KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK — ESAT
Kasteelpark Arenberg 10, B–3001 Leuven-Heverlee

PROBABILISTIC LANGUAGE MODELING WITH LEFT CORNER PARSING

Promotor:
Prof. Dr. ir. P. Wambacq

Co-Promotor:
Prof. Dr. ir. D. Van Compernolle

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Dong Hoon VAN UYTSEL

September 2003

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK — ESAT
Kasteelpark Arenberg 10, B–3001 Leuven-Heverlee

PROBABILISTIC LANGUAGE MODELING WITH LEFT CORNER PARSING

Jury:

Prof. Dr. ir. L. Froyen, voorzitter
Prof. Dr. ir. P. Wambacq, promotor
Prof. Dr. ir. D. Van Compernolle, promotor
Prof. Dr. R. Bod
Prof. Dr. F. Van Eynde
Prof. Dr. ir. H. Van hamme
Prof. Dr. ir. Y. Willems

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Dong Hoon VAN UYTSEL

U.D.C. 681.3∗I27

September 2003

D/2003/7515/39

ISBN 90–5682–428–7

Aan Goedele en Eline

Voorwoord

Op het einde van mijn ingenieursopleiding nam ik me voor om deel te nemen aan ingenieursonderzoek, waarin er een plaats zou zijn voor mijn sluimerende interesse voor talen en grammatica. Automatische spraakherkenning leek een geschikt werkterrein. Het lag immers voor de hand dat luisterende computers, net zoals mensen, moeten kunnen beslissen welke woorden, als voortzetting van een gedeeltelijk uitgesproken zin, waarschijnlijk zijn, en welke niet. En dat hierin grammaticale noties een belangrijke rol moesten spelen, dat leed ook geen twijfel...

Bij mijn eerste ontmoeting met Dirk Van Compernolle, toen hoofd van de spraakgroep, bleek mijn interesse de 'statistische taalmodellering' te betreffen. Dirk toonde zich enthousiast en gaf me meteen de kans om een doctoraat bij hem te beginnen, waarvoor ik hem dankbaar ben.
Het werd echter snel duidelijk dat het idee over de bijdrage van grammaticale kennis aan het taalmodel op dat moment noch origineel, noch succesvol was in spraakherkenning met een grote woordenschat. De vooruitgang in statistische taalmodellering was grotendeels het resultaat van verfijningen van algemeen toepasbare statistische schattingstechnieken, waarin taalwetenschap geen enkele ernstige rol speelde.
De eerste jaren van mijn doctoraat gingen op aan de studie, de verbetering en de efficiente toepassing van reeds gekende taalmodelleringstechnieken. Hierbij kwam heel wat programmeerwerk en aan te pas; voor computerproblemen of programmeervragen kon ik op bijna elk willekeurig moment terecht bij mijn collega Kris Demuynck, die graag zijn technische ervaring met me deelde. Ik dank hem daarvoor van harte.
In deze periode deed zich ook de kans voor om een jaar lang te werken als gastonderzoeker bij de Interactive Systems Labs (ISL) te Karlsruhe en Pittsburgh. Daar onderzocht ik onder andere methoden voor de aanpassing van een taalmodel aan een andere stijl, onder meer door de statistieken van woordcategorieën te scheiden van de statistieken van de gebruikte woorden zelf. Dit verblijf was mogelijk met de steun van het IWT, en met de hulp van Alex Waibel, hoofd van de ISL. Wat ik van deze ervaring heb meegedragen, is vooral de praktische vertrouwdheid met conventionele taalmodelleringstechnieken, maar ook de inspiratie die ik opdeed uit vele leerzame gesprekken met Klaus Ries, John Lafferty en Roni Rosenfeld.
Na mijn terugkomst uit Pittsburgh deed de gelegenheid zich voor om in samenwerking met Filip Van Aelten, Kristin Daneels en Marc Hogenhout van Lernout&Hauspie een

project te starten dat zou voortbouwen op het ‘gestructureerde taalmodel’ van Ciprian Chelba en Fred Jelinek. Het project was riskant: er viel veel programmeer- en experimenteerwerk te doen, vooraleer duidelijk zou kunnen zijn of we effectief wetenschappelijke vooruitgang zouden realiseren — wat achteraf gelukkig wel bleek. Deze aangename en stimulerende samenwerking bracht me weer dichter bij mijn originele motivatie om aan onderzoek op taalmodellering te doen. Het was Filip die mijn aandacht vestigde op linkerhoekontleding als mogelijke basis voor een nieuw taalmodel, en daarvoor ben ik hem zeer erkentelijk.

