4) bei dieser Auswahl sicher ein lokales Minimum gefunden wird, was bei probabilistischer Auswahl nicht der Fall ist. Anscheinend werden bei stochastischer Relaxation durch optimale Kategorieauswahl (W-DET-BEST) schlechtere Ergebnisse erzielt.

Wenn man jedoch die Anzahl der Funktionsauswertungen (siehe Bild 4.6 und Bild B.3) vergleicht, so schneidet W-DET-BEST am besten ab. Insbesondere die probabilistische Nachbarschaftsauswahlen (W-RAN-RAN und W-RAN-BEST) benötigen erheblich mehr Funktionsauswertungen.

Es läßt sich ein Schätzwert für die Zahl der (inkrementellen) Funktionsauswertungen S_{HC} anhand der Größe des Problems bestimmen.

$$S_{HC}(\text{W-DET-DET}) \approx |\mathcal{C}| \cdot |\mathcal{W}| \cdot s \quad (4.9)$$

Bei der Nachbarschaftsauswahl W-DET-DET gilt $s \approx 13$ und bei Auswahl der optimalen Kategorie (W-DET-BEST) gilt $s \approx 10$. Die Auswahl der optimalen Kategorie ist also effizienter.

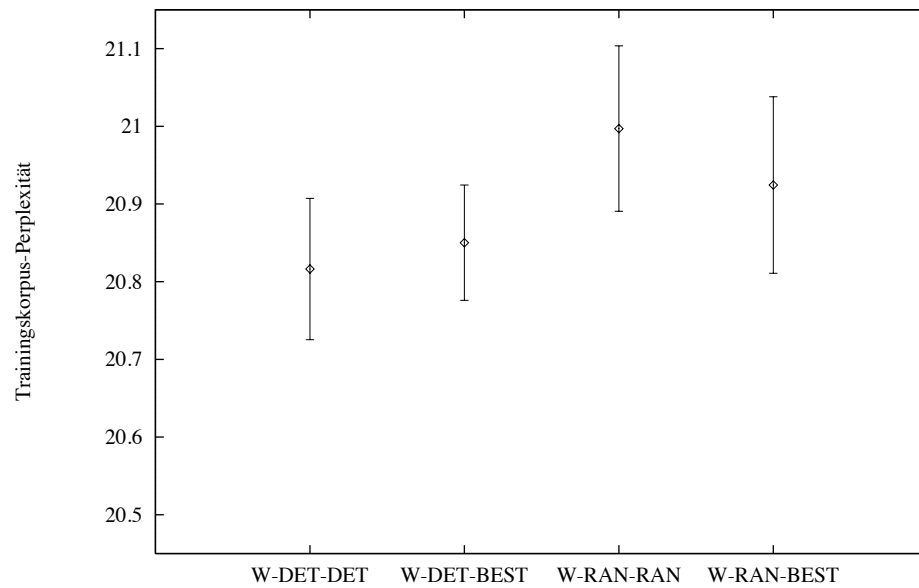


Bild 4.5: Vergleich der erreichten Perplexität verschiedener Nachbarschaftsauswahlen (Intercity-Korpus, 80 Kategorien, zufällige Initialisierung (I-RAN)).

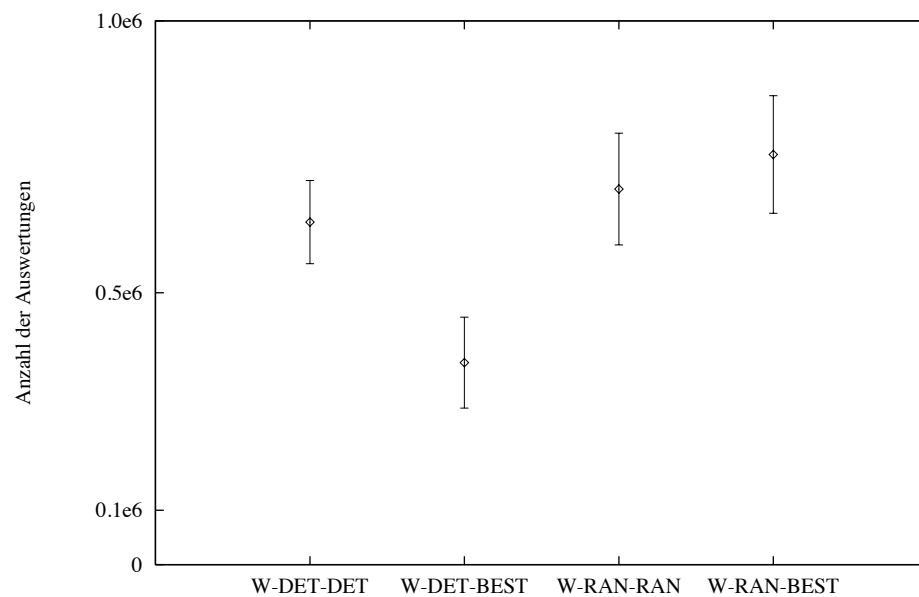


Bild 4.6: Vergleich der Anzahl der Funktionsauswertungen verschiedener Nachbarschaftsauswahlen (Intercity-Korpus, 80 Kategorien, zufällige Initialisierung (I-RAN)).

Im letzten Abschnitt 4.2 wurde die Initialisierung I-LW-RW (mit Nachbarschaftsauswahl W-RAN-RAN) als besonders gut ausgezeichnet. In diesem Abschnitt wurde die Nachbarschaftsauswahl W-DET-BEST (mit Initialisierung I-RAN) als besonders gut ausgezeichnet. Vergleicht man nun diese beiden Möglichkeiten (Bilder 4.4 und 4.5), so erkennt man, daß eine probabilistische Initialisierung einer probabilistischen Nachbarschaftsauswahl vorzuziehen ist.

4.4 Endkriterium

Dem Endkriterium kommt die Entscheidung zu, wann ein Optimierungslauf abgebrochen werden soll. Hier soll ein Optimierungslauf beendet werden, wenn die Abkühlung abgeschlossen ist und ‘längere Zeit’ keine Verbesserung der Kosten mehr stattgefunden hat. Das Ende der Abkühlung zu erwarten ist sinnvoll, da bei hoher Temperatur die Kostenfunktion sehr schwanken kann.

Es muß nun noch festgelegt werden, wie lange auf eine Verbesserung gewartet werden soll. Bei stochastischer Relaxation kann die Suche nach besseren Kosten abgebrochen werden, wenn die gesamte Nachbarschaft schon überprüft wurde und keine Verbesserung möglich war. In diesem Falle wurde ein lokales Optimum berechnet. Dies kann bei deterministischer Wort- und Kategorieauswahl (W-DET-DET) gewährleistet werden durch ein Endkriterium, das einen Optimierungsablauf beendet, wenn über $|\mathcal{C}| \cdot |\mathcal{W}|$ Iterationsschritte hinweg die Kosten nicht verbessert wurden (siehe Abschnitt 4.3). Bei der Nachbarschaft W-DET-BEST erhält man ein entsprechendes Endkriterium, wenn über $|\mathcal{C}|$ Iterationsschritte hinweg keine Verbesserung der Kosten aufgetreten ist. Diese Zeitspannen werden für die anderen iterativ-optimierenden Verfahren übernommen.

4.5 Zusammenfassung

In diesem Kapitel wurden die Optimierungskriterien so formuliert, daß die inkrementelle Auswertung möglichst effizient durchgeführt werden kann. Die vollständige Auswertung der Optimierungskriterien anhand der Kategorien-Häufigkeitsstatistik hat die Komplexität $\mathcal{O}(|\mathcal{C}|^2)$ und ist also quadratisch abhängig von der Zahl der Kategorien. Durch die inkrementelle Auswertung wird der konstante Aufwand $\mathcal{O}(|R(w)| + |L(w)|)$ benötigt, wobei $|R(w)| + |L(w)|$ vom verwendeten Korpus abhängt. Dies kann als nochmalige Reduktion des in [Kne93] beschriebenen Aufwandes von $\mathcal{O}(|\mathcal{C}|)$ angesehen werden.

Es wurden verschiedene Möglichkeiten für die Erzeugung eines initialen Zustandes verglichen. Die Perplexitäten der Initialisierungen unterscheiden sich sehr stark. Dies besitzt jedoch nur geringe Auswirkungen auf die schließlich erreichte Perplexität. Dabei schnitt die deterministische Initialisierung I-LW-RW am besten ab, bei der die häufigsten Wörter und deren benachbarte Wörter ($L(w)$ und $R(w)$) jeweils eine Kategorie für sich erhalten.

Es wurden verschiedene deterministische und probabilistische Möglichkeiten für die Implementierung der Nachbarschaftsauswahl verglichen. Dabei wurde festgestellt, daß (bei

stochastischer Relaxation) die zufällige Initialisierung I-RAN und die deterministische Nachbarschaftsauswahl W-DET-DET die besten Ergebnisse liefern. Die geringste Zahl von Funktionsauswertungen benötigt die optimale Kategorienauswahl W-DET-BEST. Im Kapitel 5 werden deshalb nur noch die Initialisierung I-RAN und die Nachbarschaftsauswahlen W-DET-BEST und W-DET-DET betrachtet.

Es wurde ein Endkriterium festgelegt, das einen Optimierungslauf beendet, wenn die ‘Temperatur’ auf einen bestimmten Wert abgekühlt ist und wenn während der Überprüfung der gesamten Nachbarschaft keine Kostenverringerung aufgetreten ist.

Kapitel 5

Vergleich der Optimierungsverfahren

In diesem Kapitel erfolgt ein Vergleich der in Kapitel 3 vorgestellten Optimierungsverfahren. Hierzu werden erst die freien Parameter der iterativ-optimierenden Verfahren eingestellt.

5.1 Parameter-Einstellung

Eine besondere Bedeutung bei Anwendung der iterativ-optimierenden Verfahren kommt der Bestimmung ihrer freien Parameter zu. Nur wenn für jedes Verfahren eine gute Parameter-Einstellung durchgeführt wurde, kann ein ‘gerechter’ Vergleich stattfinden. In Abschnitt 3.1 wurden die Algorithmen Sinflutalgorithmus, simuliertes Ausfrieren, Schwellwertakzeptanz und Record-To-Record Travel zusammen mit ihren Parametern erläutert. Nun soll gezeigt werden, wie diese Parameter bestimmt werden und welche Werte sich dabei ergeben. In den folgenden Diagrammen entspricht jeder Meßpunkt dem Mittelwert von 10 zufällig initialisierten Läufen mit einer gewählten Parameter-Einstellung.

Der Sinflutalgorithmus (Abschnitt 3.1.5) ist der einzige Algorithmus, der durch einen Parameter (α) vollständig bestimmt ist. Dieser Parameter bestimmt, wie schnell der ‘Wasserspiegel’ sich dem aktuellen Wert der Kostenfunktion annähert. In Bild 5.1 sind die erreichten Kosten für verschiedene α -Werte verglichen. Es ist ersichtlich, daß bei kleinerem α auch bessere Ergebnisse erzielt werden, allerdings nimmt auch die Zahl der Funktionsauswertungen zu (Bild 5.2).

Die anderen Verfahren besitzen einen Parameter der grob festlegt, nach welcher Anzahl von Iterationen im Vergleich zu stochastischer Relaxation die Abkühlung beendet sein soll. Dies sind ν_{SA} , ν_{TA} und ν_{RRT} . Ein Schätzwert für die notwendige Zahl der Funktionsauswertungen S_{HC} für einen Lauf mit stochastischer Relaxation wird anhand von Gleichung (4.9) bestimmt. Alle iterativ-optimierenden Verfahren, mit Ausnahme der stochastischen Relaxation, erlauben Verschlechterungen der Kostenfunktion. Dadurch ergibt sich, daß diese Verfahren mehr Iterationen als die stochastische Relaxation benötigen. Es soll bis auf weiteres $\nu_{SA} = \nu_{TA} = \nu_{RRT} = 2$ gesetzt werden, da dies einen Kompromiß zwischen geringer Rechenzeit und hoher Ergebnisqualität darstellt. Wie sich eine Veränderung dieser

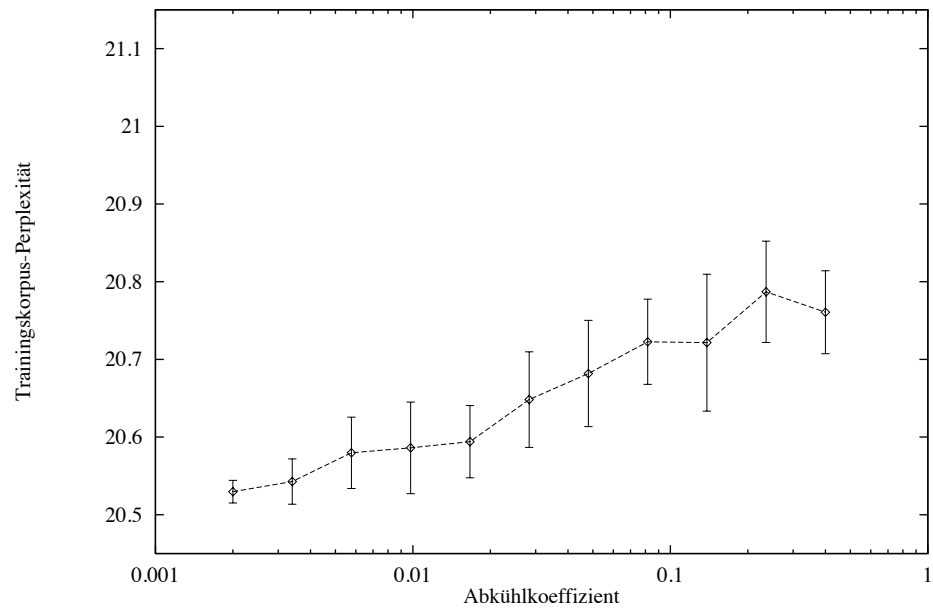


Bild 5.1: Trainingskorpus-Perplexität beim Sinflutalgorithmus für verschiedene Werte des Abkühlkoeffizienten α (Intercity-Korpus, 80 Kategorien, W-DET-DET).

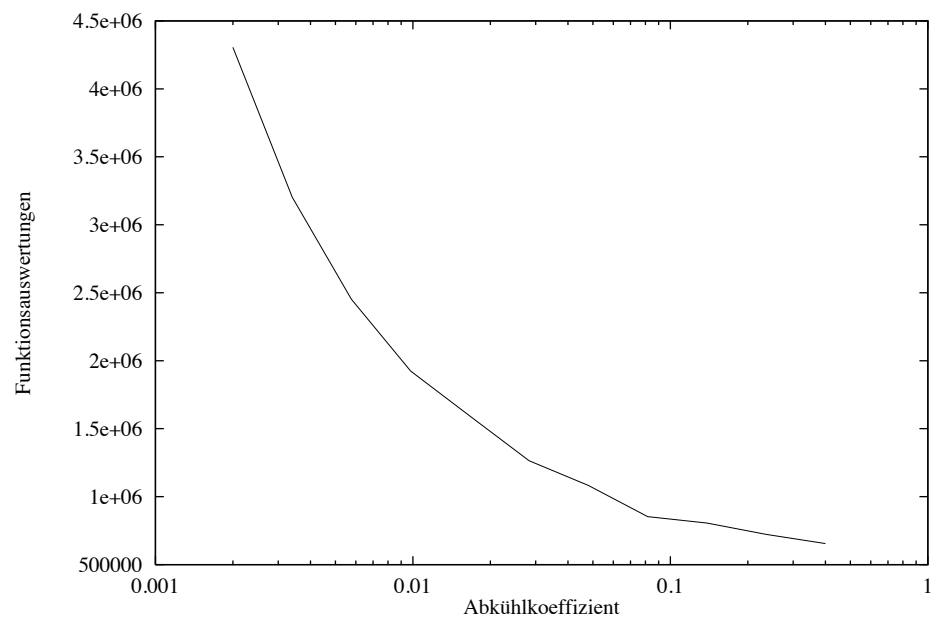


Bild 5.2: Mittlere Zahl der Funktionsauswertungen beim Sinflutalgorithmus für verschiedene Werte des Abkühlkoeffizienten α (Intercity-Korpus, 80 Kategorien, W-DET-DET).

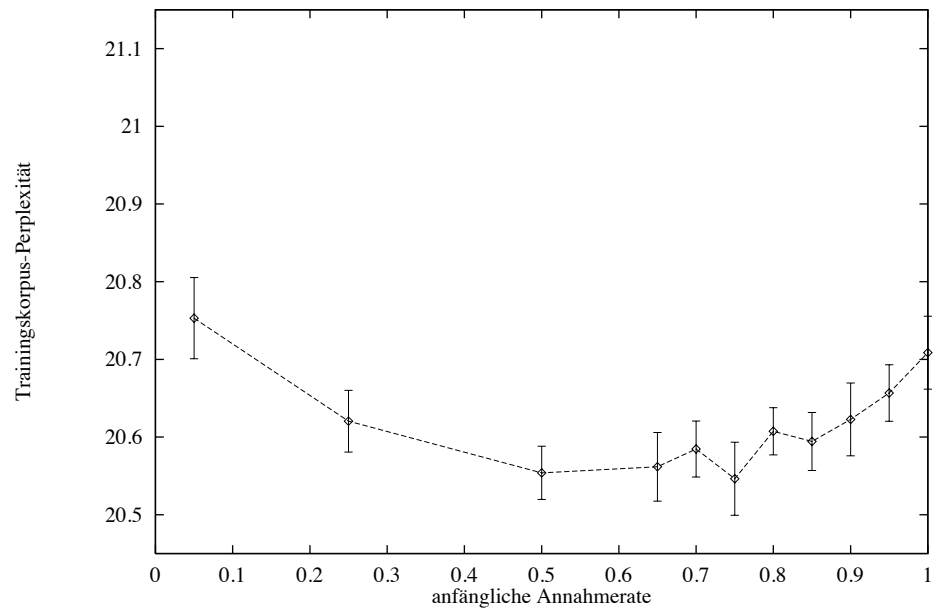


Bild 5.3: Trainingskorpus-Perplexität bei Record-To-Record Travel für verschiedene Werte der anfänglichen Annahmerate γ_{RRT} (Intercity-Korpus, 80 Kategorien, W-DET-DET).

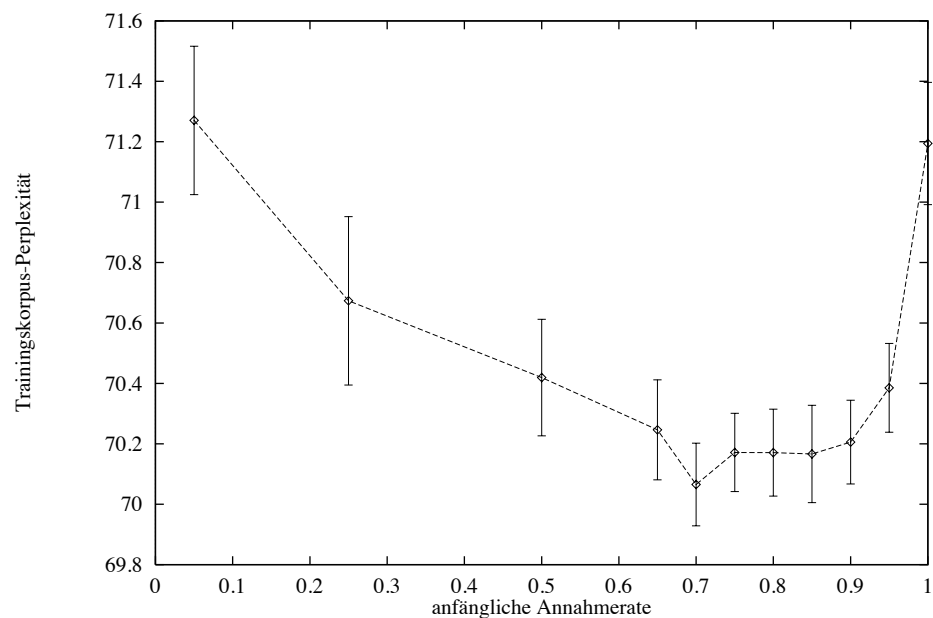


Bild 5.4: Trainingskorpus-Perplexität bei Record-To-Record Travel für verschiedene Werte der anfänglichen Annahmerate γ_{RRT} (Verbmobil-Korpus, 100 Kategorien, W-DET-DET).

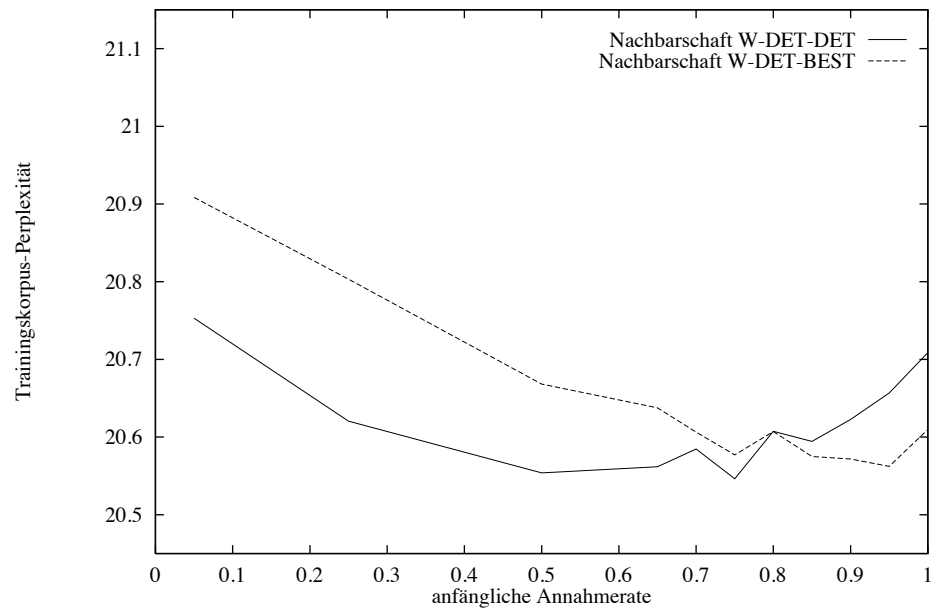


Bild 5.5: Trainingskorpus-Perplexität bei Record-To-Record Travel für verschiedene Werte der anfänglichen Annahmerate γ_{RRT} (Intercity-Korpus, 80 Kategorien).

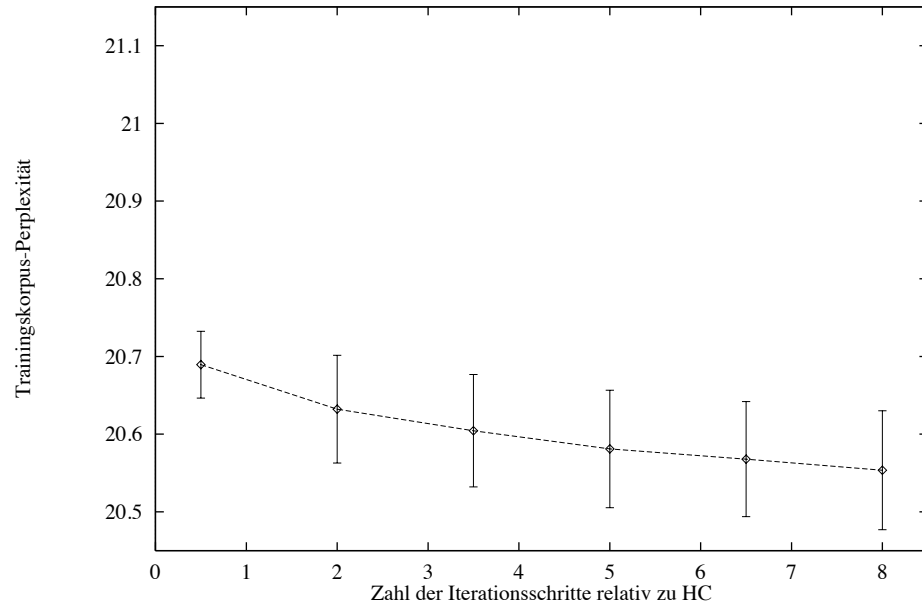


Bild 5.6: Trainingskorpus-Perplexität bei Record-To-Record Travel bei unterschiedlich langsamer Abkühlung (Intercity-Korpus, 80 Kategorien, W-DET-DET).

	γ_{TA}	γ_{RRT}	SA: γ_0	SA: γ_{end}	GDA: α	$\nu_{SA}, \nu_{TA}, \nu_{RRT}$
W-DET-DET	0.4	0.75	0.9	$1 \cdot 10^{-9}$	0.0125	2
W-DET-BEST	0.7	0.95	0.9	$1 \cdot 10^{-9}$	0.05	2

Tabelle 5.1: Parameter-Einstellungen für die iterativ-optimierenden Verfahren. Der Parameter α für den Sinflutalgorithmus (GDA) wurde so eingestellt, daß etwa dieselbe Zahl an Funktionsauswertungen wie bei den anderen Verfahren benötigt wurde.

Werte auswirkt, wird später beschrieben.

Bei Schwellwertakzeptanz und Record-To-Record Travel sind die anfänglichen Annahmeraten γ_{TA} bzw. γ_{RRT} zu bestimmen. In Bild 5.3 sind die erreichten Kosten für verschiedene Annahmeraten bei Record-To-Record Travel dargestellt. Die kleinsten Kosten werden für $\gamma_{RRT} = 0.75$ erzielt. Entsprechend wurde die Annahmerate γ_{TA} für Schwellwertakzeptanz auf 0.4 eingestellt (siehe Bild B.5). Man sieht, daß sowohl bei zu kleiner als auch bei zu großer Annahmerate schlechte Kosten erreicht werden. Insbesondere bei einer Annahmerate ≈ 0 verhält sich der Algorithmus wie stochastische Relaxation. In Bild 5.5 (für γ_{TA} siehe Bild B.6) ist dieser Vergleich zusätzlich mit der Nachbarschaftsauswahl W-DET-BEST ausgeführt. Man erkennt, daß bei dieser Nachbarschaftsauswahl eine andere anfängliche Annahmerate eingestellt werden sollte, um gute Ergebnisse zu erhalten.

Bei simuliertem Ausfrieren sind die zwei Parameter γ_0 und γ_{end} einzustellen. Man müßte hierzu ein zweidimensionales Optimierungsproblem lösen. Die anfängliche Annahmerate γ_0 sollte nach [Laa87, Fis94] so eingestellt werden, daß anfänglich fast alle Zustandsänderungen angenommen werden. Man kann also $\gamma_0 = 0.9$ setzen und dann die schließliche Annahmerate γ_{end} bestimmen (siehe hierzu Bild B.8).

Es wurde schon beim Sinflutalgorithmus festgestellt, daß bei kleinerem α , also bei langsamerer Abkühlung, die Qualität der Ergebnisse verbessert wird. Dies erwartet man auch für die anderen Optimierungsverfahren bei Vergrößerung von ν_{SA} , ν_{TA} und ν_{RRT} . In Bild 5.6 ist dieser Vergleich für Record-To-Record Travel durchgeführt (für andere Verfahren siehe Bilder B.11, B.12). Man sieht, daß bei einer langsameren Abkühlung geringfügig bessere Ergebnisse erzielt werden.

In Tabelle 5.1 sind die Einstellungen für alle Verfahren zusammengefaßt, welche anhand des relativ kleinen Intercity-Korpus bestimmt wurden. Es stellt sich jedoch die Frage, inwieweit die getroffenen Einstellungen auch für andere Korpora gute Ergebnisse erzielen. Der Einsatz größerer Korpora ist aufgrund der sich ergebenden Rechenzeiten sehr aufwendig. In Bild 5.4 sieht man ein Bild 5.3 entsprechendes Diagramm für das Verbmobil-Korpus (in Bild B.13 für das Limas-Korpus). Die Kurve hat einen anderen Verlauf. Das Minimum wird jedoch fast an derselben Stelle, bei $\gamma_{RRT} = 0.7$, angenommen. Man kann also davon ausgehen, daß die bestimmten Parameter auch auf anderen Korpora gute Ergebnisse liefern.

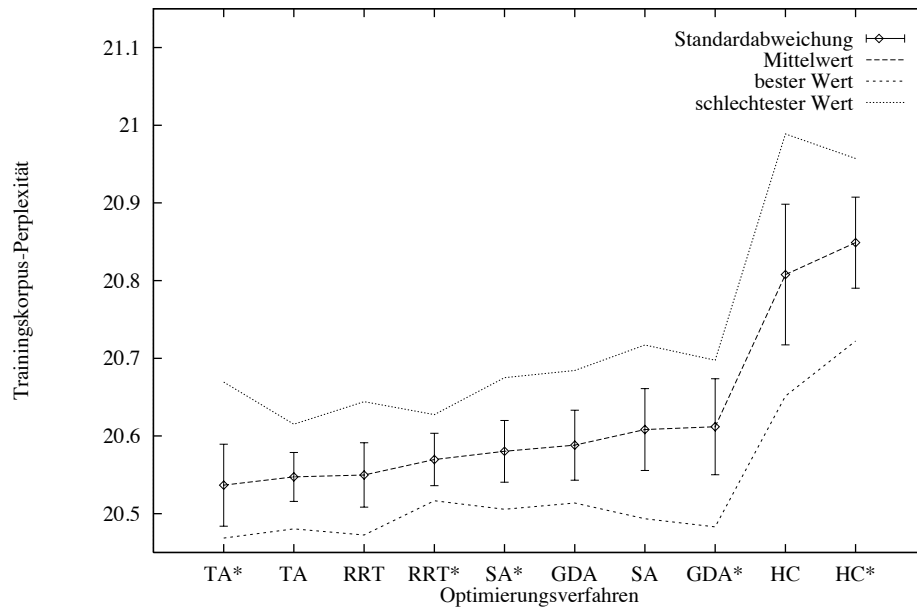


Bild 5.7: Vergleich verschiedener Optimierungsverfahren und der Nachbarschaften W-DET-DET und W-DET-BEST (gekennzeichnet mit *) (Intercity-Korpus, 80 Kategorien).

5.2 Vergleich der erreichten Kosten

Nun erfolgt der Vergleich der Optimierungsverfahren anhand der mit ihnen erreichbaren Werte der Kostenfunktion. In der Literatur findet man viele verschiedene Möglichkeiten diesen Vergleich durchzuführen ([Ack87, Nur93, Dor95]).

Soll beim Vergleich der Optimierungsverfahren nach einer bestimmten Zahl von Iterationsschritten oder einer bestimmten Zeit abgebrochen werden? In dieser Arbeit wird für einen einzelnen Optimierungslauf kein Zeitlimit vorgegeben. Eine Möglichkeit der Einflußnahme auf die Länge des Optimierungslaufes besteht bei simuliertem Ausfrieren, Schwellwertakzeptanz, Record-To-Record Travel und Simulated Annealing durch Veränderung des entsprechenden Parameters (ν_{SA} , ν_{TA} , ν_{RRT} , α).

Sollen die Mittelwerte verschiedener Läufe verglichen werden, oder die besten Werte (weil diese interessieren) oder die schlechtesten Werte (um auf der sicheren Seite zu sein)? Für den Anwender ist immer nur der beste Lauf interessant. Verwendet man diesen als Vergleichskriterium, so ist es möglich, daß Ausreißer das Ergebnis verfälschen. Im folgenden Vergleich werden verschiedene Vergleichsmöglichkeiten angewendet.

Zuerst erfolgt ein Vergleich der iterativ-optimierenden Verfahren untereinander. Darauf aufbauend werden dann die Ergebnisse der multidirektionalen Suche und der iterierten Zustandsraum-Reduktion betrachtet.

Iterativ-optimierende Verfahren: In Bild 5.7 sind die erreichten Perplexitäten für die iterativ-optimierenden Verfahren auf dem Intercity-Korpus für 20 Läufe dargestellt. In

Bild B.14 ist eine entsprechende Darstellung für das Verbmobil-Korpus. Jeder Optimierungslauf wurde zufällig initialisiert und mit den Nachbarschaftsauswahlen W-DET-DET und W-DET-BEST (gekennzeichnet durch einen Stern) getestet. Die Darstellung ist nach aufsteigenden Mittelwerten sortiert.

Vergleicht man nun die iterativ-optimierenden Verfahren (mit Nachbarschaft W-DET-BEST) anhand der erreichten Mittelwerte auf dem Intercity-Korpus, so fällt sofort auf, daß stochastische Relaxation die schlechtesten Ergebnisse liefert (Trainingskorpus-Perplexität: 20.85). Bei den iterativ-optimierenden Verfahren mit Abkühlung liefert Schwellwertakzeptanz (20.54) die besten und der Sinflutalgorithmus (20.61) die schlechtesten Ergebnisse. Im hier diskutierten Experiment liefern Record-To-Record Travel (20.57) und Schwellwertakzeptanz bessere Ergebnisse als das bekannte simulierte Ausfrieren. Anhand des größeren Verbmobil-Korpus (siehe Bild B.14) ergibt sich zwar eine etwas andere Rangfolge der Verfahren, jedoch liefert auch hier Schwellwertakzeptanz sehr gute und stochastische Relaxation die schlechtesten Ergebnisse.

In Abschnitt 4.3 wurde beschrieben, daß die Nachbarschaft W-DET-BEST starke Auswirkungen auf Verfahren haben kann, die auch Verschlechterungen akzeptieren. Anhand der Bilder 5.7 und B.14 ist es schwierig zu entscheiden, welche Nachbarschaft die besseren Ergebnisse erzielt, da dies offensichtlich vom verwendeten Verfahren abhängt. Wenn man jedoch berücksichtigt, daß die Nachbarschaftsauswahl W-DET-BEST eine geringere Zahl von Auswertungen benötigt, so kann man diese bevorzugen.

Bisher wurden die Mittelwerte aus 20 Optimierungsläufen verglichen. Erstellt man eine Rangfolge der Verfahren anhand der besten Werte (aus Bild 5.7), so liefert wieder Schwellwertakzeptanz mit Nachbarschaftsauswahl W-DET-BEST die besten Ergebnisse (20.47). Erstellt man eine Rangfolge anhand der schlechtesten Werte, so liefert Schwellwertakzeptanz mit Nachbarschaftsauswahl W-DET-DET die besten Ergebnisse (20.62).

Multidirektionale Suche mit Beschneidung: Es soll nun multidirektionale Suche mit Beschneidung (Abschnitt 3.2) mit in die Vergleiche einbezogen werden. Das Ziel dieses Verfahrens besteht in einer Einsparung an Rechenaufwand. Es werden die Beschneidungstabellen MSB-1, MSB-2 und MSB-3 aus Tabelle 5.2 verwendet. Die Einsparungen von 31%, 43% und 60.5% sind anhand der Gleichung (3.19) berechnet und gehen von 20 Läufen aus. In der Praxis fällt die Einsparung leicht geringer aus und liegt bei etwa 20%, 30% und 50%.

In Tabelle 5.3 werden die Ergebnisse von 20 Läufen mit den Ergebnissen der multidirektionalen Suche mit Beschneidung verglichen. Man erkennt, daß die Verfahren Record-To-Record Travel und Schwellwertakzeptanz auch bei großen Einsparungen noch die beste Lösung liefern. Dies ist ein positives Ergebnis, da diese Verfahren an sich sehr gute Ergebnisse erwarten lassen. Insbesondere Record-To-Record Travel liefert bei einer Einsparung von 60.5% (MSB-3) noch die beste Lösung. Die sehr schlechten Ergebnisse von simuliertem Ausfrieren lassen sich dadurch erklären, daß der Verlauf der Kostenfunktion lange Zeit sehr veränderlich ist und damit die Kosten zu einem frühen Zeitpunkt wenig Information über die schließlich erreichten Kosten liefern.

Iterierte Zustandsraum-Reduktion: Die iterierte Zustandsraum-Reduktion (Abschnitt 3.3) soll nun gegenüber dem mehrfachen Aufruf von iterativ-optimierenden Verfah-

MSB-1:	i	1	2	3	4
	r_i	0.5	0.6	0.7	0.8
	b_i	0.2	0.4	0.6	1.0

MSB-2:	i	1	2	3	4
	r_i	0.5	0.6	0.7	0.8
	b_i	0.3	0.5	0.7	1.0

MSB-3:	i	1	2	3	4
	r_i	0.2	0.3	0.4	0.7
	b_i	0.3	0.5	0.8	1.0

Tabelle 5.2: Beschneidungstabellen MSB-1 (Einsparung: 31%) und MSB-2 (Einsparung: 43%) und MSB-3 (Einsparung: 60.5%).

Verfahren	TA	RRT	SA	GDA	HC
Mittelwert	20.55	20.55	20.58	20.61	20.85
bester Wert	20.48	20.47	20.49	20.51	20.65
MSB-1	20.48	20.47	20.56	20.53	20.65
MSB-2	20.48	20.47	20.60	20.55	20.65
MSB-3	20.52	20.47	20.64	20.55	20.65

Tabelle 5.3: Ergebnisse von 20 Läufen mit den iterativ-optimierenden Verfahren mit multidirektionaler Suche mit Beschneidung (Intercity-Korpus, 80 Kategorien, W-DET-DET).