Bij de voltooiing van deze tekst, en ter gelegenheid van de openbare verdediging ervan, dank ik oprecht de leden van mijn jury: Dirk Van Compernolle, Patrick Wambacq, Ludo Froyen, Rens Bod, Frank Van Eynde, Hugo Van hamme en Yves Willems, voor de tijd en moeite die ze hebben besteed aan het nalezen van de tekst en voor de talloze suggesties die de tekst leesbaarder en correcter hebben gemaakt. Tom Laureys en Sigrid Maene hebben ook gedeelten van de tekst nagelezen, wat zeer nuttig bleek. Hartelijk dank.

Samenvatting

Deze doctoraatsstudie levert een bijdrage tot het onderzoeksdomein van de statistische taalmodellering. Hierin wordt de menselijke taalproductie als een probabilistische datastroom van zinnen voorgesteld. Het doel is optimale schatters te vinden voor de waarschijnlijkheid waarmee een willekeurige zin wordt geproduceerd, binnen de context van een natuurlijke-taaltoepassing. Statistische taalmodellen zijn een essentieel onderdeel van o.a. automatische continue spraakherkenning met een groot vocabularium, naast andere natuurlijke-taaltoepassingen.

Het onderzoek beschreven in deze thesis is toegespitst op de klasse van grammaticagebaseerde taalmodellen. In tegenstelling tot andere conventionele taalmodelleringstechnieken baseren deze modellen de schatting van de probabiliteit van een zin, die observeerbaar is, op een ambigue intermediaire predictie van de grammaticale structuur van deze zin, die niet observeerbaar is. Deze aanpak werd lange tijd niet geschikt geacht voor statistische spraakherkenning met een groot vocabularium. Samen met enkele andere recente bijdragen, toont dit doctoraatswerk aan dat wezenlijke verbeteringen van de kwaliteit van het taalmodel toch mogelijk zijn door rekening te houden met grammaticale structuur.
Het belangrijkste concrete resultaat van dit werk is de ontwikkeling van een taalmodel, dat een gelexicaliseerde linkerhoekzinsontledingsstrategie aanwendt. Mijn aanpassing van deze welbekende strategie gebruikt een ruimte- en tijdsefficiënte voorstelling en uitbreiding van verscheidene gedeeltelijke zinsontledingen en hun probabilititeiten. Voor de initialisatie van het taalmodel is een verzameling handmatig of automatisch ontlede zinnen nodig, maar het model optimaliseert zich nadien verder op ongeanalyseerd tekstmateriaal. Het laat tevens toe om de conditionele probabiliteit van een woord te voorspellen, uitgaande van de daaraan voorafgaande woorden. Dit is belangrijk voor de integratie met andere conventionele taalmodelleringstechnieken, en voor een zo vroegtijdig mogelijke combinatie met de andere kennismodules door het zoekmechanisme van een spraakherkenner, zodat de zoektocht gerichter kan gebeuren.

Tenslotte wordt aangetoond dat het voorgestelde taalmodel verbeteringen in spraakherkenningsnauwkeurigheid verwezenlijkt op de ‘Wall Street Journal’-taak, dit ten opzichte van een combinatie van een woordgebaseerd trigrammodel en een klassegebaseerd 4-grammodel. De geobserveerde verbetering ten opzichte van een ander

recent grammatica-gebaseerd taalmodel is kleiner maar bij gelijke prestaties vereist het taalmodel, dat gebaseerd is op linkerhoekzinsontleding, slechts een fractie van de leertijd van het andere.

Een uitgebreide samenvatting is opgenomen als Appendix E vanaf bladzijde 123.

Summary

This thesis contributes to the research domain of statistical language modeling. In this domain, the human generation of natural language is represented as a probabilistic data stream of sentences. Language modeling research attempts to find optimal estimators of the probability that some sentence is produced, albeit within the context of a given natural language application. Statistical language models are an essential part of automatic speech recognition systems, amongst other natural language applications.

The research described in this thesis is limited to the class of grammar-based language models. In contrast with other conventional language modeling techniques, these models predict the probability of the input sentence, which is observable, based on an ambiguous intermediate prediction of the grammatical structure of that sentence, which is not observable. This approach was believed unsuited for statistical large-vocabulary speech recognition. Together with a few other recent publications, this doctorate study shows that taking grammatical structure into account enables significant improvements in language model performance.
The most important concrete result is the development of a language model that follows a lexicalized left corner parsing strategy. My adaptation of this well-known parsing strategy represents several concurrent sentence analyses and their probabilities efficiently in both time and space. The proposed language model is initialized with a set of hand- or machine-parsed sentences, but is then optimized on plain text. It allows to predict the conditional probability of a next word, given the preceding ones. This is important for integration with other conventional language modeling techniques, and for the early combination with the other knowledge sources by the search engine of a speech recognizer.

Finally the proposed language model is shown to improve the recognition accuracy on the Wall Street Journal task, with respect to a combination of a word-based trigram model and a class-based 4-gram model. The observed improvement with respect to another recently published grammar-based model is smaller, but at equal performance levels, the left corner parsing model only requires a fraction of the learning time of the other model.

Contents

Contents **xiii**

List of notations and acronyms **xvii**

List of Figures **xxi**

List of Tables **xxiii**

Introduction **xxv**

1 A background of statistical language modeling **1**
1.1 What is a statistical language model? 1
1.1.1 Representing language 1
1.1.2 The probability of a sentence 2
1.1.3 The probability of the next word 4
1.2 Application of statistical language models 4
1.3 Estimating CLMs from observed data 6
1.3.1 Measuring language model performance 6
1.3.2 Maximum-likelihood estimation and the need for smoothing . 8
1.4 Simple language model classes 9
1.4.1 Word-based *n*-grams 10
1.4.2 Category-based *n*-grams 11
1.4.3 Delayed *n*-grams 12
1.4.4 Cache-based models and triggers 12
1.4.5 Grammar-based language models 13
1.5 Discounting 14
1.5.1 Laplace's law of succession 14
1.5.2 Jelinek-Mercer smoothing (deleted or held-out estimation) . . 15
1.5.3 Good-Turing and Katz discounting 16
1.5.4 Nádas smoothing 18
1.5.5 Absolute discounting 19
1.6 Model combination 20

1.6.1 Katz back-off . . . 20
1.6.2 Back-off using linear and non-linear interpolation . . . 20
1.6.3 A special case: Kneser-Ney modified back-off distribution . . 21
1.6.4 Maximum-entropy modeling . . . 23
1.6.5 Log-linear interpolation . . . 24
1.7 Empirical comparison of language modeling techniques . . . 24
1.8 Conclusion . . . 27

2 Grammar-based language models 29
2.1 Motivation . . . 29
2.2 Grammatical theories and generalization . . . 30
2.3 Finite-state grammars . . . 32
2.4 Context-free grammars . . . 33
2.5 Beyond context-free . . . 36
2.5.1 Unification-based grammars . . . 36
2.5.2 Dependency and link grammars . . . 38
2.5.3 History-based grammars . . . 41
2.5.4 Current state of the art: grammar-based LMs for LVCSR . . . 41
2.6 Conclusion . . . 47

3 A language model based on probabilistic left corner parsing 49
3.1 Problem formulation and methods . . . 49
3.2 Definitions and notation . . . 51
3.2.1 Stochastic grammars and parsing . . . 51
3.2.2 Parsing and language models . . . 52
3.2.3 Push-down automata . . . 53
3.3 Probabilistic left corner parsing . . . 54
3.3.1 Non-probabilistic left corner parsing . . . 54
3.3.2 Probabilistic left corner parsing: previous art . . . 57
3.3.3 Probabilistic left corner parsing: extension . . . 58
3.3.4 Inducing a probabilistic left corner grammar from a treebank . 65
3.3.5 A toy example . . . 67
3.3.6 Summary . . . 68
3.4 PLCG parsing in a compact network . . . 68
3.4.1 Efficient representation of PLCG derivations . . . 68
3.4.2 The PLCG network . . . 69
3.4.3 Computing sums of path probabilities . . . 70
3.4.4 Synchronous parsing algorithm . . . 73
3.4.5 Pruning . . . 74
3.5 The PLCG-based language model . . . 75
3.5.1 The PLCG-based language model . . . 75
3.5.2 The PLCG-based conditional language model . . . 76

3.5.3 Maximum-likelihood training 77
3.6 Summary 80

4 Experiments with the PLCG-based language model 83
4.1 Parameterization and optimization of the submodels 83
4.1.1 Modeling 83
4.1.2 Measurements 90
4.2 Rescoring speech recognition hypotheses lists 94
4.2.1 Modeling 94
4.2.2 Word error rate 95
4.2.3 Grammaticality 96
4.3 Summary 97

5 Conclusion and perspectives 99
5.1 Original contributions 99
5.2 Perspectives 100

Bibliography 103

A Proofs of Lemma's 1 and 2 115

B Derivation of recursion formulas (3.17), (3.18) and (3.32) for forward, inner and outer probabilities 117