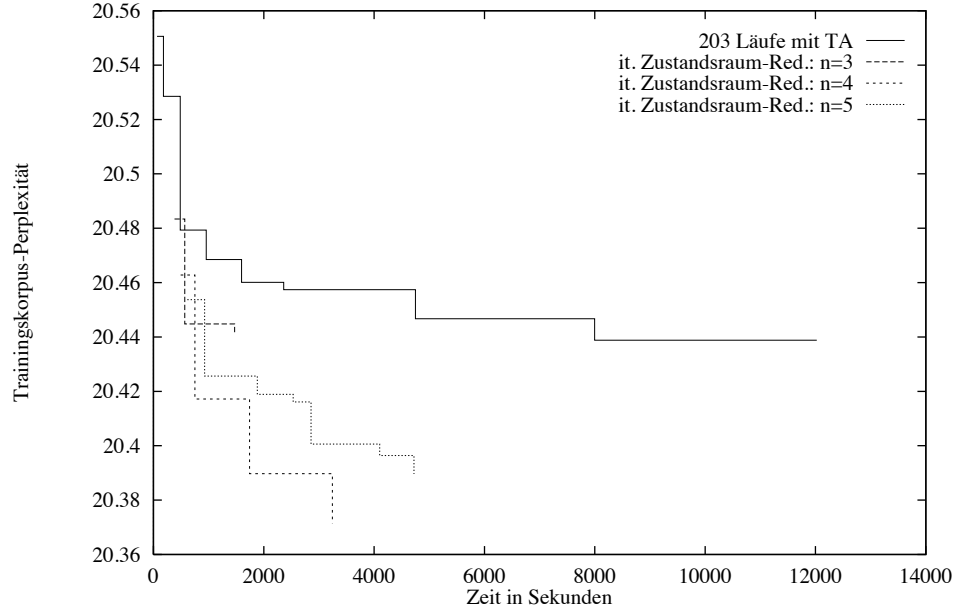


Bild 5.8: Ergebnisse einer großen Zahl von Läufen mit Schwellwertakzeptanz und mit iterierter Zustandsraum-Reduktion (Intercity-Korpus, 80 Kategorien, W-DET-BEST).

ren verglichen werden. Da Schwellwertakzeptanz als besonders gut festgestellt wurde, soll dieses Verfahren verwendet werden, um die benötigten guten Kategoriensysteme zu berechnen. In Bild 5.8 (Bild B.15) wurden die besten Ergebnisse von vielen Optimierungsläufen während 12 000 Sekunden CPU-Zeit von Schwellwertakzeptanz mit den Ergebnissen der iterierten Zustandsraum-Reduktion verglichen. Auf der waagrechten Achse ist die Rechenzeit in Sekunden aufgetragen. Der Parameter n gibt die Anzahl der Kategoriensysteme an, die an einer Zustandsraum-Reduktion beteiligt sind (siehe auch Abschnitt 3.3). Man erkennt, daß bei einer immer größeren Anzahl von iterativ-optimierenden Läufen die Ergebnisqualität nicht mehr stark verbessert werden kann (Perplexität: 20.44, 12 000 Sekunden). Mittels iterierter Zustandsraum-Reduktion ist es möglich, erheblich kleinere Kosten zu erreichen (beste Perplexität: 20.37, $n = 4$, 3244 Sekunden).

Wie man an den Ergebnissen am Verbmobil-Korpus (Bild B.15) noch deutlicher sehen kann, können also mittels der iterierten Zustandsraum-Reduktion erheblich bessere Ergebnisse erzielt werden, als es mit mehrfachem Aufruf von iterativ-optimierenden Verfahren möglich ist. Damit ist dieses Verfahren auch besser als multidirektionale Suche mit Beschneidung, denn hier werden keine besseren Ergebnisse erzielt, sondern höchstens gleichwertige Ergebnisse in kürzerer Zeit.

5.3 Zusammenfassung

Für die iterativ-optimierenden Verfahren mit Abkühlung mußten deren freie Parameter eingestellt werden. Dies geschah durch Vergleichen der Ergebnisse für verschiedene Parameter-Einstellungen. In Tabelle 5.1 sind alle vorgenommenen Einstellungen zusammengefaßt.

Es wurde dann ein Vergleich der Optimierungsverfahren anhand der erreichten Kosten durchgeführt. Hierbei wurden die Erwartungen in bezug auf stochastische Relaxation erfüllt und festgestellt, daß dieses Verfahren die schlechtesten Ergebnisse erzielt. Bei den anderen iterativ-optimierenden Verfahren liefert Schwellwertakzeptanz die besten Ergebnisse. Für die multidirektionale Suche mit Beschneidung wurde festgestellt, daß bei Schwellwertakzeptanz und Record-To-Record Travel fast ohne Qualitätsverlust bis zu 50% an Rechenaufwand eingespart werden kann. Mit iterierter Zustandsraum-Reduktion gelang es, die Ergebnisse der iterativ-optimierenden Verfahren deutlich zu verbessern.

Kapitel 6

Ergebnisse auf verschiedenen Korpora

In diesem Kapitel werden weitere Ergebnisse von Testläufen auf verschiedenen Korpora vorgestellt. Insbesondere wird die Perplexität auf einem vom Trainingskorpus unabhängigen Testkorpus betrachtet, wodurch unter anderem ein Vergleich mit einem manuell erstellten Kategoriensystem und ein Vergleich der Optimierungskriterien ML und LO durchgeführt werden kann. Es erfolgt dann noch eine Beschreibung der benötigten Rechenzeiten und der syntaktischen bzw. semantischen Eigenschaften der automatisch bestimmten Kategoriensysteme.

6.1 Testumgebung

Es wurden Tests mit verschiedenen Korpora durchgeführt, welche sich insbesondere in ihrer Größe unterscheiden. In Tabelle 6.1 sind einige Kenngrößen für die verwendeten Korpora dargestellt, die zusätzlich wie folgt charakterisiert werden können:

- Intercity-Korpus: spontansprachliche Benutzeräußerungen aus dem Diskursbereich Intercity-Auskunft
- Verbmobil-Korpus: spontansprachliche Benutzeräußerungen aus dem Diskursbereich

Korpus	Intercity	Verbmobil	Limas-04	Limas-16	Limas-80
Anzahl Sätze	2027	2600	1548	6122	41401
Anzahl Wörter	10894	59490	31652	124284	834389
Anzahl versch. Wörter	684	1975	10110	25838	97358
Anzahl einmal auftr. Wörter	234	65	7263	16690	57914
$ R(w) + L(w) $	10.89	16.07	5.37	7.18	10.14

Tabelle 6.1: Kenndaten der verwendeten Trainingskorpora.

Terminplanung

- **Limas-Korpus:** 500 deutschsprachige Texte verschiedenster Art. Das gesamte Korpus wurde in 3 Teile aufgeteilt. Die ersten 400 Texte wurden zum Korpus Limas-80 zusammengefaßt (beinhaltet 80% der Texte). Die Sätze der restlichen Texte wurden derart auf die Korpora Limas-16 und Limas-04 verteilt, daß sie 16% bzw. 4% des gesamten Korpus beinhalten.

Das Intercity- und das Verbmobil-Korpus sind aufgeteilt in einen Trainingsanteil — für den die Kategoriensysteme berechnet werden — und einen Testanteil. Dementsprechend wird zwischen Trainingskorpus-Perplexitäten und Testkorpus-Perplexitäten unterschieden. Die Testkorpus-Perplexitäten werden durch das Programm Polygramm ([Sch94a, Kuh94b]) berechnet.¹

Einzig für das Intercity-Korpus existiert ein manuell erstelltes Kategoriensystem, das in die Vergleiche einbezogen werden kann. Dieses Kategoriensystem ist nach syntaktischen, semantischen und morphologischen Gesichtspunkten entwickelt worden ([Wit92]). Das Limas-Korpus wurde insbesondere verwendet, um zu testen, ob auch sehr große Korpora verarbeitet werden können.

6.2 Auswertung

Bei den hier beschriebenen Auswertungen wurde das Verfahren Schwellwertakzeptanz mit der Nachbarschaftsauswahl W-DET-BEST verwendet, weil dieses Verfahren in Kapitel 5 als besonders gut eingestuft wurde. Jeder Meßpunkt in den Abbildungen entspricht dem besten Ergebnis (auf dem Trainingskorpus) aus mehreren Läufen.

Perplexität auf dem Trainingskorpus: Zuerst soll dargestellt werden, wie die Perplexität auf dem Trainingskorpus mit steigender Kategorienzahl verläuft. In Bild 6.1 ist die Trainingskorpus-Perplexität (ohne Interpolation für ungesehene Ereignisse) für verschiedene Kategorienzahlen zu sehen. Wie zu erwarten nimmt die Perplexität mit höherer Kategorienzahl stetig ab. Es ist auch das Ergebnis des manuell erstellten Kategoriensystems dargestellt. Man erkennt, daß die berechneten Kategoriensysteme eine erheblich kleinere Trainingskorpus-Perplexität aufweisen.

Perplexität auf dem Testkorpus und Bestimmung der Kategorienzahl: Eine unverfälschte Bewertung der Ergebnisse muß jedoch anhand eines unabhängigen Korpus erfolgen. Bei der Berechnung der Perplexität muß dann Wahrscheinlichkeitsmasse auf ungesehene Ereignisse verteilt werden. Da bei höherer Kategorienzahl auch mehr ungesehene Ereignisse auftreten können, ist es nun nicht mehr so, daß bei höherer Kategorienzahl zwangsläufig bessere Ergebnisse erzielt werden. In Bild 6.2 ist die Perplexität auf zwei verschiedenen Testkorpora für das Intercity-Korpus dargestellt. Das Minimum mit einer Perplexität von 37.87 der vereinigten Teststichproben befindet sich bei 80 Kategorien.

¹Es wurden folgende Einstellungen für Polygramm verwendet: context=2, mw=20, minocc=1, bayes=1, unseen=0, alpha=4711.

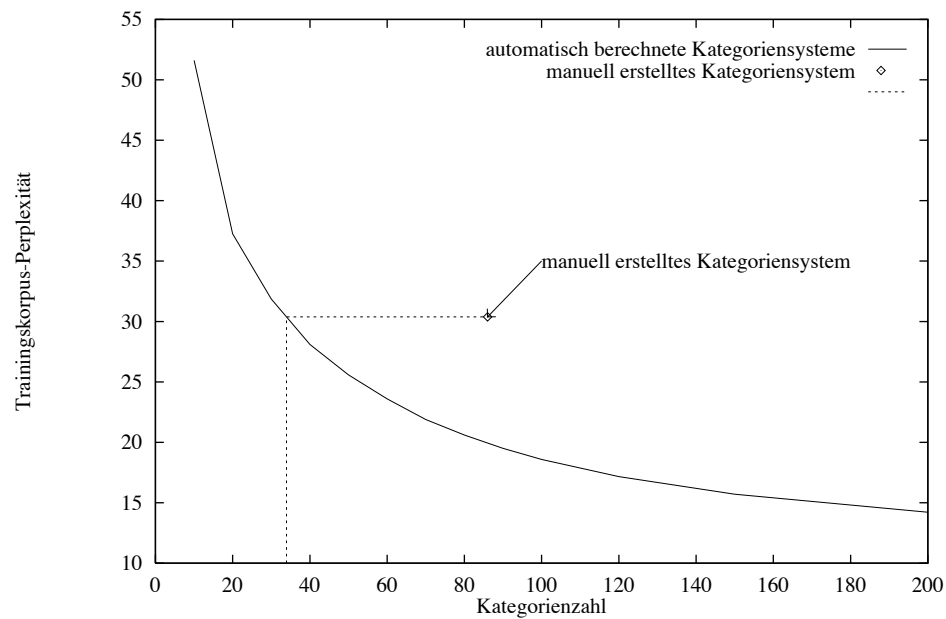


Bild 6.1: Trainingskorpus-Perplexität für verschiedene Kategorienzahlen (Intercity-Korpus).

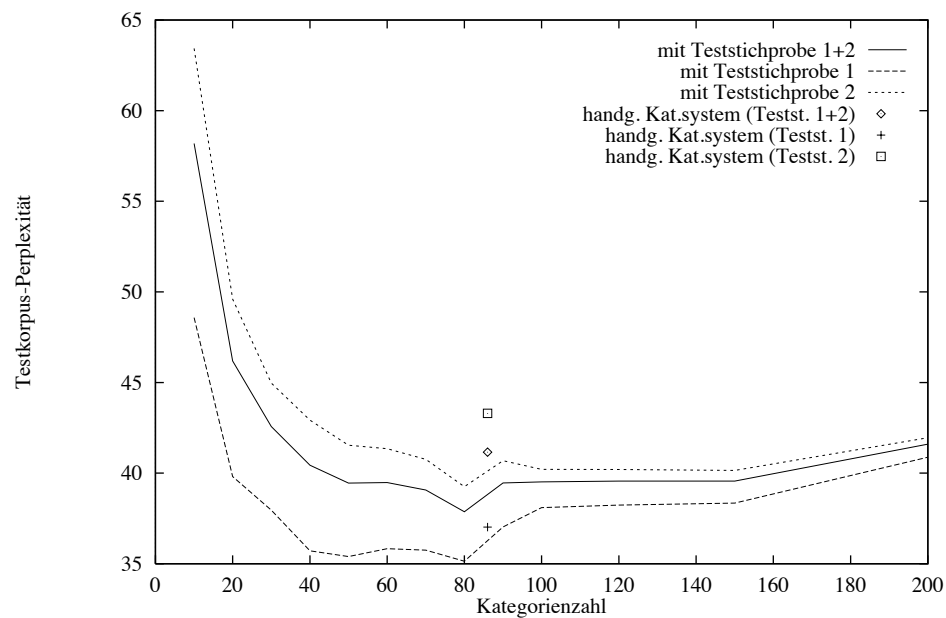


Bild 6.2: Testkorpus-Perplexität für verschiedene Kategorienzahlen (Intercity-Korpus).

Wie erfolgt die Erzeugung eines Kategoriensystems für eine bestimmte Domäne? Man benötigt zwei Korpora: ein (großes) Trainingskorpus und ein (kleineres) Testkorpus. Es werden für verschiedene Kategorienzahlen ‘gute’ Kategoriensysteme berechnet und anhand der Testkorpus-Perplexität verglichen. Die optimale Kategorienzahl kann durch Minimierung der Testkorpus-Perplexität bestimmt werden. Für diese Kategorienzahl kann dann durch intensiven Rechenzeiteinsatz ein ganz besonders gutes Kategoriensystem berechnet werden.

Es wurde nun versucht, ein hervorragendes Kategoriensystem für 80 Kategorien zu erhalten. Dabei wurden verschiedenste Testläufe durchgeführt. Die besten Ergebnisse erzielte die iterierte Zustandsraum-Reduktion. Das dabei ermittelte Kategoriensystem besitzt eine Trainingskorpus-Perplexität von 20.37 und ist im Anhang A abgedruckt.

Vergleich mit dem manuell erstellten Kategoriensystem: Der entscheidende Vergleich der automatisch berechneten Kategoriensysteme erfolgt jedoch mit dem manuell erstellten Kategoriensystem. Das Vokabular dieses Kategoriensystems mußte auf das Trainingskorpus-Vokabular eingeschränkt werden, da es viele weitere Wörter beinhaltet, welche die Perplexitäts-Berechnung beeinflussen würden.² Dieses modifizierte Kategoriensystem besitzt eine Perplexität von 41.16. Anhand des Bildes 6.2 kann man erkennen, daß die automatisch berechneten Kategoriensysteme (abhängig von der Kategorienzahl) bessere Testkorpus-Perplexität liefern. Bei 80 Kategorien wird mit einer Perplexität von 37.87 ein deutlich besseres Ergebnis erzielt.

Vergleich der Optimierungskriterien ML und LO: Um das Optimierungskriterium LO einsetzen zu können, muß noch ρ für die *absolute discounting*-Methode bestimmt werden (siehe Abschnitt 2.2.2). Es wurde in [Ney93] vorgeschlagen $\rho = 0.75$ zu setzen. Ein Vergleich der erreichten Testkorpus-Perplexität für verschiedene ρ -Werte (Bild B.16) ergibt ebenfalls dieses Ergebnis.

Anhand der Testkorpus-Perplexität lassen sich auch die Optimierungskriterien ML und LO vergleichen (Bilder B.20 und B.21). Dabei stellt man fest, daß sich ML und LO nicht prinzipiell in ihrer Qualität unterscheiden. Dies entspricht auch dem Ergebnis von [Kne93].

Es wäre denkbar, daß sich das Kriterium LO bei sehr kleinen Trainingsstichproben positiv auswirkt. In einem Test (Bild B.22) wurde das Intercity-Korpus bis auf 10% seiner ursprünglichen Größe verkleinert. Dabei wurde im Gegenteil sogar festgestellt, daß ML bessere Ergebnisse liefert.

Bei Testläufen in [Kne93] konnte mit LO die richtige Kategorienzahl bestimmt werden. Bei Testläufen mit dem Intercity-Korpus konnte man beobachten, daß bei sehr hoher vorgegebener Kategorienzahl nicht alle Kategorien belegt wurden. Die dadurch automatisch bestimmte Kategorienzahl liegt zwischen 200 und 220 und somit erheblich über der oben bestimmten Zahl von 80. Die Ursache könnte darin liegen, daß das Intercity-Korpus zu klein ist. Allerdings lieferte LO auch bei dem größeren Verbmobil-Korpus erheblich überhöhte Kategorienzahlen. Hier beträgt die optimale Kategorienzahl 100 (Bild B.17) und durch LO wird eine Zahl knapp über 1000 ermittelt.

²Es ergeben sich qualitativ die gleichen Ergebnisse, wenn die Vereinigung von Trainingskorpus-Vokabular und Testkorpus-Vokabular verwendet wird. Dies muß jedoch einheitlich in jedem Kategoriensystem geschehen, da die Berechnung der Testkorpus-Perplexität stark vom explizit im Kategoriensystem genannten Vokabular abhängt.

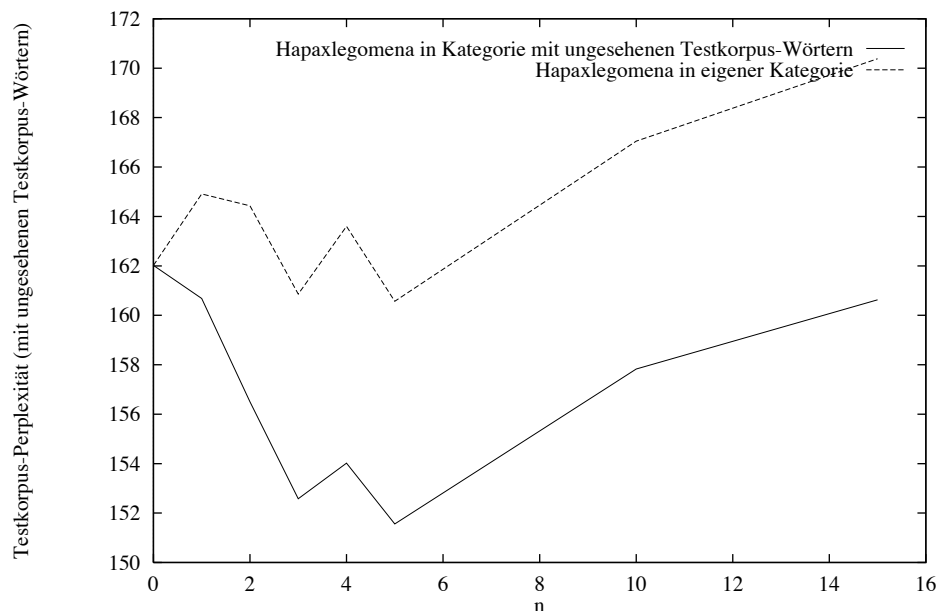


Bild 6.3: Testkorpus-Perplexitäten mit und ohne Berücksichtigung von Hapaxlegomena (Verbmobil-Korpus).

Berücksichtigung von Hapaxlegomena: Unter einem Hapaxlegomenon wird im allgemeinen ein Wort einer (heute nicht mehr gesprochenen) Sprache verstanden, das nur ein einziges Mal belegt ist. Im folgenden soll ein Wort, das im Trainingskorpus nur ein einziges Mal oder sehr selten auftritt als Hapaxlegomenon bezeichnet werden. Von diesen Wörtern kann man erwarten, daß eine Einordnung in die ‘richtige’ Kategorie schwierig ist, da nur wenig Information über dieses Wort vorhanden ist. Es ist nun jedoch denkbar, daß die Information über diese Hapaxlegomena genutzt werden kann, um die Auftretswahrscheinlichkeiten von im Testkorpus auftretenden, vorher ungesehenen Wörtern besser zu modellieren. Dies geschieht, indem man die Hapaxlegomena in einer dafür bereitgestellten Kategorie zusammenfaßt und für das ungesehene Testkorpus-Vokabular die Bigramm-Wahrscheinlichkeiten dieser Kategorie annimmt, also das ungesehene Testkorpus-Vokabular dieser Kategorie zuordnet. Die Hoffnung ist nun, daß dadurch die Testkorpus-Perplexitäten verringert werden.