C Derivation of the expected move frequency formula (3.31) 119

D The Penn Treebank Tag Set 121
D.1 Part-of-speech tags 121
D.2 Syntactic constituent labels 122
D.3 Function tags 122

E Extended Dutch summary: Probabilistische taalmodellering met linkerhoekontleding 123
E.1 Statistische taalmodellering 125
E.1.1 Wat is een statistisch taalmodel? 125
E.1.2 Toepassing van statistische taalmodellen 126
E.1.3 Taalmodellen schatten uit observaties 127
E.1.4 Clustertechnieken 128
E.1.5 Discounttechnieken 129
E.1.6 Modelcombinatie 131
E.1.7 Empirische vergelijking van taalmodelleringstechnieken . . . 132
E.1.8 Besluit 133
E.2 Grammatica-gebaseerde taalmodellen 133

E.2.1 Motivatie 133
E.2.2 Grammatica-gebaseerde taalmodellen: een overzicht 134
E.2.3 Besluit 138
E.3 Een taalmodel gebaseerd op probabilistische linkerhoekontleding . . . 138
E.3.1 Probleemstelling en methodologie 138
E.3.2 Probabilistische linkerhoekontleding 139
E.3.3 PLCG-ontleding in een compact netwerk 140
E.3.4 Het PLCG-gebaseerde taalmodel 141
E.3.5 Besluit 141
E.4 Experimenten met het PLCG-gebaseerde taalmodel 142
E.4.1 Parametrisatie en optimalisatie van de submodellen 142
E.4.2 N-best-lijsten van spraakherkenningshypothesen herscoren . . 143
E.5 Besluit en perspectieven 144
E.5.1 Oorspronkelijke bijdragen 144
E.5.2 Perspectieven 145

List of notations and acronyms

Typographic conventions

function	type	examples
new term	bold	**tokenization**
literals, tags	cursive typewriter	*</s>*, *NP*
operations	small caps	SHIFT(w)
sets of constituents, or graphs	calligraphic	$\mathscr{G}, \mathscr{H}$

Notation

notation	meaning
(a,b,c)	sequence or tuple of three variables a, b and c
ε	the empty sequence
w_i	variable denoting the word at position i
(w_i^j)	$(w_i, w_{i+1}, \ldots, w_j)$, or ε if $j < i$
abc	(a,b,c)
α	Greek lowercase letters α, β, ... denote sequences
$\alpha\beta$	concatenation of sequence α and sequence β
$\delta(e)$	equals 1 if boolean expression e is true, otherwise it equals 0
$P(x)$	'real' probability of the event x
$P(x\|y)$	'real' conditional probability of the event x given event y
$p(x)$	probability of the event x according to probability density function p
$f(x)$	probability density function of a real vector x
$p(x\|y)$	conditional probability of the event x given event y according to conditional probability mass function p
$p(x\|y)$	conditional probability of the event x given event y according to conditional probability mass function p
$\|V\|$	number of elements in V if it is a set, length of V if it is a sequence
$\hat{x}$	approximation, estimate of x
θ	set of model parameters

notation	meaning
$E[x]$	probabilistic expectation of x
$c_O(x)$	number of occurrences of x in observation sequence O
$\doteq$	'is defined as' relation
$\Rightarrow$	'derives to' relation
$\Rightarrow_L$	leftmost 'derives to' relation
$\angle$	'is a left corner of' relation
R^*	reflective and transitive closure of some relation R
$(t_1^m)_X$	local tree having mother category X and pointing to daughter local trees $t_1 \ldots t_m$
$t\Delta_i s$	local tree t is the i-th daughter of local tree s
$R(t)$	category of the local tree t
$[_i\, \alpha\, _j\, \beta]_X$	constituent (Fig. 3.1)
$[_i\, \alpha\, _j\, \beta \mid \vec{g}]_X$	constituent with local tree context $\vec{g}$ (Fig. 3.4)
$\mathrm{pos}(q)$	position index of constituent q
$\mathrm{cat}(q)$	category of constituent q
$\vdash$	valid transition between two PDA instantaneous descriptions
$\underline{X} = X/x$	composite category consisting of syntactic category X and lexical category x
$\underline{\alpha}$	sequence of composite categories
$\langle q_1, q_2 \rangle$	set of all paths from q_1 to q_2
$\mu(q)$	forward probability of q
$\nu(q)$	inner probability of q
$\xi(q)$	outer probability of q
$\prec$	'precedes' relation

Acronyms

acronym	meaning
API	advanced programming interface
ASR	automatic speech recognition
CFG	context-free grammar
CKY	Cocke-Kasami-Younger (parsing algorithm)
CLM	conditional language model
CSG	context-sensitive grammar
DP	dynamic programming
EM	expectation maximization
FSG	finite-state grammar
HBG	history-based grammar
HMM	hidden Markov model

acronym	meaning
KL	Kullback-Leibler (distance)
LM	language model
LSA	latent semantic analysis
LVCSR	large-vocabulary (automatic) continuous speech recognition
ME	maximum entropy (modeling)
ML	maximum likelihood (estimation)
MMSE	minimal mean-square error
NAB	North American Business news (corpus)
PCFG	probabilistic context-free grammar
PDA	push-down automaton
pdf	probability density function
PFSG	probabilistic finite-state grammar
PLCA	probabilistic left corner automaton
PLCG	probabilistic left corner grammar
pmf	probability mass function (discrete pdf)
POS	part-of-speech
PPL	perplexity
PTB	Penn Treebank
UBG	unification-based grammar
WER	word error rate

List of Figures

1.1 Relative cross-entropies of trigrams using different smoothing techniques at varying training data size. . . . 25
1.2 Relative cross-entropies of different language model classes using Kneser-Ney linear interpolation back-off smoothing. . . . 26
1.3 Word error rate versus cross-entropy. . . . 27

2.1 Example of a headword annotated sentence. . . . 39
2.2 A local tree and its corresponding dependency representation. . . . 39
2.3 C&J shift-reduce parser. . . . 43
2.4 A partial parse in C&J. . . . 44

3.1 A graphical representation of a constituent $q = [_i AB_j CD]_X$. . . . 52
3.2 Marking sentence boundaries in parsing-based language modeling. . . . 53
3.3 Left corner derivation of a small parse tree. . . . 55
3.4 Local tree context. . . . 60
3.5 Context inheritance after a SHIFT. . . . 61
3.6 Context inheritance after a PROJECT. . . . 62
3.7 Context inheritance after an ATTACH. . . . 63
3.8 Fragment of a PLCG network. . . . 71
3.9 The PLCG network is not Markov. . . . 71
3.10 Word-synchronous PLCG parsing algorithm. . . . 74

4.1 Example sentence from the PTB. . . . 84
4.2 Eliminating unary productions. . . . 85
4.3 Binarization. . . . 85
4.4 Preprocessed example sentence. . . . 86
4.5 Greedy parameterization optimization of the GT smoothed p_{pa} model. . . . 88
4.6 Estimated mean and standard deviation of the number of shift submodel calls at a given word position under three different pruning settings. . . . 91

C.1 Summing probabilities of all paths containing a given PROJECT move. . . . 119
C.2 Summing probabilities of all paths containing a given ATTACH move. . . . 120

List of Tables

2.1 Test set perplexities of a baseline trigram and various grammar-based language models. 46

3.1 A PLCA trace generating the sentence *<s> ann likes john </s>*. . . . 67

4.1 Conditional perplexities of selected submodel parameterizations on the PTB development set. 88
4.2 Evaluated submodel smoothing techniques. 89
4.3 Influence of pruning parameters on PPL and execution time. 90
4.4 Test set PPL of differently smoothed PLCG-based LMs. 91
4.5 Influence of reestimation on PPL. 93
4.6 Comparing PPLs obtained with other grammar-based LMs. 94
4.7 Word error rates on eval92 obtained with GT smoothed models. 95
4.8 Word error rates on eval92 obtained with DI smoothed models. 96

E.1 Vergelijking van perplexiteiten bekomen met het PLCG-gebaseerde taalmodel en andere grammatica-gebaseerde taalmodellen. 144

Introduction

Since the early years of speech recognition, statistical language modeling has been a great source of frustration to many researchers. When Jelinek and Mercer [1980] proposed the *n*-gram language model, a naive Markov model of order $n-1$, they did not anticipate that their 'first shot' would remain the state of the art for at least more than 20 years. Up till now, nobody was able to propose a replacement that is significantly more generic, more powerful, yet computationally as simple as the *n*-gram model.
On the other hand, language modeling research has been fruitful in other ways, realizing advances in statistical smoothing and model combination techniques.