Führt man einen Vergleich anhand des Intercity-Korpus durch, so stellt sich heraus, daß sich die Testkorpus-Perplexitäten nicht verringern (siehe Bild B.18). Im Gegensatz hierzu kann man beim Verbmobil-Korpus eine starke Verringerung feststellen. In Bild 6.3 ist die Testkorpus-Perplexität mit und ohne Berücksichtigung von Hapaxlegomena für das Verbmobil-Korpus aufgetragen. Der Parameter n gibt an, wie oft ein Wort höchstens auftreten darf, damit es als Hapaxlegomenon gilt. Der Parameter $n = 0$ bedeutet, daß Hapaxlegomena nicht verwendet werden. Es ist zu erkennen, daß ein Zusammenfassen der Hapaxlegomena mit dem ungesehenen Testkorpus-Vokabular (Testkorpus-Perplexität: 151.56

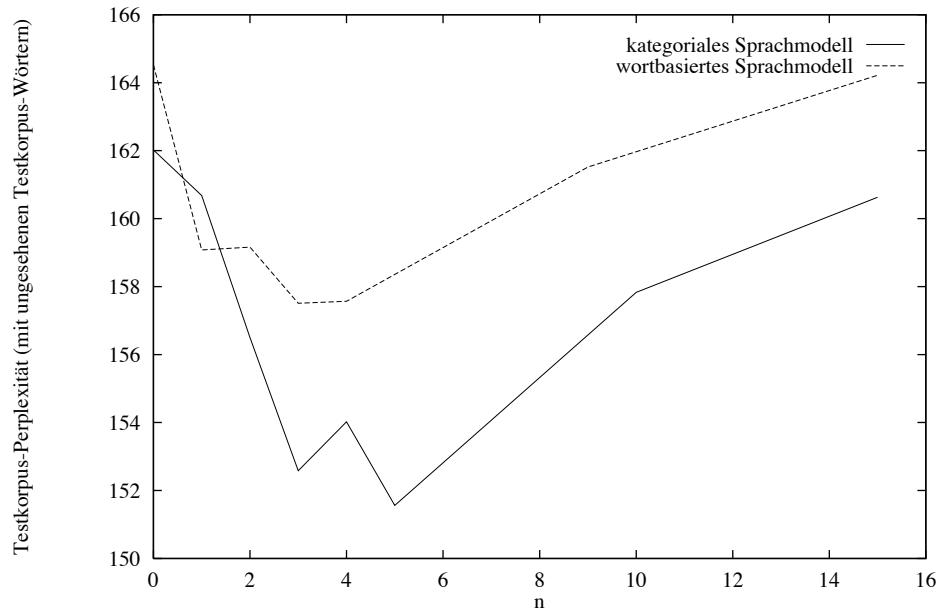


Bild 6.4: Testkorpus-Perplexität eines wortbasierten Modells gegenüber berechneten Kategoriensystemen für verschieden große Hapaxlegomena-Kategorien (Verbmobil-Korpus).

bei $n = 5$) eine deutliche Verringerung um mehr als 5% gegenüber einer abgeschlossenen Hapaxlegomena-Kategorie liefert (Testkorpus-Perplexität: 160.57 bei $n = 5$).

Vergleich mit wortbasiertem Modell: In Bild 6.4 (Bild B.19) wird die Testkorpus-Perplexität eines wortbasierten Modells mit berechneten Kategoriensystemen verglichen. Dabei wurde wieder eine Hapaxlegomena-Kategorie verwendet, um die Wahrscheinlichkeit für ungesehene Wörter zu schätzen. Man sieht, daß das kategoriebasierte Modell normalerweise bessere Ergebnisse liefert.

Rechenzeiten: In Tabelle 6.2 sind Rechenzeiten und Anzahl der Funktionsauswertungen für Schwellwertakzeptanz dargestellt. Anhand dieser Ergebnisse kann man erkennen, daß dieses Verfahren sehr effizient ist. Es gelingt innerhalb von 57 Sekunden ein Katego-

Korpus	Kategorien	Auswertungen [10^6]	Zeit [s]
Intercity	80	1.24	$5.7 \cdot 10$
Verbmobil	100	4.66	$2.6 \cdot 10^2$
Limas-04	100	34.8	$2.2 \cdot 10^3$
Limas-16	100	106.8	$8.0 \cdot 10^3$
Limas-80	100	324.3	$1.8 \cdot 10^4$

Tabelle 6.2: Rechenzeiten für einen Optimierungslauf mit Schwellwertakzeptanz auf einer HP9000/735 mit 124 MIPS für verschiedene Korpora.

riensystem mit 80 Kategorien für das Intercity-Korpus zu ermitteln, welches eine bessere Testkorpus-Perplexität als das manuell erstellte Kategoriensystem besitzt. Das Verfahren von [Kuh94b] benötigte auf demselben Korpus erheblich größere Rechenzeiten und es gelang nicht, Kategorien für das sehr große Limas-Korpus zu ermitteln ([Kyb93]), was mit den kombinatorischen Optimierungsverfahren durchaus möglich ist.

Nicht berücksichtigt in Tabelle 6.2 ist die Rechenzeit für die Auszählung der Bigramm-Statistiken, welche mit dem in [Hen95] beschriebenen Programm durchgeführt wird. Bei kleinen Korpora (Intercity, Verbmobil) ist der dadurch verbrauchte Rechenzeit-Anteil sehr gering (unter 10 Sekunden). Beim großen Limas-Korpus werden jedoch zwischen 500 (Limas-04) und 95000 Sekunden (Limas-80) benötigt. Die Rechenzeiten steigen also überproportional zur Größe des Vokabulars. Die Ursache hierfür dürfte zum Teil in der verwendeten Bibliothek NIHCL ([Gor90]) liegen, welche einen großen Speicherplatz- und Rechenzeitbedarf verursacht. Weiterhin ist das Programm aus [Hen95] für allgemeinere Problemstellungen entwickelt und ist nicht auf die Auszählung sehr großer Bigramm-Statistiken optimiert.

Zum Abschluß dieses Kapitels sollen die berechneten Kategoriensystemen noch näher betrachtet werden.

Eigenschaften der berechneten Kategoriensysteme: Es ist interessant sich ein berechnetes Kategoriensystem näher anzusehen. Die folgenden Kategorien stammen aus einem automatisch berechneten Kategoriensystem mit 80 Kategorien für das Intercity-Korpus, welches im Anhang A vollständig abgedruckt ist. Die mit ‘*’ gekennzeichneten Wörter treten nur einmal im Trainingskorpus auf.

1. wo wann ob mitnehmen*
2. Vormittag Nachmittag Mittag Abend März Zeiten*
3. einundzwanzigsten zehnten einunddreißigsten* vierten siebten elften Weihnachtsfeiertag zweiten fünften ersten vierundzwanzigsten sechsten* siebzehnten zweiundzwanzigsten* neunzehnten* dreiundzwanzigsten siebenundzwanzigsten* dritten zehnter*
4. Koblenz Hof* Dortmund Saarbrücken Osnabrück* Ulm Augsburg Frankfurt Paris Nürnberg Göttingen* Köln Bebra Weihnachten Heidelberg Würzburg Bonn
5. Ochtrup Mannheim Bamberg* Hamburg Athen Düsseldorf Graz* Berlin Abensberg* Solingen* Kiel* Oberstaufen* Utting* London Aachen Bremen Regensburg Wien Hause* Münster Stuttgart Rom Ansbach* Offenburg* Wuppertal* Hannover Karlsruhe Amsterdam*
6. Februar April Mai Juni Juli August September* Oktober Dezember* zweiundneunzig Vormittags einundneunzig Feiertag* neunzehnhunderteinundneunzig* Weihnachtstag
7. vierzehn wieviel neunzehn einundzwanzig fünfzehn zehn zwei vierundzwanzig dreizehn zweiundzwanzig zwanzig dreiundzwanzig achtzehn
8. Feiertagen Fahrrad* Gültigkeit* sechzehnten* Abfahrtszeit Woche S-Bahn

Es zeigt sich, daß viele Kategorien gebildet werden, die syntaktisch oder semantisch zusammengehörige Wörter enthalten. Beispielsweise sind die Wörter in Kategorie 1 Fragewörter, in Kategorie 2 sind Tageszeiten, in Kategorie 3 sind hauptsächlich deklinierte Ordinalzahlen und in den Kategorien 4 und 5 sind Städtenamen. Es gibt jedoch auch sehr gemischte Gruppen, wie Kategorie 6 wo Monatsnamen ('November' tritt im Intercity-Korpus nicht auf) zusammen mit einer ganzen Reihe anderer Wörter vorkommen und Kategorie 8, der wohl keine eindeutige semantische oder syntaktische Kategorie zuzuordnen ist.

Man sieht, daß teilweise die Kategorien der einmal auftretenden Wörter nicht sehr sinnvoll erscheinen. Beispielsweise befindet sich in den Kategorien 1 und 2 jeweils ein 'unerwünschtes' Wort ('mitnehmen' und 'Zeiten'). Andererseits ist bei den Städtenamen (Kategorien 4 und 5) festzustellen, daß auch einmal auftretende Städte in eine 'sinnvolle' Kategorie transportiert werden. Ähnlich gute Kategorisierungen seltener Wörter sind auch bei den Monatsnamen und anderen Kategorien festzustellen. Man kann dies dadurch erklären, daß Städtenamen im Diskursbereich 'Intercity-Auskunft' immer ähnlich verwendet werden und deshalb eine automatische Kategorisierung gute Ergebnisse liefert.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit beschäftigte sich mit der Entwicklung von Verfahren zur automatischen Bestimmung disjunkter Kategoriensysteme für statistische Sprachmodelle.

Es wurden unter Verwendung eines Maximum-Likelihood-Ansatzes und eines Kreuzvalidierungs-Ansatzes (leave-one-out) die zwei Optimierungskriterien ML und LO vorgestellt. Das Optimierungskriterium ML kennzeichnet die Güte eines Kategoriensystems für das Trainingskorpus und das Optimierungskriterium LO versucht auf unabhängige Korpora zu verallgemeinern.

Zur Auswertung dieser Kriterien wurde eine Reihe von iterativ-optimierenden Verfahren vorgestellt: stochastische Relaxation, simuliertes Ausfrieren, Schwellwertakzeptanz, Record-To-Record Travel und Sinflutalgorithmus. Die multidirektionale Suche mit Beschneidung ermöglicht es, durch parallele Verarbeitung mehrerer Optimierungsläufe effizienter zu guten Lösungen zu gelangen. Die iterierte Zustandsraum-Reduktion ist ein problemspezifisches Optimierungsverfahren, das auf den iterativ-optimierenden Verfahren aufbaut und darauf beruht, die gleichen Anteile in verschiedenen guten Kategoriensystemen zusammenzufassen, um einen kleineren Zustandsraum zu erhalten.

Die objektorientierte Implementierung erfolgte in C++. Es wurden zwei Klassenhierarchien für Optimierungsverfahren und Optimierungsprobleme verwendet. Durch die Definition von Methodenprotokollen können sehr allgemeine Typen von Optimierungsproblemen durch jedes allgemeine Optimierungsverfahren ausgewertet werden.

Für die Kriterien ML und LO wurde eine effiziente inkrementelle Auswertung vorgestellt und verschiedene deterministische und probabilistische Nachbarschaftsauswahlen und Initialisierungen verglichen.

Nach einer Einstellung der freien Parameter für die iterativ-optimierenden Verfahren wurden diese verglichen. Dabei erzielte stochastische Relaxation eindeutig die schlechtesten Ergebnisse (mittlere Trainingskorpus-Perplexität 20.85; Intercity-Korpus) und Schwellwertakzeptanz die besten Ergebnisse (20.54). Auch die anderen Verfahren Record-To-Record Travel (20.57), simuliertes Ausfrieren (20.58) und Sinflutalgorithmus (20.61) lie-

fernten bessere Ergebnisse als stochastische Relaxation. Durch multidirektionale Suche mit Beschneidung gelang es bei den Verfahren Record-To-Record Travel und Schwellwertakzeptanz bis zu 50% an Rechenzeit einzusparen, ohne die Qualität der berechneten Kategoriensysteme entscheidend zu verschlechtern. Die besten Ergebnisse wurden mit iterierter Zustandsraum-Reduktion erzielt. Es wurde auf dem Intercity-Korpus eine Perplexität von 20.37 im Gegensatz zu einer Perplexität von 20.44 bei mehrfachem Aufruf von Schwellwertakzeptanz erzielt.

Es ergab sich folgendes Verfahren zur Bestimmung eines Kategoriensystems für eine Domäne: Für das Trainingskorpus wird für verschiedene Kategoriengrößen ein Kategoriensystem berechnet. Dies kann mittels eines iterativ-optimierenden Verfahrens geschehen. Die berechneten Kategoriensysteme werden anhand der Testkorpus-Perplexität verglichen. Die optimale Kategoriengröße ergibt sich durch Minimierung der Testkorpus-Perplexität. Für diese Kategoriengröße kann dann durch ein aufwendigeres Optimierungsverfahren ein besonders gutes Kategoriensystem berechnet werden.

Die automatisch erzeugten Kategorien wurden mit (für das Intercity-Korpus) manuell erstellten Kategorien verglichen. Es wurde eine deutliche Verbesserung der Testkorpus-Perplexität von 41.16 auf 37.87 erzielt.

Bei experimentellen Untersuchungen an verschiedenen Korpora wurde festgestellt, daß die Testkorpus-Perplexitäten der beiden Optimierungskriterien ML und LO keine großen Unterschiede aufweisen. Es war nicht möglich mit LO die optimale Kategoriengröße ohne explizite Berechnung der Testkorpus-Perplexität einzustellen.

Durch Vereinigung der selten im Trainingskorpus auftretenden Wörter mit den ungesesehenen Wörtern aus dem Testkorpus können (abhängig vom Korpus) die Testkorpus-Perplexitäten deutlich reduziert werden. Beim Verbmobil-Korpus wurde eine Verbesserung um mehr als 5% erzielt.

Das hier vorgestellte Verfahren der automatischen Bestimmung von Kategoriensystemen erwies sich als sehr effizient. Für das Intercity-Korpus benötigt ein Lauf mit dem guten Optimierungsverfahren Schwellwertakzeptanz wenige Sekunden. Auch beim großen Limas-Korpus ist es möglich, in wenigen Stunden ein Kategoriensystem zu berechnen.

7.2 Ausblick

Es sind eine Reihe von Erweiterungs- und Einsatzmöglichkeiten des hier vorgenommenen Ansatzes denkbar.

Erweiterungsmöglichkeit: Die hier entwickelten Optimierungskriterien sind für kategoriale Bigramm-Sprachmodelle entwickelt. Als Erweiterung dieses Ansatzes wäre es denkbar, daß direkt Kategorien für n-Gramm-Sprachmodelle optimiert werden. Es würden dann aufwendigere Optimierungskriterien entstehen, in denen über n-dimensionale Statistiken summiert wird. Für diese müßte dann wieder eine effiziente Auswertung von Kostendifferenzen entwickelt werden. Es ist allerdings nicht sicher, ob sich dadurch bessere Kategoriensysteme bestimmen lassen.

Ein interessanter Ansatz ergibt sich, wenn das akustische Modell direkt in das Op-

timierungskriterium einbezogen wird. Es wurde schon in Abschnitt 2.2 festgestellt, daß der Zusammenhang zwischen Maximum-Likelihood auf dem linguistischen Sprachmodell und der Minimierung der Fehlerwahrscheinlichkeit nichttrivial ist. Es müßte also eine Kostenfunktion ermittelt werden, welche direkt die Fehlerwahrscheinlichkeit minimiert. Der entscheidende Punkt wird hierbei sein, daß es für diese Kostenfunktion eine effiziente inkrementelle Auswertung gibt, und die Auswirkung des Transportes eines Wortes von einer Kategorie in eine andere effizient ermittelt werden kann. Wenn dies ohne große Vereinfachungen möglich ist, dann kann man für die maschinelle Spracherkennung bessere Ergebnisse als mit dem hier vorgenommenen Maximum-Likelihood-Ansatz erwarten.

Einsatzmöglichkeiten: Die hier vorgestellten Optimierungsverfahren und ihre Implementierung sind durch die logische Trennung vom Optimierungsproblem allgemein verwendbar. Es sollte einfach möglich sein ein neues Problem damit zu behandeln, indem eine Klasse in die dafür vorgesehene Klassenhierarchie eingegliedert wird und die spezifizierten Funktionen bereitgestellt werden.

Die Implementierung des hier behandelten Optimierungsproblems ist direkt verwendbar, wenn Bigramm-Wahrscheinlichkeiten $P(a|b)$ für eine Menge von Elementen $X = \{a, b, \dots\}$ anhand von relativen Häufigkeiten $n(a, b)$ geschätzt werden sollen und es sinnvoll ist, eine Kategorisierung der Elemente in X vorzunehmen.

Eine interessante Anwendungsmöglichkeit dieses Ansatzes ergibt sich für die maschinelle Übersetzung. Es gibt Versuche, nur durch statistische Analyse zweisprachiger paralleler Korpora die Technik der Übersetzung automatisch zu lernen ([Bro90]). Dazu müssen die Übersetzungswahrscheinlichkeiten $P(w_1^{(B)} \dots w_{m_2}^{(B)} | w_1^{(A)} \dots w_{m_1}^{(A)})$ von Sätzen $w_1^{(A)} \dots w_{m_1}^{(A)}$ in Sprache A in Sätze $w_1^{(B)} \dots w_{m_2}^{(B)}$ der Sprache B bestimmt werden. In [Bro90] wird hierzu ein Modell entworfen, das diese Wahrscheinlichkeiten unter anderem auf die Übersetzungswahrscheinlichkeiten $P(w^{(A)} | w^{(B)})$ für einzelne Wörter zurückführt. Diese Wahrscheinlichkeiten werden durch relative Häufigkeiten geschätzt. Zur besseren Modellierung dieser Wahrscheinlichkeit könnte der hier verwendete Ansatz eingesetzt werden. Man kann sich vorstellen, daß dann Kategorien von Wörtern gebildet werden, die jeweils ineinander übersetzbar sind.

Eine andere Einsatzmöglichkeit dieses Verfahrens könnte bei kommerziellen Diktiersystemen liegen. Ein kommerzielles Diktiersystem sollte Möglichkeiten zur Erweiterung des Vokabulars durch den Benutzer besitzen. Für diese neuen Wörter muß das System nun die Auftretenswahrscheinlichkeiten schätzen. Dies ist jedoch schwierig, da das Trainingskorpus sehr klein ist — es besteht nur aus den vom Benutzer gesprochenen Sätzen. Man könnte sich nun vorstellen, daß eine automatische Kategorisierung der neuen Wörter in ein vorhandenes Kategoriensystem gute Ergebnisse liefert, da in Kapitel 6 festgestellt wurde, daß teilweise auch seltene Wörter gut in syntaktische oder semantische Kategorien transportiert werden.