Apart from these, language modeling research did yield modest improvement in speech recognition accuracy. Alternative techniques such as trigger words, cache models, mixing in topic information through LSA (latent semantic analysis) or maximum entropy modeling etc. were proven to complement the basic *n*-gram model successfully. Recent valuable experiments, combining different language modeling techniques on a large training data set (NAB, 284 million tokens), were presented by Goodman [2001a]. He obtained a cross-entropy decrease of 0.74 bits or a perplexity reduction by 40%, and a word error rate decrease of 8.9% in speech recognition experiments, relative to the word error rate obtained with a trigram language model.[1]
Unfortunately, Goodman did not evaluate *statistical grammar-based language models*. The interest in statistical grammar-based language modeling for large-vocabulary speech recognition has resurged since the mid-90s. First encouraging results were published by Chelba and Jelinek [1999]. Grammar-based language models are particularly attractive because they intend to generalize language patterns in a similar way human experts do, namely through syntactic grammars; grammars (or large collections of syntactically analyzed text) are trusted prior knowledge that other traditional language models do not exploit.

This thesis presents a novel language model based on *probabilistic left corner syntactic parsing*.[2] It is related and competitive with Chelba's and Roark's grammar-based language models [Chelba and Jelinek, 2000, Roark, 2001], but uses a different parsing

1. These evaluation measures are explicated in Sec. 1.3.1.
2. (Probabilistic) left corner parsing is reviewed in Sec. 3.3.

algorithm. Cross-entropy reductions of 0.32 bits (20% perplexity reduction) were measured, and a relative word error rate reduction of 12% when used in combination with a baseline 3-gram model. With a baseline model consisting of a word-based 3-gram and a class-based 4-gram, it was also found that interpolating the baseline model with the left corner parsing model still yields a relative word error rate reduction of 6.2%. Like other grammar-based language models, the proposed model is still computationally intensive, when compared with other traditional language modeling techniques, but its performance in terms of accuracy is remarkable.

Limitations

From a scientific point of view, I believe that the left corner parsing model's test results and Goodman's are quite substantial. However, advanced language modeling techniques cannot serve many practical purposes due to their great marginal price versus performance ratio — in building the language models, as well as in deploying them. An early version of [Goodman, 2001a] vents a similar pragmatic concern, but concludes:

> "To summarize, language modeling is a very difficult area, but not one that is completely hopeless. Basic research is still possible, and there continue to be new, if not practical, then certainly interesting language modeling results. There also appear to be a few areas in which useful language modeling research is promising. But language modeling, like most research, but perhaps more so, is not an area for the faint of heart or easily depressed." [Goodman, 2001b]

This quote again raises the question *why* getting results from advanced language modeling is difficult. There are a few boundary conditions in the formulation of the language modeling problem that limit the accuracy of the language models that can ever be obtained. The next paragraphs discuss these limitations.

1. Hidden discourse context. Virtually all characteristics of the circumstances in which a discourse takes place influence the statistics of language patterns seems endless: medium (dialog, report), topic, register (casual, formal), speaker (age, gender), listener(s), location, time (Monday morning, Saturday night), ... Language models are trained on text only (transcriptions or digitized text collections), while text is rather a result of a far more complex, but hidden, process.
The common work-around is to keep the discourse context rather constant, and estimate a language model specific for that discourse context. Therefore one needs *homogeneous* (discourse context invariant) training corpora. But then, robustness is seriously reduced, since the language model is only fit for an ever tighter domain. A more fundamental solution would start with finding language 'invariants', those elements that remain relatively unchanged over different discourse contexts. Grammar

may deliver such elements to a certain extent, which motivates the exploration of grammar-based language models.
2. Inherent heterogeneity of language. Even within a short text fragment, the discourse context changes unceasingly. Consequently, a training corpus of any significant size is never homogeneous.
3. No analysis of word structure. In the classical problem formulation, language models see natural language words as atomic units. They are blind for word-internal structure. A word evidently contains much more information than can be empirically derived from its coocurrence with other words. For instance, the word 'ontsnaafd' does not exist in Dutch, but native speakers will expect it to function as a past participle, and will recognize that it has a privative meaning. Integrating morphological knowledge into the language model would already be a significant step forward.
The current standard approach is to collect the N most frequent word tokens into a vocabulary; a word type is completely equivalent with its index into that vocabulary. N should be large enough to minimize the rate of unknown words, causing errors, but not too large to keep the language model's size manageable, and to allow sufficient training for all vocabulary words given a fixed amount of training data. For large-vocabulary speech recognition, N is typically 65,000. This naive approach is viable for English in topic-constrained settings where enough training data (ten million running tokens or more) is available. It is less suited for morphology-rich languages, and applications where the language model should be robust against unexpected changes of topic or discourse.
4. Passive learning. Language models are trained on digitized text collections. As such, they only see positive examples. Human language learning, on the other hand, is an interactive process.
5. Adverse conditions. Computational language models have to operate in 'adverse' conditions. For instance, in speech recognition, the acoustic model is inferior to its (imaginary) human equivalent, and there is no explicit inference of meaning. As a consequence, the language model has to compare hundreds or thousands of concurrent partial hypotheses, while humans need surprisingly little intellectual effort for speech recognition. This discrepancy can be compared with the enormous ambiguity faced by automatic parsing of natural language, of which most humans are unaware.