Anhang A

Ergebnisse für das Intercity-Korpus

Im folgenden ist ein Kategoriensystem für das Intercity-Korpus mit 80 Kategorien abgedruckt. Es entstand mit dem Optimierungskriterium ML. Die Optimierung erfolgte durch iterierte Zustandsraum-Reduktion mit Schwellwertakzeptanz. Das Kategoriensystem besitzt eine Trainingskorpus-Perplexität von 20.37.

Durch diese Liste kann ein Überblick gewonnen werden, inwieweit syntaktische und semantische Kategorien gebildet werden. Dabei muß allerdings beachtet werden, daß solche Kategorien nicht das Ziel dieser Arbeit waren. Es ist jedoch durchaus interessant zu sehen, wieviel semantische Information in einer Bigramm-Statistik zu finden ist. Die genau einmal im Trainingskorpus auftretenden Wörter sind mit ‘*’ gekennzeichnet.

A.1 Ein Kategoriensystem

1. Gleis* Rollstuhlfahrer* Abständen*
Tag Zwischenaufenthalt
Ausschlußtag* Schnellzugbahnhof* IC
Zugs Zug Intercityzuschlag*
Superspartarif* Nachtzug Intercity
Hotel* Möglichkeit paar
2. zirka so nachlösen* bestehen*
ziemlich*
3. sechster* erster dreizehnter*
Wiedersehen Ostersonntag* danke
dritter tschüß herzlichsten_Dank*
auf_Wiedersehen* besten_Dank*
Wiederhören Mitte* vielen_Dank
auf_Wiederhören Januar
Danke_schön* elfter* zweiter* sofort
okay* jawohl* Fronleichnam* fünfter
4. Früh sobald* spät Nacht bald
Donnerstags* schnell*
5. wenig* mit dieses* jeden* ohne keinen
6. über Richtung
7. München Mainz Essen* Salzburg*
Trostberg*
8. umzusteigen* Jahres* Stop*
umsteigen durchfahren bin
Schlafwagen rechtzeitig*
9. Abfahrt wir stündliche* abfahrende
welches mehrere ankommende
heutigen* Abfahrtsort das was
Ankunftsort* welche wieviele*
10. diesem* dem nämlich welchem
durchgehenden früheren* Fall kurzem
11. neu* Mitternacht sechs null acht
Geschäftsschluß* Tutzing* eins elf
sechzehn drei zwölf neun fünf
siebzehn vier sieben
12. frühen zwölften schnellsten späteren
Bahnhof Wochenende
Aschermittwoch* späten besten
liebsten günstigsten
fünfundzwanzigsten* Ostersonntag*
Unterhaching*
13. ankommt los bestellen* ankomme
kommen rechnen* Grenze*
anschauen* verkehren* halten* sein
14. absolut* dort da angekommen
zweiunddreißig* richtig*
15. vierzehnter* zwanzigsten*
Taufkirchen* am
16. zurück lange fest* schlecht später
schneller Abreisedatum* lang* früher
teuer* oft unbedingt* wichtig*
möglich Ziel anders
17. ich wieso
18. nächster des übernächste dieser im
kommende* den
19. Zugauskunft* durchgehende*
Information günstige
Direktverbindung frühere
besonderen* Platz* ermäßigte*
direkte wegfahre* Fahrplanauskunft*
Wege* direkten nächstfrühere*
Rheinschleife* spätere Frage
Rundreise* spätestmögliche
20. noch viertel stündlich Intercityzüge*
21. einundzwanzigsten zehnten
einunddreißigsten* vierten siebten
elften Weihnachtsfeiertag zweiten
fünften ersten vierundzwanzigsten
sechsten* siebzehnten
zweiundzwanzigsten* neunzehnten*
dreiundzwanzigsten

- siebenundzwanzigsten* dritten
zehnter*
22. Zügen* Sonderzüge Wochentag
Tragflächenboot* Rhein* Eurozug*
losfahre Werktag Flughafen losführe*
gilt* aus Oma
23. Samstags denn immer* darauf man
Montags* dann
24. einen kein welchen ein überall*
welcher
25. erst gerne unter* für mich gern
26. direkter* von
27. als vor dreiviertel bis circa halb
28. Stunden Uhr Tagen Minuten Wochen
29. fährt Werktags* braucht* Sonntags
geht erreichen* Anfang* kommt
30. vierzehn wieviel neunzehn
einundzwanzig fünfzehn zehn zwei
vierundzwanzig dreizehn
zweiundzwanzig zwanzig
dreiundzwanzig achtzehn
31. werde fahre muß denke müßte
erreiche*
32. nächsterreichbaren* letzten*
übernächsten frühesten kommenden
entweder spätesten nächsten
nächstmöglichen diesen
33. mein wieder* zwar wie etwas falls*
Starnberg* wirklich* Nordenham*
34. um
35. fünfzehnte* späteste* nächste
nächstmögliche* üblichen genaue
letzte blaue* früheste erste allererste*
Mittagszeit*
36. zwischen gegen
37. wo wann ob mitnehmen*
38. ungefähr spätestens allerspätstens*
pünktlich normalerweise* frühestens
etwa
39. dahin* sondern Ankunft
Supersparpreis frühe* auch
zwischen durch Alternative Aufenthalt
sonst
40. höchstens* ausnutzen*
fünfundvierzig* mittags dreißig
nachts abgeholt* fünfunddreißig*
genug* Nachmittags Abends
anzukommen* Morgens
41. versteht* sind* möchten wollen* wäre
Alternativen* reicht ist Uhrzeit* gib*
42. möglichst gar überhaupt vielleicht
Freitags eigentlich vom wären*
frühmorgens
43. zu unterbrechen an parallel*
Emmerich
44. gibt wenn
45. gewußt sagen wissen fragen Auskunft
erkundigen* dabei* Informationen
Liegewagen*
46. Morgen ganz
47. direktem* jede keine folgende* eine
48. paßt sie hilft
49. lieber und
50. Feiertagen Fahrrad* Gültigkeit*
sechzehnten* Abfahrtszeit Woche
S-Bahn Orte* nehme damit* sechste*
Tage laufe Gepäckwagen*
Supersparpreises* Tages
51. kenne täglich* bräuchte dachte* suche
hätte benötigte brauche wollte
52. oder meiner zum einem ICs Ostern
abfahrenden* Grenzbahnhof* Stadt*

53. gleich Neujahr anschließend* egal
nirgends Heiligabend schon
prinzipiell* diese nichts mehr* gut
54. auf maximal* aber interessiert* doch
nur mal
55. Intercities* reservieren* Verbindung
Enzian* Vorlieben* Stunde Zeit
Zugfahrt* Konferenz* Fahrt Strecke
Fahrkarte Zugverbindung
IC-Verbindung* Rückfahrkarte*
56. nach einmal*
57. hallo* kurz ja guten_Abend
guten_Tag ach zuerst*
58. bei in durch*
59. können machen könnten hält zeigen*
haben nennen hat verkehrt geben
60. Freiburg* hier Hausham Rosenheim*
Anschluß selbst* Erding*
61. die Wegen* Vorschläge einige alle
62. genauen* Ankunftszeiten schnellere*
gerade* frühestmögliche* Ingolstadt
Ankunftszeit halben meine* übliche*
letztmögliche* erstmögliche* einer
schnellste Abfahrtszeiten kürzeste
schnellstmögliche* billigste*
63. Ochtrup Mannheim Bamberg*
Hamburg Athen Düsseldorf Graz*
Berlin Abensberg* Solingen* Kiel*
Oberstaufen* Utting* London Aachen
Bremen Regensburg Wien Hause*
Münster Stuttgart Rom Ansbach*
Offenburg* Wuppertal* Hannover
Karlsruhe Amsterdam*
64. schaffe guten_Morgen darf* besteht*
Grüß_Gott nein naja*
65. sich* mir
66. aller der
67. Hauptsache* komme* daß weit*
käme kann komme könnte weiß habe
68. Montag Dienstag Karfreitag
dreizehnten* Samstag Monat
Faschingsdienstag vierzehnten*
Sonntag Freitag Mittwoch Donnerstag
Ostermontag
69. will würde sollte* glaube möchte
70. August Februar zweiundneunzig Mai
Oktober Juli Dezember* Vormittags
einundneunzig Feiertag* September*
neunzehnhunderteinundneunzig*
April Weihnachtstag Juni
71. sicher ab
72. Vormittag Nachmittag März Abend
Zeiten* Mittag
73. Koblenz Hof* Dortmund Saarbrücken
Osnabrück* Ulm Augsburg Frankfurt
Paris Nürnberg Göttingen* Köln
Bebra Weihnachten Heidelberg
Würzburg Bonn
74. er* es
75. Pasing möglicherweise* wohl* genau
bitte gehen
76. abfährt nehmen Lüdenscheid hin
ausgeben* schlafen* Westbahnhof*
fahren
77. spätabends irgendwann Dienstags
steht* jetzt übermorgen heute
78. preisgünstig* dorthin nicht
Ostbahnhof vorher* direkt eher
79. wegfahren Fahrzeit Hauptbahnhof
abfahren herum ankommen Ost*
aussteigen* losfahren
80. heißt* genügt Intercitys Züge
Möglichkeiten Feiertage* liegt*
Zugverbindungen Pfingsten*
Verbindungen Mainschleife* Bahn
hängt morgigen* Nachtzüge

A.2 Ergebnisse einer Zustandsraum-Reduktion

Im folgenden sind ‘stark zusammengehörige Anteile’ (siehe Abschnitt 3.3) eines Kategoriensystems mit 80 Kategorien für das Intercity-Korpus zu sehen. Die Liste entstand während eines Laufes der iterierten Zustandsraum-Reduktion und gibt die Wörter an, die in der ersten Zustandsraum-Reduktion zu Wortclustern zusammengefaßt wurden. Es wurde dabei das Infimum von 10 verschiedenen mit Schwellwertakzeptanz bestimmten lokalen Minima gebildet. Man kann erkennen, daß diese ‘stark zusammengehörigen Anteile’ noch ‘bessere’ syntaktische und semantische Gruppen von Wörtern bilden, als die eigentlichen Kategorien.

- | | |
|---|--|
| 1. gar überhaupt eigentlich | April |
| 2. spätabends Dienstags jetzt
übermorgen heute | neunzehnhunderteinundneunzig*
Weihnachtstag Juni |
| 3. übernächsten kommenden nächsten
diesen | 13. schnellsten Wochenende
Aschermittwoch* |
| 4. letztmögliche* erstmögliche* übliche*
schnellste kürzeste schnellstmögliche* | 14. Auskunft Informationen |
| 5. späteste* nächste nächstmögliche*
üblichen letzte früheste allererste*
erste | 15. Ausschlußtag* Intercityzuschlag* Zug
Superspartarif* Nachtzug Intercity |
| 6. Ochtrup Bamberg* Athen Graz*
Berlin Kiel* Solingen* Aachen
Bremen Regensburg Hause* Münster
Wien Rom Ansbach* Offenburg*
Wuppertal* Karlsruhe Amsterdam* | 16. Bahnhof besten günstigsten
Ostersonntag* liebsten |
| 7. Vormittag Nachmittag Abend Mittag | 17. Koblenz Hof* Bebra |
| 8. höchstens* fünfundvierzig* dreißig
mittags nachts abgeholt*
fünfunddreißig* Nachmittags Abends
anzukommen* Morgens | 18. Dortmund Köln Würzburg Bonn |
| 9. Abfahrt welches Abfahrtsort das
Ankunftsort* was | 19. Hamburg Düsseldorf Hannover |
| 10. Ankunftszeiten Abfahrtszeiten | 20. besten_Dank* Danke_schön* |
| 11. Abreisedatum* Ziel | 21. Dienstag Montag Monat Sonntag
Freitag |
| 12. August Februar Mai Juli Dezember*
Feiertag* einundneunzig September* | 22. Zugauskunft* Direktverbindung
Fahrplanauskunft* |
| | 23. Freiburg* hier Rosenheim* selbst*
Erding* |
| | 24. Essen* Salzburg* |
| | 25. Wochentag Eurozug* |
| | 26. Konferenz* Zugfahrt* Fahrkarte |
| | 27. Verbindung Fahrt IC-Verbindung*
Zugverbindung |

- | | |
|--|--|
| 28. wegfahren Fahrzeit abfahren herum
ankommen losfahren | 52. aber doch |
| 29. Ostermontag Faschingsdienstag | 53. abfahrende ankommende |
| 30. Züge Feiertage* Zugverbindungen
Pfingsten* Verbindungen morgigen*
Nachtzüge | 54. absolut* dort da richtig* |
| 31. Rhein* Tragflächenboot* Flughafen | 55. ja guten_Abend ach zuerst* |
| 32. Frankfurt Paris | 56. vierzehn wieviel einundzwanzig
fünfzehn zehn dreizehn
vierundzwanzig zwei zweiundzwanzig
zwanzig dreiundzwanzig achtzehn |
| 33. Wiedersehen danke Ostersonntag*
tschüß herzlichsten_Dank*
Wiederhören vielen_Dank
auf_Wiederhören sofort jawohl*
Fronleichnam* okay* | 57. einige alle |
| 34. Göttingen* Heidelberg | 58. aller der |
| 35. Mitternacht sechs acht
Geschäftsschluß* Tutzing* eins elf
sechzehn zwölf drei neun fünf vier
sieben | 59. allerspätestens* frühestens |
| 36. schaffe darf* Grüß_Gott | 60. zwanzigsten* am |
| 37. Hauptbahnhof Ost* aussteigen* | 61. ankommt bestellen* los ankomme
kommen rechnen* anschauen*
verkehren* sein |
| 38. Information Rundreise* | 62. bald schnell* |
| 39. noch stündlich Intercityzüge* | 63. kenne bräuchte hätte benötige
brauche wollte |
| 40. zweiter* Januar | 64. dreiviertel circa halb |
| 41. Samstag Mittwoch | 65. daß weit* |
| 42. Samstags denn Montags* | 66. diesem* dem welchem kurzem |
| 43. zwar Nordenham* | 67. nächster dieser |
| 44. Oberstaufen* Utting* | 68. durchgehende* günstige frühere
direkte spätere |
| 45. Woche Orte* S-Bahn | 69. einundzwanzigsten zehnten
einunddreißigsten* siebten elften
zweiten vierundzwanzigsten fünften
ersten siebzehnten
zweiundzwanzigsten* neunzehnten*
dreiundzwanzigsten dritten |
| 46. Osnabrück* Ulm | 70. durchgehenden früheren* |
| 47. nächstfrühere* Rheinschleife* | 71. einen welchen ein welcher |
| 48. fährt Werktags* Sonntags kommt | 72. nach einmal* |
| 49. Stunden Uhr Wochen | 73. gewußt wissen erkundigen* |
| 50. vierten Weihnachtsfeiertag | 74. ungefähr spätestens pünktlich etwa |
| 51. abfährt hin ausgeben* schlafen*
Westbahnhof* fahren | |

- 75. zwölften fünfundzwanzigsten*
- 76. wie falls*
- 77. fest* schlecht später früher
unbedingt* wichtig*
- 78. frühen späten
- 79. letzten* nächsterreichbaren*
frühesten spätesten nächstmöglichen
- 80. möglicherweise* wohl* genau
- 81. gerne unter* gern mich
- 82. sind* wäre reicht gib* ist
- 83. machen haben
- 84. jeden* keinen
- 85. komme* käme kann komme
- 86. können könnten
- 87. sollte* will möchte
- 88. möchten wollen*
- 89. teuer* möglich
- 90. muß müßte
- 91. maximal* nur
- 92. stündliche* mehrere
- 93. nein naja*
- 94. neu* siebzehn
- 95. paßt sie
- 96. sechsten* siebenundzwanzigsten*
- 97. welche wieviele*

Anhang B

Diagramme

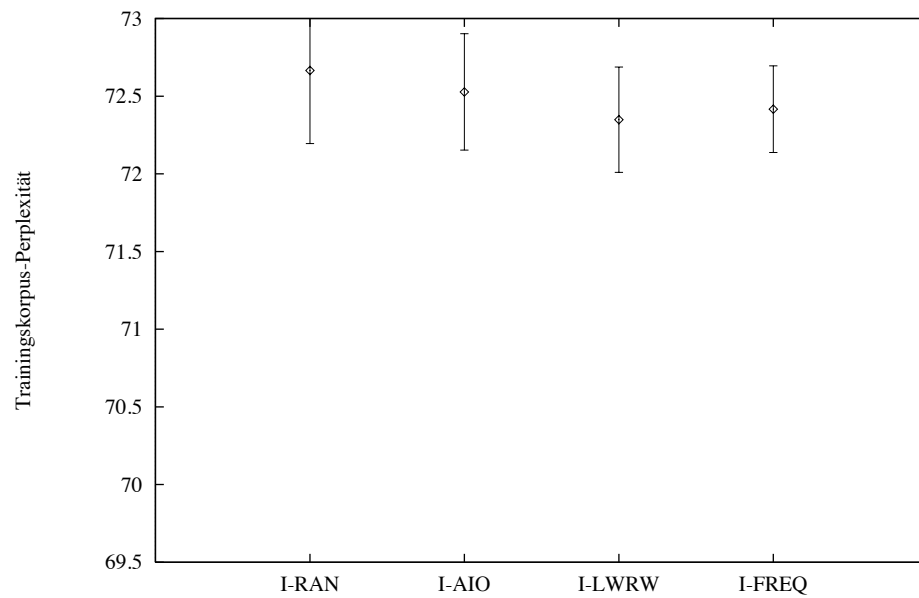


Bild B.1: End-Perplexität verschiedener Initialisierungen (Verbmobil-Korpus, 100 Kat., W-RAN-RAN).

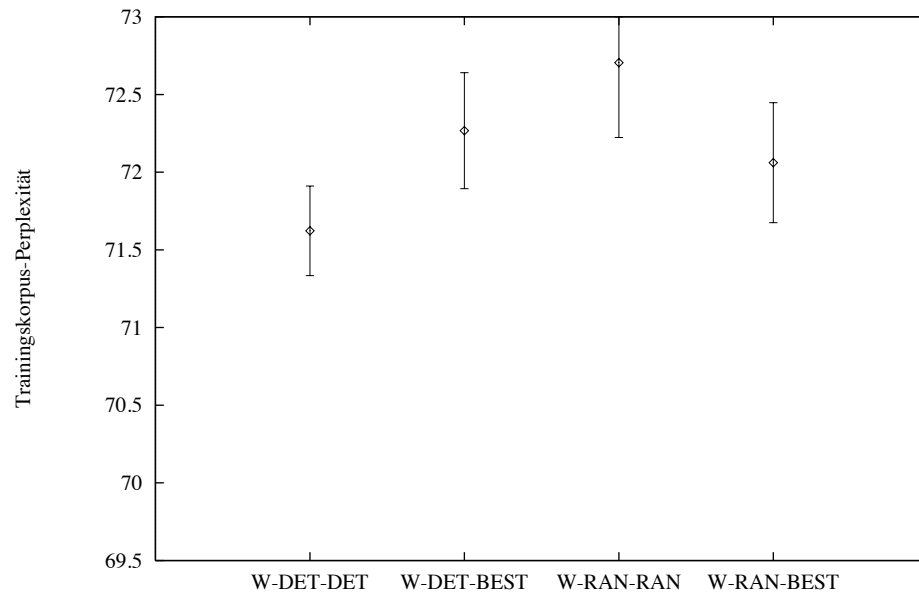


Bild B.2: End-Perplexität verschiedener Nachbarschaftsauswahlen (Verbmobil-Korpus, 100 Kat., I-RAN).

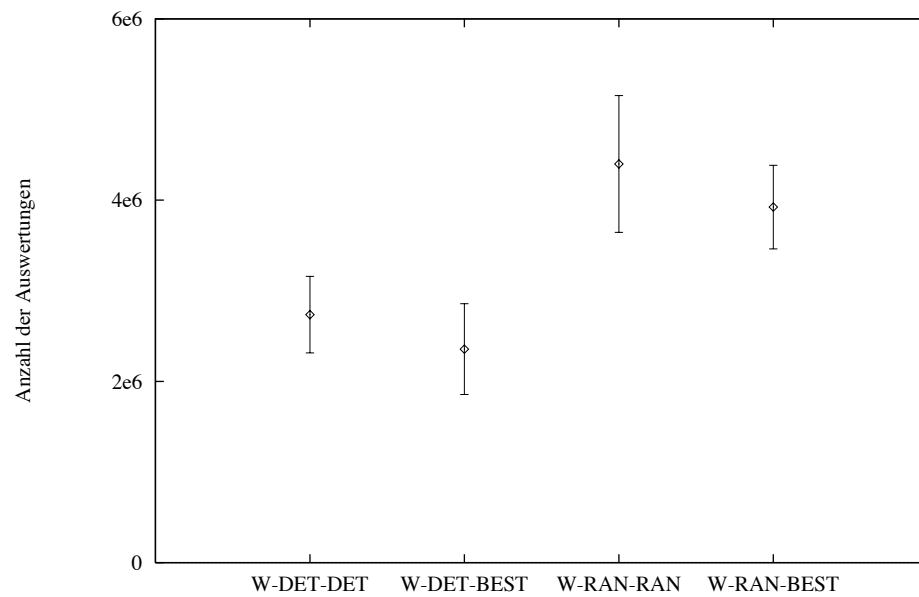


Bild B.3: Anzahl der Funktionsauswertungen verschiedener Nachbarschaftsauswahlen (Verbmobil-Korpus, 100 Kat., I-RAN).

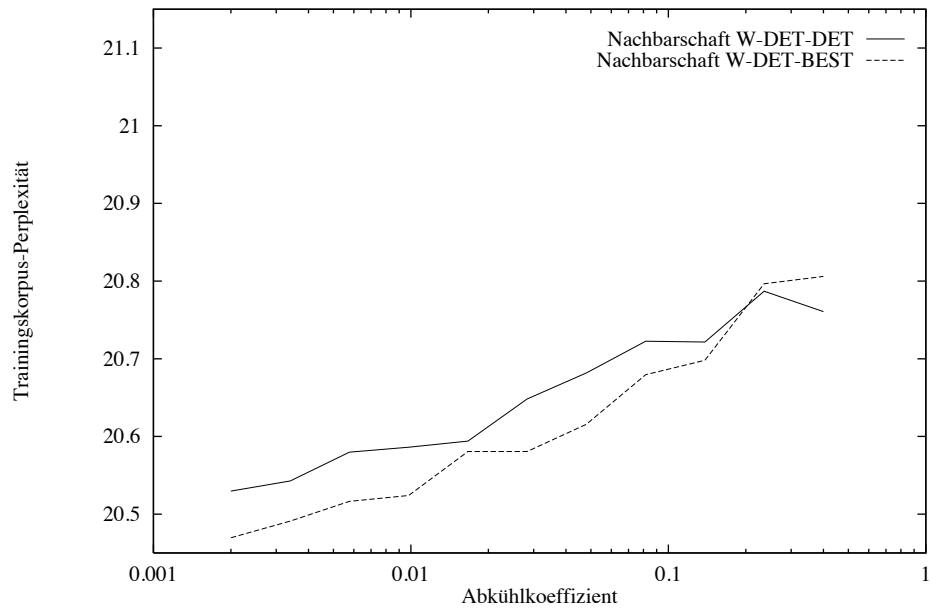


Bild B.4: Trainingskorpus-Perplexität beim Sinflutalgorithmus für verschiedene Werte des Abkühlkoeffizienten α und die Nachbarschaftsauswahlen W-DET-DET und W-DET-BEST (Intercity-Korpus, 80 Kat.).

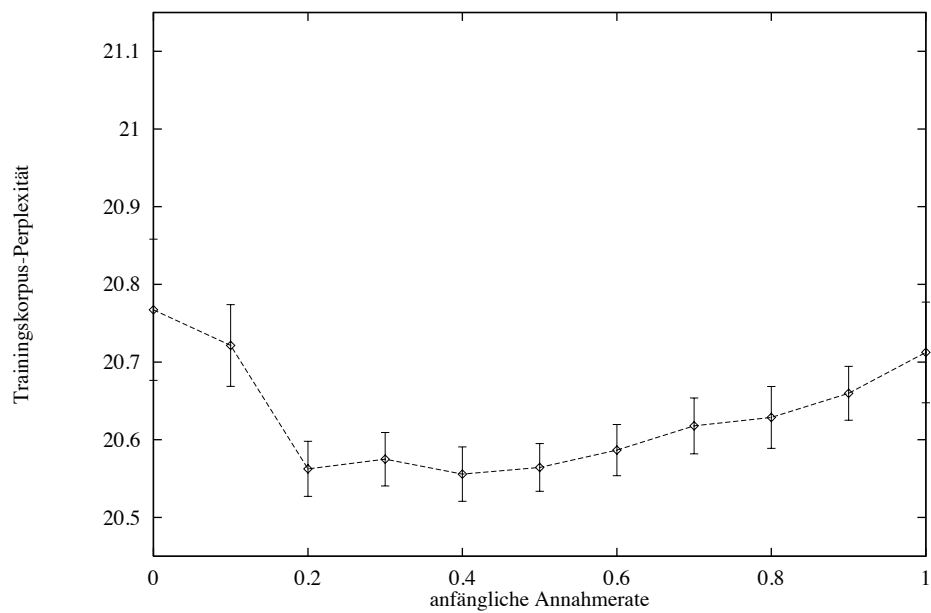


Bild B.5: Trainingskorpus-Perplexität bei Schwellwertakzeptanz für verschiedene Werte der anfänglichen Annahmerate γ_{TA} (Intercity-Korpus, 80 Kat., W-DET-DET).

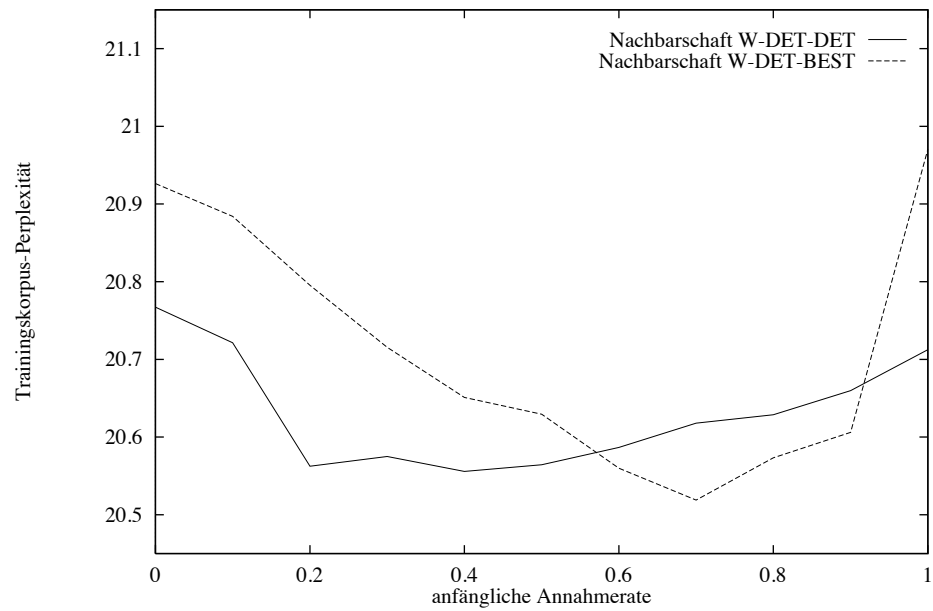


Bild B.6: Trainingskorpus-Perplexität bei Schwellwertakzeptanz für verschiedene Werte der anfänglichen Annahmerate γ_{TA} (Intercity-Korpus, 80 Kat.).

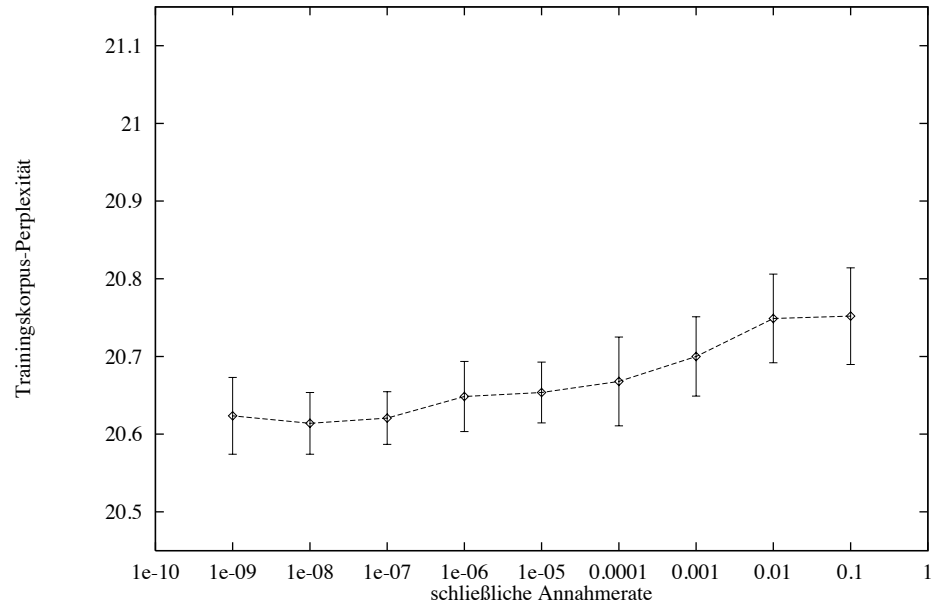


Bild B.7: Trainingskorpus-Perplexität bei simuliertem Ausfrieren für verschiedene Werte der schließlichen Annahmerate γ_{end} (Intercity-Korpus, 80 Kat., W-DET-DET).

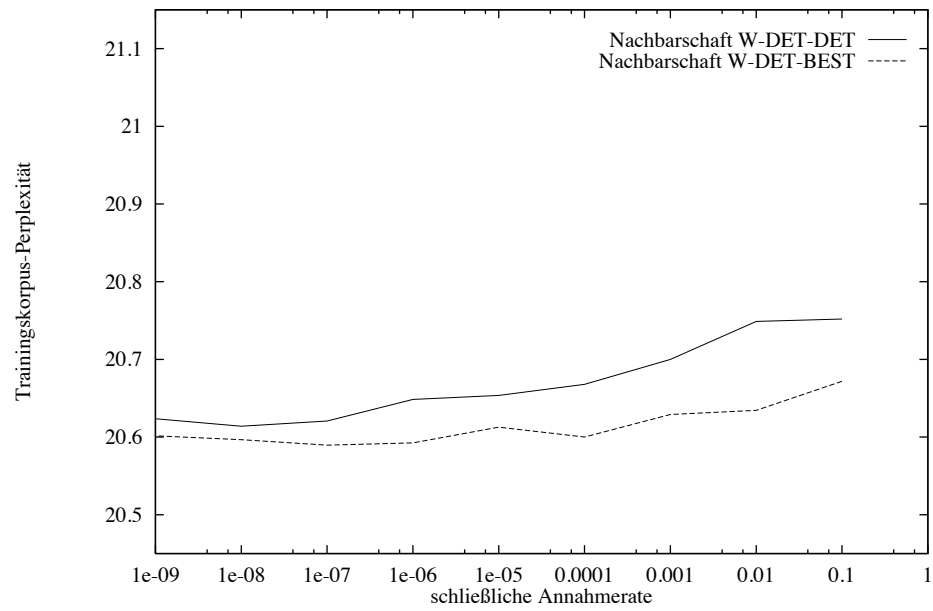


Bild B.8: Trainingskorpus-Perplexität bei simuliertem Ausfrieren für verschiedene Werte der schließlichen Annahmerate γ_{end} (Intercity-Korpus, 80 Kat.).

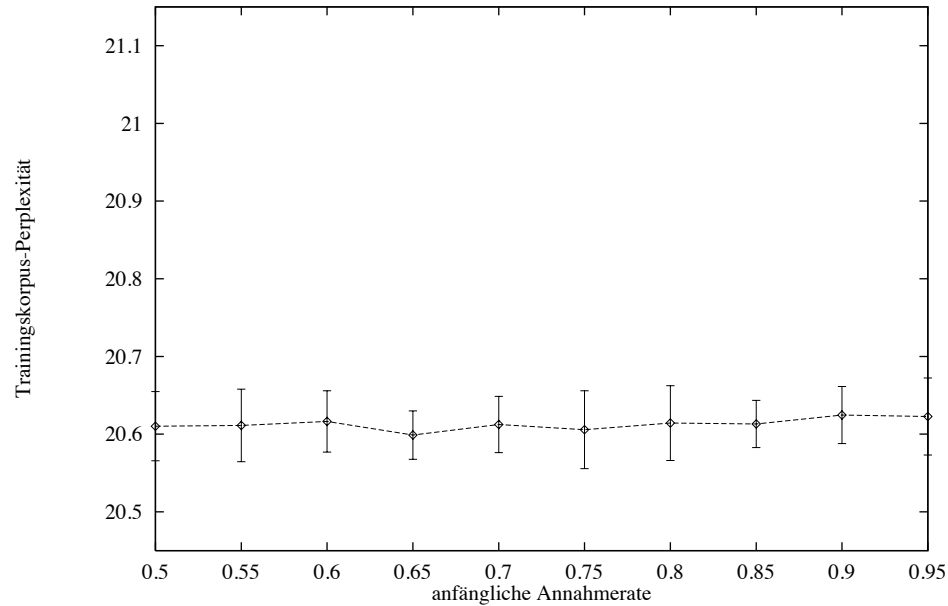


Bild B.9: Trainingskorpus-Perplexität bei simuliertem Ausfrieren für verschiedene Werte der anfänglichen Annahmerate γ_0 (Intercity-Korpus, 80 Kat.).

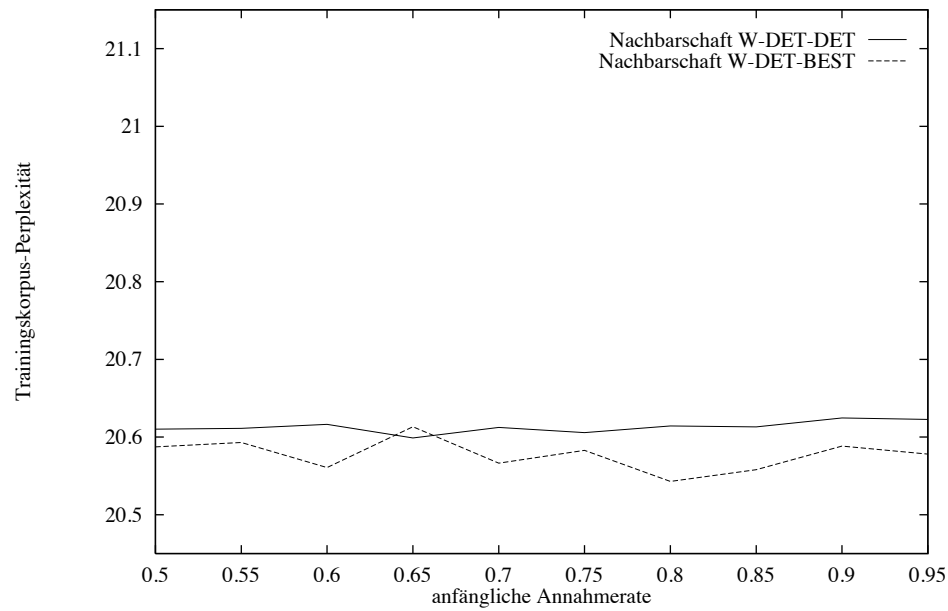


Bild B.10: Trainingskorpus-Perplexität bei simuliertem Ausfrieren für verschiedene Werte der anfänglichen Annahmerate γ_0 (Intercity-Korpus, 80 Kat., W-DET-DET).

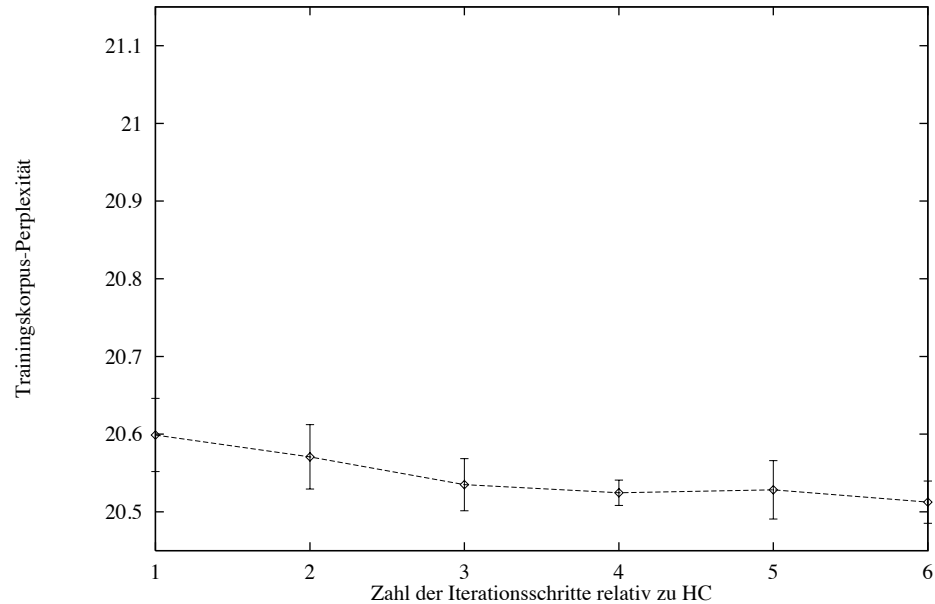


Bild B.11: Trainingskorpus-Perplexität bei Schwellwertakzeptanz bei unterschiedlich langsamer Abkühlung (Intercity-Korpus, 80 Kat., W-DET-DET).