The language model presented in this thesis is subject to all of the above limitations. However, it is a modest attempt to combat the first two of them by integrating human knowledge on syntax, assuming that syntax rules are relatively invariant under different circumstances.
It is additionally limited in the following ways:

- Its scope is limited to within-sentence information.
- Syntactic structure is represented with tree graphs without crossing branches or non-local indexing. Tree nodes are annotated with simple tags, there are no features.

- Only syntactic, no semantic category labels are used.
- The model is trained on annotated treebanks; human grammatical competence is acquired indirectly, which excludes competence that only shows in sentences that happen to fall outside the training set.
- Words are not analyzed morphologically. This makes the parsing task more difficult, especially on sentences containing out-of-vocabulary words.

At the end of the thesis, possible extensions to the model are proposed to overcome some of these shortcomings.

Thesis overview

The next two chapters, 1 and 2, present introductory material. Chapter 1 introduces the reader to the field of statistical language modeling, tells where and how statistical language models are applied, and surveys most commonly used language modeling techniques. Chapter 2 describes previous research on grammar-based language modeling.

My main contribution, a language modeling method based on probabilistic left corner parsing, is described in chapters 3 and 4. The theoretical aspects of the language model are discussed in Chapter 3, while Chapter 4 summarizes experimental results with the language model in a speech recognition task.

The last chapter summarizes the thesis and discusses future possibilities and extensions of the left corner parsing language model.

CHAPTER 1

A background of statistical language modeling

This chapter introduces the reader to the field of statistical language modeling. The general problem of language modeling is outlined and it is explained where and how statistical language models can be applied. A survey of common statistical language modeling techniques is given in order to situate the work on the probabilistic left-corner parsing language model within the current body of research into language modeling and as a quick reference for the following chapters.

1.1. What is a statistical language model?

In a general sense, a **language model** is a computational model that generates language. This thesis deals with human natural language: the language models discussed here attempt to mimic the natural language generation behavior of humans.

1.1.1. Representing language

In most current natural language processing (NLP) applications, natural language is represented as a stream of utterances while each utterance is represented as a finite sequence of words. Then, depending on the detail of representation, each word can again be represented as a finite sequence of sub-word units or it can be regarded as an atomic unit. In this thesis, words are regarded as atomic units, since it is common practice in large-vocabulary continuous speech recognition (LVCSR), which is the most important testbench for language models.

The following notations and definitions are introduced. **Sequences** of variables and/or constants are written separated by a comma and enclosed in parentheses, e.g. (x, y, z),

or concatenated, like *xyz*. Greek lowercase letters $\alpha, \beta, \ldots$, denote sequences; (Y, β) and $Y\beta$ denote the same sequence starting with Y. The empty sequence $()$ is denoted by ε. The expression w_i^j is a short notation for the sequence $(w_i, w_{i+1}, \ldots, w_j)$, or for ε if $j < i$. An initial substring of a sequence is called a **prefix**. The set of sequences of length n containing elements from the same set V is denoted by V^n. The union $\{\varepsilon\} \cup V \cup V^2 \cup \ldots$ is denoted by V^*.

The atomic units of language are **word types**. A **vocabulary** V is a set of word types, including the sentence boundary markers `<s>` and `</s>`. A **word token** is an instantiation of a word type (e.g. appearing in a text). A **word string** W is a finite sequence of word tokens. A **sentence** $S = w_0^N$ is a word string where $w_0 =$ `<s>`, $w_N =$ `</s>` and $w_i \notin \{$`<s>`, `</s>`$\}$ for $0 < i < N$. A **phrase** is a substring of a sentence. A **sentence prefix** is an initial substring of a sentence.

The process of mapping a text to a sequence of word tokens is called **tokenization**. The natural language phrase 'I'm here' could be tokenized as [`i'm`, `here`], or as (`i`, `am`, `here`), or as (`I`, `apostrophe`, `m`, `here`), ..., depending on the selected tokenization scheme. The mapping function is usually many-to-one: the representation discards the information that is deemed irrelevant for the application that will deploy the language model. For instance, if a language model should be case-insensitive (as dictated by the application specifications), then tokenization involves casting all characters to lower- or uppercase. Tokens do not necessarily correspond with orthographic words. It all depends on which information is relevant for the model or for the application.