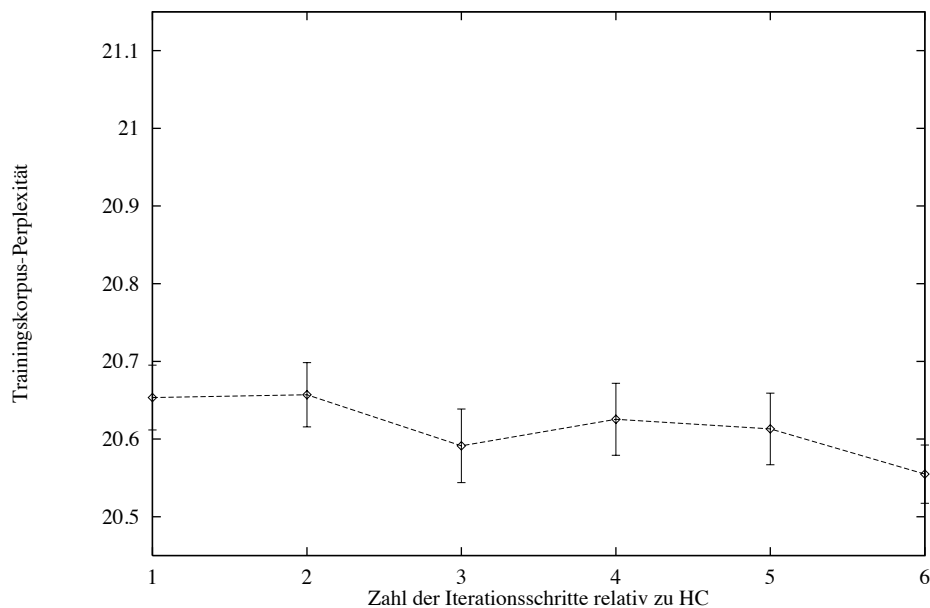


Bild B.12: Trainingskorpus-Perplexität bei simuliertem Ausfrieren bei unterschiedlich langsamer Abkühlung (Intercity-Korpus, 80 Kat., W-DET-DET).

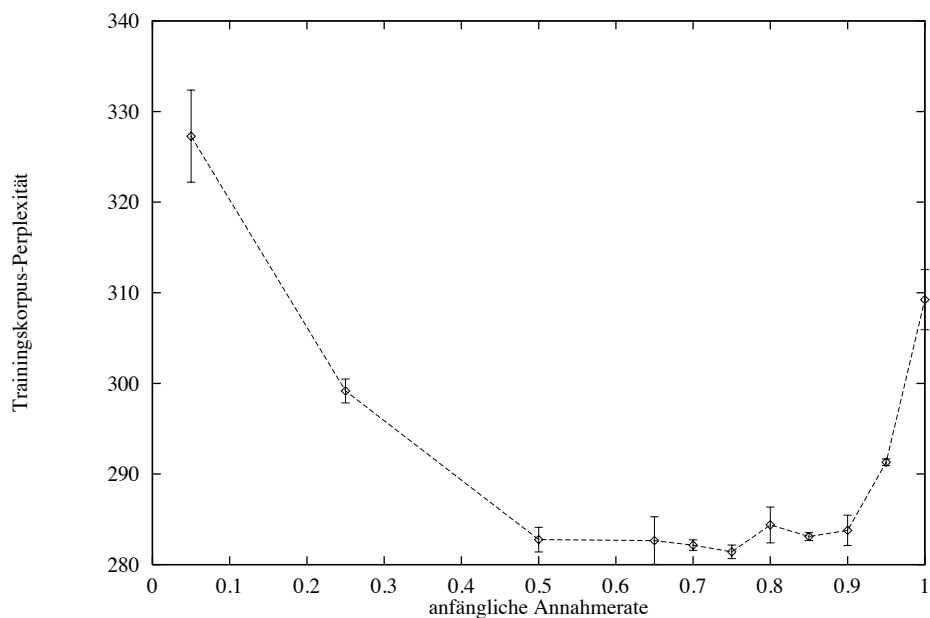


Bild B.13: Trainingskorpus-Perplexität bei Record-To-Record Travel für verschiedene Werte der anfänglichen Annahmerate γ_{RRT} (Limas-04-Korpus, 100 Kat., W-DET-DET).

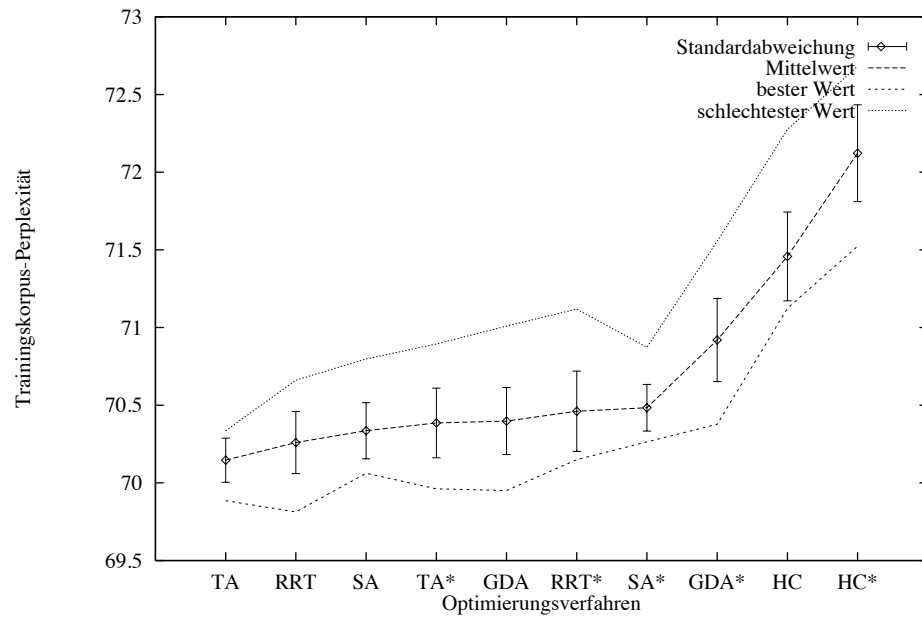


Bild B.14: Vergleich verschiedener Optimierungsverfahren anhand der Trainingskorpus-Perplexität (Verbmobil-Korpus, 100 Kat.).

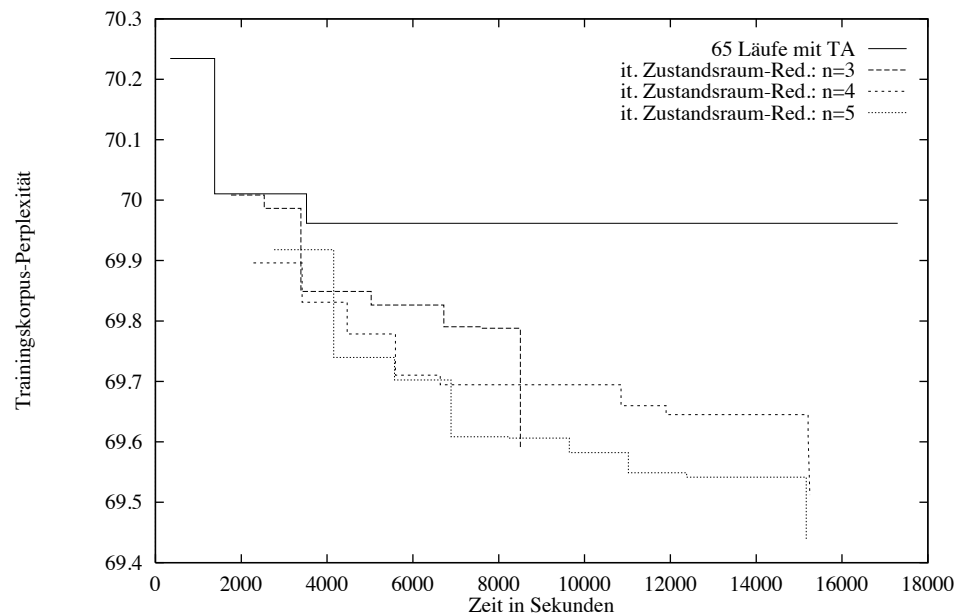


Bild B.15: Ergebnisse einer großen Zahl von Läufen mit Schwellwertakzeptanz und mit iterierter Zustandsraum-Reduktion (Verbmobil-Korpus, 100 Kat.).

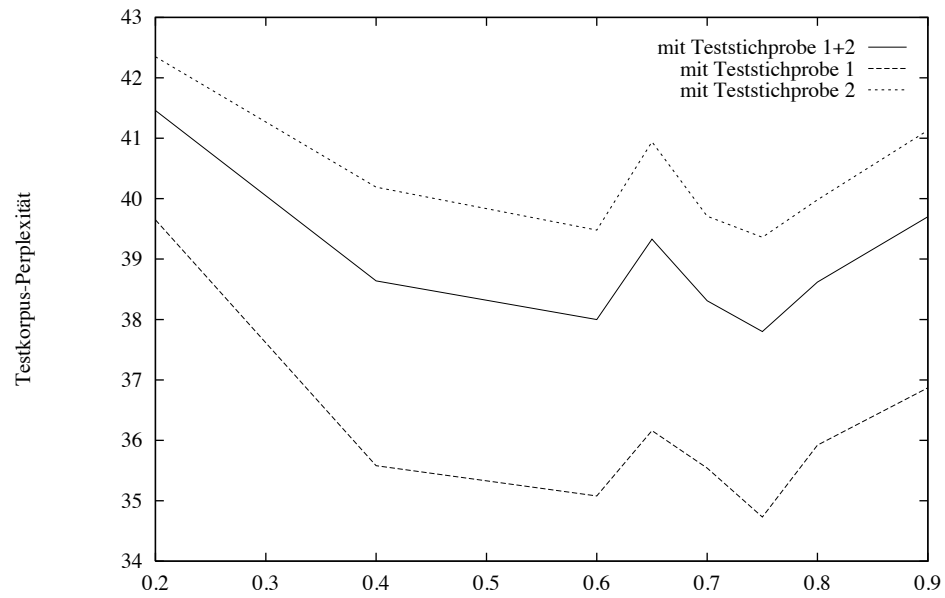


Bild B.16: Testkorpus-Perplexität für mit LO berechnete Kategoriensysteme für verschiedene ρ -Werte (Intercity-Korpus, 80 Kat.).

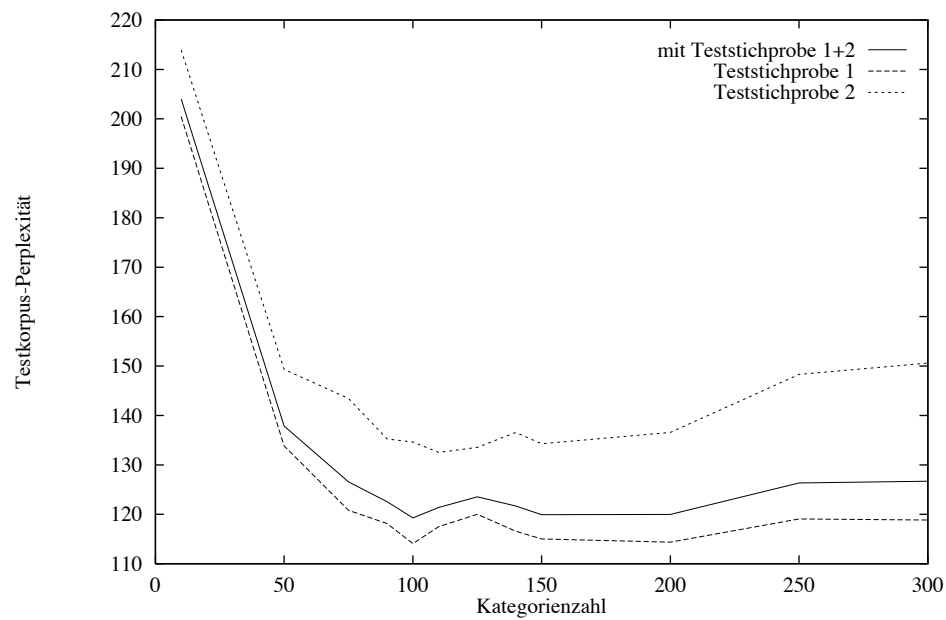


Bild B.17: Testkorpus-Perplexität für verschiedene Kategoriengrößen (Verbmobil-Korpus).

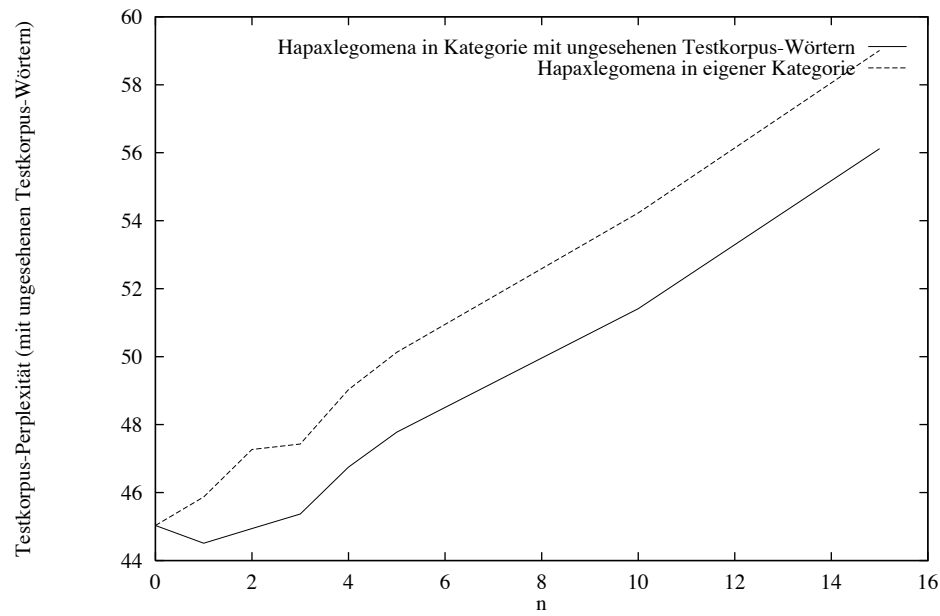


Bild B.18: Testkorpus-Perplexität mit und ohne Berücksichtigung von Hapaxlegomena (Intercity-Korpus, 80 Kat.).

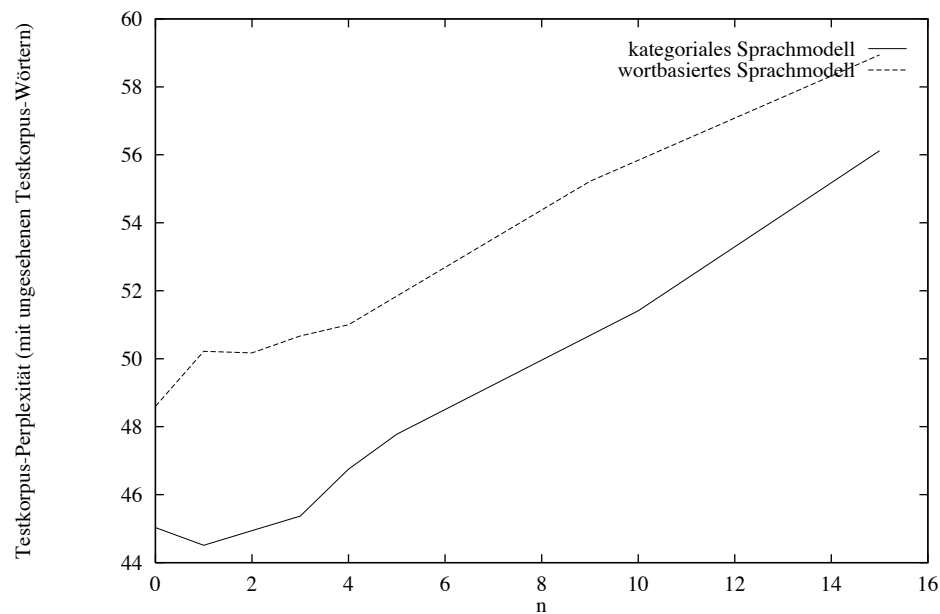


Bild B.19: Testkorpus-Perplexität eines wortbasierten Modells gegenüber berechneten Kategoriensystemen für verschiedene Hapaxlegomena-Kategorien (Intercity-Korpus, 80 Kat.).

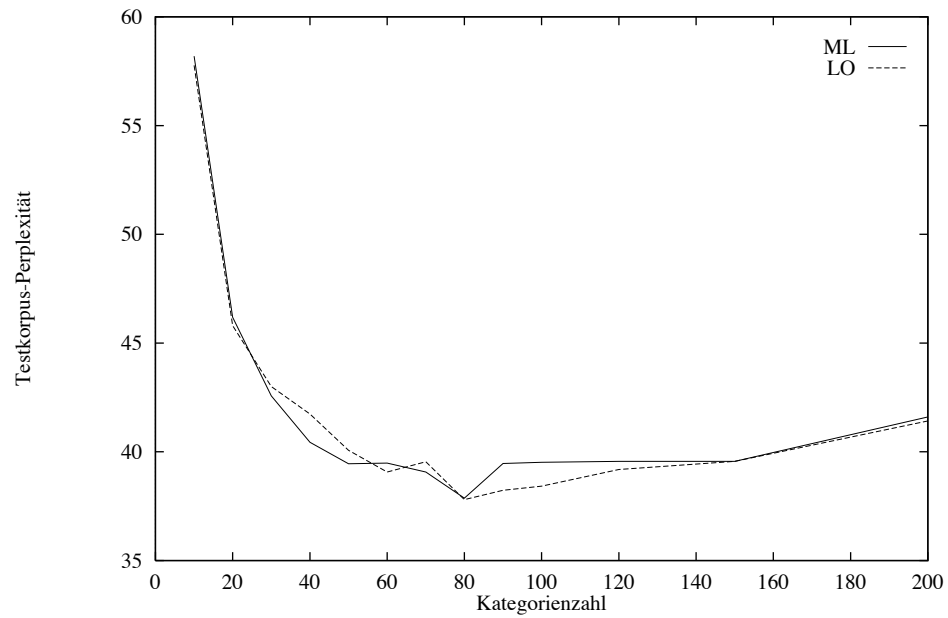


Bild B.20: Vergleich der Optimierungskriterien ML und LO anhand der Testkorpus-Perplexität (Intercity-Korpus).

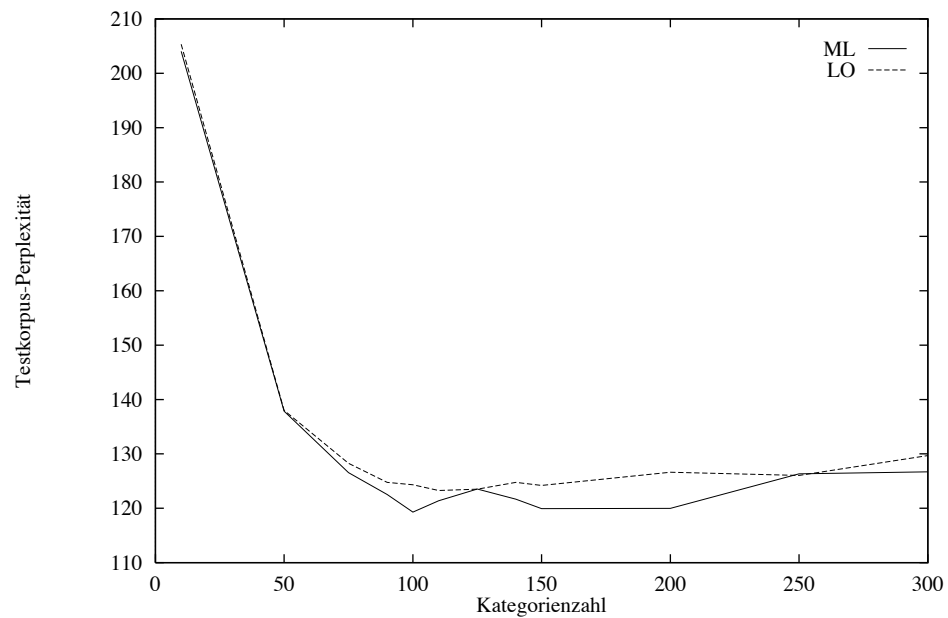


Bild B.21: Vergleich der Optimierungskriterien ML und LO anhand der Testkorpus-Perplexität (Verbmobil-Korpus).

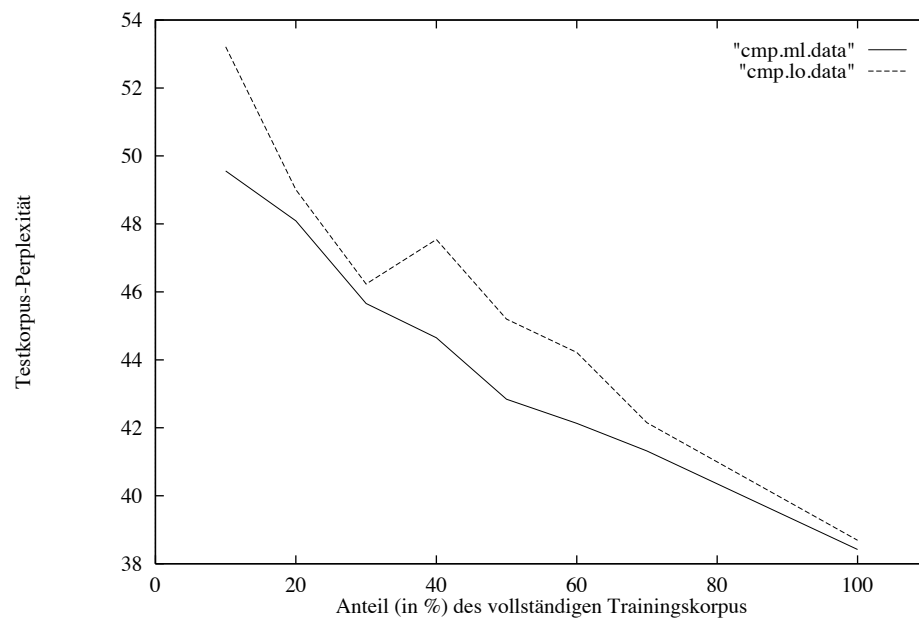


Bild B.22: Vergleich der Optimierungskriterien ML und LO anhand der Testkorpus-Perplexität mit verkleinertem Trainingskorpus (Intercity-Korpus, 80 Kat.).

Anhang C

Programm-Manual

Im Rahmen der Studienarbeit wurde das Programm `mkcls` erstellt, welches die Erzeugung von Kategoriensystemen mittels verschiedener Optimierungsverfahren ermöglicht und die Parameter-Einstellung unterstützt.

Folgende Aufrufmöglichkeiten stehen zur Verfügung:

- `mkcls [Parameter] [-Vfile] [-ssec] [-Mnr] [-ofile] opt`
Optimierung mit iterativ-optimierenden Verfahren (siehe Abschnitt 3.1)
- `mkcls [Parameter] [-Vfile] msb-opt 1|2|3`
Optimierung mit multidirektionaler Suche mit Beschneidung (siehe Abschnitt 3.2). Die Zahl nach `msb-opt` kennzeichnet die Beschneidungstabelle, die verwendet werden soll (MSB-1, MSB-2, MSB-3; siehe Seite 59).
- `mkcls [Parameter] [-Vfile] msb-opt ‘‘[’]’ nr00 nr10 ... nr90 ‘[’]’`
Optimierung mit multidirektionaler Suche mit Beschneidung (siehe Abschnitt 3.2). Durch `nr00`, `nr10`, `nr20` ... `nr90` werden die Werte der Beschneidungstabelle zu Beginn, nach 10%, nach 20% usw. angegeben.
- `mkcls [Parameter] [-Vfile] [-ssec] izr-opt`
Optimierung mit iterierter Zustandsraum-Reduktion (siehe Abschnitt 3.3)
- `mkcls [Parameter] [-tnr] meta-opt parameter`
Durchführung einer Parameter-Einstellung für iterativ-optimierende Verfahren. Der Wert *parameter* gibt den einzustellenden Parameter an (siehe Tabelle C.1). Da im Rahmen dieser Studienarbeit eine umfangreiche Einstellung für alle Parameter durchgeführt wurde (siehe Kapitel 5) dürfte diese Aufrufmöglichkeit nur in Spezialfällen verwendet werden müssen.

Bei jedem Aufruf können die folgenden (optionalen) Parameter verwendet werden:

- `-vflag` Verbose-Modus (Default: 0, also keine zusätzlichen Ausgaben)

parameter	1	2	3	4	5	6	7	8
Parameter	γ_0 (SA)	γ_e (SA)	ν_{SA}	γ_{TA}	ν_{TA}	γ_{RRT}	ν_{RRT}	α (GDA)

Tabelle C.1: Bedeutung von **parameter** bei der Einstellung und der Bestimmung der Parameter für iterativ-optimierende Verfahren (Option -e bzw. Kommando **meta-opt**).

- **-nnr** Anzahl der Optimierungsläufe (bei **izr-opt**: Default 3, ansonsten Default 1): Bei der iterierten Zustandsraum-Reduktion (**izr-opt**) entspricht dies der Anzahl der an der Infimum-Bildung beteiligten lokalen Minima (Parameter n).
- **-l[nr]** Optimierungskriterium LO: Normalerweise wird das Optimierungskriterium ML verwendet. Der optionale Wert nr bestimmt den Parameter ρ (Default: 0.75).
- **-pfile** Filename des Trainingskorpus (Default: **lern**)
- **-init [file]** Initialisierung (Default: 1): Die Einstellungsmöglichkeiten für *init* sind **ran** (I-RAN), **aio** (I-AIO), **lwrw** (I-LWRW), **freq** (I-FREQ), **other** (I-OTHER). Bei der Initialisierung I-OTHER muß noch das File angegeben werden, von dem das Kategoriensystem gelesen werden soll.
- **-wword** Wortauswahl (Default: **det**): Hierdurch kann die Wortauswahl auf **det** (deterministisch) oder **ran** (zufällig) gesetzt werden. Zusammen mit der Kategorienauswahl (Option **-k**) ergeben sich die Nachbarschaftsauswahlen aus Abschnitt 4.3.
- **-kcat** Kategorienauswahl (Default: **best**): Hierdurch kann die Kategorienauswahl auf **det** (deterministisch), **ran** (zufällig) oder **best** (optimal) gesetzt werden.
- **-cnr** Anzahl der Kategorien (Default: 100)
- **-mnr** Grenzwert für Hapaxlegomena (Default: 0, keine Hapaxlegomena): Durch diese Option kann eingestellt werden, ab wann ein Wort als Hapaxlegomenon betrachtet werden soll und in eine dafür bereitgestellte Kategorie transportiert wird. Beispielsweise beim Wert 3 werden alle Wörter, die 3 mal oder seltener auftreten als Hapaxlegomena interpretiert und in einer im Laufe der Optimierung unveränderlichen Kategorie zusammengefaßt.
- **-eparameter nr** Einstellung der Parameter für iterativ-optimierende Verfahren. Es wird dabei der Parameter *parameter* auf den durch nr gegebenen Wert gesetzt. Dabei besitzt *parameter* die in Tabelle C.1 beschriebene Bedeutung. Die Default-Einstellung ist durch Tabelle 5.1 gegeben und es ist zu erwarten, daß diese normalerweise nicht geändert werden muß, da schon eine umfangreiche Einstellung für alle Parameter durchgeführt wurde (siehe Kapitel 5).
- **-rnr** Initialisierung des Zufallszahlengenerators

Die anderen Parameter haben die folgende Bedeutung:

- **-averfahren** Iterativ-optimierendes Verfahren (Default: **ta**, Schwellwertakzeptanz): Die Einstellungsmöglichkeiten sind stochastische Relaxation (**hc**), simuliertes Ausfrieren (**sa**), Schwellwertakzeptanz (**ta**), Record-To-Record Travel (**rrt**) und Simulated Annealing (**gda**).
- **-Vfile** Filename für Kategoriensystem: Es werden zwei Files erzeugt. In *file* steht für jedes Wort seine Kategorie. In *file.cats* steht für jede Kategorie eine Liste der darin enthaltenen Wörter.

Beispiel für *file*:

```
Baum CAT:2
Haus CAT:2
essen CAT:1
trinken CAT:1
```

Beispiel für *file.cats*:

```
CAT:1: essen, trinken
CAT:2: Baum, Haus
```

- **-Mnr** Maximale Zahl der Iterationen für einen Optimierungslauf (Default: -1: kein Maximum)
- **-ssec** Grenze für die Rechenzeit in Sekunden (Default: -1, keine Grenze): Gibt an, nach wievielen Sekunden die Optimierung abgebrochen werden soll. Ein gerade bearbeiteter Optimierungslauf wird jedoch bis zum Ende berechnet.
- **-Nnr** Anzahl der Parameter-Einstellungen, die bei der Option **meta-opt** verglichen werden sollen (Default: 10).
- **-ofile** Gibt den Wert der Kostenfunktion und die ‘Temperatur’ des Optimierungsverfahrens nach jeweils 1000 Optimierungsschritten aus. Dadurch ist es möglich, den Verlauf der Kostenfunktion z. B. mit dem Programm **gnuplot** zu visualisieren.

Anhand einiger Beispiele sollen die wichtigsten Aufrufmöglichkeiten erläutert werden:

mkcls -c80 opt

Es wird ein Optimierungslauf mit Schwellwertakzeptanz durchgeführt. Dabei wird das Trainingskorpus aus der Datei **lern** verwendet und es wird ein Kategoriensystem mit 80 Kategorien erzeugt.

mkcls -c80 -n10 -pic/lern opt

Es werden 10 Optimierungsläufe mit Schwellwertakzeptanz durchgeführt. Dabei wird das Trainingskorpus **ic/lern** verwendet und es werden Kategoriensysteme mit 80 Kategorien erzeugt.

mkcls -c80 -n1000 -s3600 -Vic/kats80 opt

Es werden maximal 1000 Optimierungsläufe auf dem Trainingskorpus **lern** durchgeführt, wobei nach einer Stunde kein weiterer Optimierungslauf gestartet wird. Das beste Kategoriensystem wird in die Datei **ic/kats80** geschrieben.

```
mkcls -c80 -n10 -s36000 -Vkats izr-opt
```

Es wird eine iterierte Zustandsraum-Reduktion mit Parameter $n = 10$ auf dem Trainingskorpus **lern** durchgeführt. Die Kategorienzahl beträgt 80 Kategorien. Nach 10 Stunden wird keine weitere Zustandsraum-Reduktion durchgeführt und die Optimierung abgebrochen. Das sich ergebende Kategoriensystem wird in die Datei **kats** geschrieben.

Literaturverzeichnis

- [Ack87] Ackley, D. H.: *An Empirical Study of Bit Vector Function Optimization*, in Davis, L. (Hrsg.): *Genetic Algorithms and Simulated Annealing*, Kap. 13, Pitman, London, 1987, S. 170–204.
- [Bah86] Bahl, L.; Brown, P.; De P., Souza; Mercer, R.: *Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition*, in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Tokyo, 1986, S. 49–52.
- [Boo89] Booker, L.; Goldberg, D.; Holland, J.: *Classifier Systems and Genetic Algorithms*, *Artificial Intelligence*, Bd. 40, Nr. 1–3, 1989, S. 235–282.
- [Bro90] Brown, P.; Cocke, J.; Della S., Pietra; DellaPietra, V.; Jelinek, F.; Lafferty, J.; Mercer, R.; Roossin, P.: *A Statistical Approach to Machine Translation*, *Computational Linguistics*, Bd. 16, Nr. 2, 1990, S. 79–85.
- [Dew87] Dewdney, A. K.: *Computer-Kurzweil*, *Spektrum der Wissenschaft*, November 1987, S. 8–12.
- [Dor95] Dorn, J.: *Iterativ Improvement Methods for Knowledge-Based Scheduling*, *Ai Communications*, Bd. 8, Nr. 1, March 1995, S. 20–34.
- [Due90] Dueck, G.; Scheuer, T.: *Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing*, *Journal of Computational Physics*, Bd. 90, Nr. 1, 1990, S. 161–175.
- [Due93] Dueck, G.: *New optimization heuristics: The great deluge algorithm and the record-to-record-travel*, *Journal of Computational Physics*, Bd. 104, Nr. 1, 1993, S. 86–92.
- [Eck92] Eckert, W.; Fink, G.; Kießling, A.; Kompe, R.; Kuhn, T.; Kummert, F.; Mast, M.; Niemann, H.; Nöth, E.; Prechtel, R.; Rieck, S.; Sagerer, G.; Scheuer, A.; Schukat-Talamazzini, E.; Seestaedt, B.: *EVAR: Ein sprachverstehendes Dialogsystem*, in Görz, G. (Hrsg.): *KONVENS 92*, Informatik aktuell, Springer, Berlin, 1992, S. 49–58.
- [Eco94] Eco, U.: *Die Suche nach der vollkommenen Sprache*, C. H. Beck, München, 1994.

- [Fis94] Fischer, V.: *Parallelverarbeitung in einem semantischen Netzwerksystem für die wissensbasierte Musteranalyse*, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, 1994.
- [Gor90] Gorlen, K. E.; Orlow, S.; Plexico, P. S.: *Data Abstraction and Object Oriented Programming in C++*, John Wiley and Sons, Chichester, 1990.
- [Gra94] Graham, R. L.; Knuth, D. E.; Patashnik, O.: *Concrete Mathematics*, Addison-Wesley Publishing Company, Reading, Massachusetts, 2. Ausg., 1994.
- [Gre87] Grefenstette, J. J.: *Incorporating Problem Specific Knowledge into Genetic Algorithms*, in Davis, L. (Hrsg.): *Genetic Algorithms and Simulated Annealing*, Kap. 4, Pitman, London, 1987, S. 42–60.
- [Hen95] Hendrych, R.: *Permutierte Polygramme zur grammatischen Sprachmodellierung*, Diplomarbeit, Lehrstuhl für Informatik 5 (Mustererkennung), Universität Erlangen-Nürnberg, 1995.
- [Hol75] Holland, J.: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich., 1975.
- [Jar93a] Jardino, M.; Adda, G.: *Automatic Word Classification Using Simulated Annealing*, in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Bd. 2, Minneapolis, 1993, S. 41–44.
- [Jar93b] Jardino, M.; Adda, G.: *Language Modelling for CSR of Large Corpus Using Automatic Classification of Words*, in *Proc. European Conf. on Speech Technology*, 1993, S. 1191–1194.
- [Jel82] Jelinek, F.; Mercer, R.; Bahl, L.: *Continuous Speech Recognition*, in Krishnaiah, P.; Kanal, L. (Hrsg.): *Handbook of Statistics*, Bd. 2, North-Holland, 1982, S. 549–573.
- [Jel90] Jelinek, F.: *Self-Organized Language Modeling for Speech Recognition*, in Waibel, A.; Lee, K. (Hrsg.): *Readings in Speech Recognition*, Morgan Kaufmann, San Mateo, CA, 1990, S. 450–506.
- [Joh86] Johnson, D.; Aragon, C.; McGeoch, L.; Schevon, C.: *Optimization by Simulated Annealing: an Experimental Evaluation*, in *List of Abstracts, Workshop on Statistical Physics in Engineering and Biology. Revised Version.*, Yorktown Heights, N.J., 1986.
- [Kir83] Kirkpatrick, S.; Gelatt Jr., C.; Vecchi, M.: *Optimization by Simulated Annealing*, *Science*, Bd. 220, 1983, S. 671–680.
- [Kne91] Kneser, R.; Ney, H.: *Forming Word Classes by Statistical Clustering for Statistical Language Modelling*, in *1. Quantitative Linguistics Conference*, 1991.

- [Kne93] Kneser, R.; Ney, H.: *Improved Clustering Techniques for Class-Based Statistical Language Modelling*, in *Proc. European Conf. on Speech Technology*, 1993, S. 973–976.
- [Kuh94a] Kuhn, T.: *Die Erkennungsphase in einem Dialogsystem*, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, Erlangen, 1994.
- [Kuh94b] Kuhn, T.; Niemann, H.; Schukat-Talamazzini, E.: *Ergodic Hidden Markov Models and Polygrams for Language Modeling*, in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Bd. 1, Adelaide, Australia, 1994, S. 357–360.
- [Kyb93] Kybelksties, D.: *Schätzung stochastischer Grammatiken für das Dialogsystem EVAR*, Studienarbeit, Lehrstuhl für Informatik 5 (Mustererkennung), Universität Erlangen-Nürnberg, 1993.
- [Laa87] Laarhoven, P. v.; Aarts, E.: *Simulated Annealing: Theory and Applications*, John Wiley & Sons, New York, 1987.
- [Mas94] Mast, M.; Kummert, F.; Ehrlich, U.; Fink, G.; Kuhn, T.; Niemann, H.; Sagerer, G.: *A Speech Understanding and Dialog System with a Homogeneous Linguistic Knowledge Base*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 16, Nr. 2, 1994, S. 179–194.
- [Met53] Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; Teller, E.: *Equation of state calculations for fast computing machines*, *Journal of Chemical Physics*, Bd. 21, Nr. 6, 1953, S. 1087–1092.
- [Muc92] Muchla, H.-J.: *Clusteranalyse mit Mikrocomputern*, Akademie Verlag, Berlin, 1992.
- [Mül87] Müller, H.: *Diskrete Algebraische Strukturen - Stichworte, Definitionen und Sätze*, Institut für mathematische Maschinen und Datenverarbeitung, Universität Erlangen-Nürnberg, Erlangen, April 1987.
- [Ney92] Ney, H.: *Automatische Spracherkennung: Architektur und Suchstrategie aus statistischer Sicht*, *Informatik Forschung und Entwicklung*, Bd. 7, 1992, S. 83–97.
- [Ney93] Ney, H.: *Estimating 'Small' Probabilities by Leaving-one-out*, in *Proc. European Conf. on Speech Technology*, September 1993, S. 973–976.
- [Nur93] Nurmela, K. J.: *Constructing Combinatorial Designs by Local Search*, 27, Digital Systems Laboratory, November 1993.
- [Pau90] Paul, D.; Baker, J.; Baker, J.: *On the Interaction Between True Source, Training, and Testing Language Models*, in *Speech and Natural Language Workshop*, Morgan Kaufmann, Hidden Valley, Pennsylvania, 1990, S. 185–189.

- [Rab93] Rabiner, L.; Juang, B.-H.: *Fundamentals of speech recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [Sch94a] Schukat-Talamazzini, E.; Kuhn, T.; Niemann, H.: *Speech Recognition for Spoken Dialog Systems*, in Niemann, H. (Hrsg.): *Progress and Prospects of Speech Research and Technology*, CRIM/FORWISS, Infix, 1994.
- [Sch94b] Schukat-Talamazzini, E. G.: *Automatische Spracherkennung*, Habilitation, Technische Fakultät der Universität Erlangen-Nürnberg, 1994.
- [Str92] Stroustrup, B.: *Die C++ Programmiersprache*, Addison-Wesley Publishing Company, Bonn, 2. Ausg., 1992.
- [Wit92] Witschel, P.; Niedermair, G.: *Experiments in Dialogue Context Dependent Language Modelling*, in Görz, G. (Hrsg.): *KONVENS 92*, Informatik aktuell, Springer, Berlin, 1992, S. 395–399.