1.1.2. The probability of a sentence

Thus far I have remained vague about the term 'language model' itself. The non-probabilistic approach to formal language theory defines language as a set of sentences (see, for instance, [Hopcroft and Ullman, 1979]). That is, a language is determined by an indicator function that tells whether a certain sentence is in a language or not. If this indicator function is adequately described by a **grammar**, which is a finite set of rules, then that grammar is said to **cover** the given language; a covering grammar is an adequate language model in this setting. If the grammar defines a strict superset of the language, then it is said to **overgenerate**. Conversely, if it defines a strict subset of the language, then it is said to **undergenerate**.

The non-probabilistic treatment of language, advocated by knowledge-based computational linguistics, leads to a distinction between the concepts of *grammaticality* and *acceptability*. A non-grammatical sentence can be acceptable, and a grammatical sentence can be non-acceptable.

The goals of language modeling, however, are closer to *quantifying acceptability*, rather than recognizing grammaticality. That does not mean that all knowledge is replaced by numbers, but that knowledge is formally applied up to the point that it is feasible, practical and insightful, and that scores, induced from actual text, can bridge

the remaining gap to a useful and working application. The probabilistic approach is a most elegant and mathematically sound implementation of this pragmatic view:

Definition 1 *A **probabilistic**, **stochastic** or **statistical language model** (LM) for a given language L is a probability mass function (pmf) $p(S)$ in the space of sentences, returning the estimated probability that the sentence S occurs in L.*

This definition is not a straightforward extension of non-probabilistic grammars, but can be motivated in the following way. In the non-probabilistic setting, given a language L and a sentence S, only two propositions are possible: $S \in L$ or $S \notin L$. In a probabilistic setting, this concept can be generalized by replacing these propositions with $P(L|S)$, the probability that it was L that generated a given S. The proposition $S \notin L$ corresponds with $P(L|S) = 0$, and $S \in L$ corresponds with $P(L|S) > 0$.

It is possible to define a probabilistic language model for L as the ensemble of probabilities $P(L|S)$, one for each S. But as it turns out it is often more convenient to think of L as given, so that $P(L) = 1$ and instead the single pmf $P(S|L) = P(S)$ is to be considered the language model, as expressed by Def. 1.

Note that it is implicitly assumed that a static probability distribution over sentences exists. However, as already mentioned in the introduction, research on large text corpora shows that natural language is inherently heterogeneous, even in very narrowly defined discourse contexts. Building one generally applicable language model means making it conditional on virtually all knowledge that a human listener would possess in the same situation. That is clearly unfeasible, although practical natural language systems can be built by sufficiently constraining the task and building specific application-dependent language models. A number of rather crude adaptation techniques have been developed to cope with the variability of the discourse within a certain application, for instance by detecting the topic in a broadcast news transcription task, or by predicting the type of the next turn in a dialogue system.

The current challenge of language modeling research is to make language models less brittle, (a) by a more efficient use of sentence-level and discourse-level information, and (b) by refining model adaptation techniques. For the latter it is necessary to separate the invariant properties of language from the variable part. My thesis was inspired by both (a) and (b): the working hypothesis was that by introducing syntactic knowledge, more efficient use can be made of all sentence-level information (instead of only of a short context window, as the *n*-gram model does), and that syntax is a viable candidate for being a language invariant.

Henceforth the adjective ‘statistical’ for language models is implicit in this text, as well as the application context which is characterized by L.

1.1.3. The probability of the next word

In some situations, for instance in statistical speech recognition, the probability of a whole sentence $S = w_0^N$ is factored using the chain rule:

$$P(S) = \prod_{i=1}^{N} P(w_i | w_0^{i-1}). \tag{1.1}$$

The product starts from $i = 1$ since always $P(w_0) = P(\texttt{<s>}) = 1$. The next-word predictors $P(w_i|w_0^{i-1})$ allow to compute the sentence prefix probability $P(w_0^i)$ as a simple update of $P(w_0^{i-1})$. This is important because it leads to search techniques that limit the space of sentences by pruning relatively improbable sentence prefixes in an early stage.
The term 'language model' is often used for the set of next-word predictors instead of the probabilities of the complete sentences. Confusion will be avoided by reserving the term 'language model' for probability estimators of whole sentences, while the term 'conditional language model' refers to sets of next-word predictors.

Definition 2 *Given a vocabulary V, a **conditional language model (CLM)** is a set of conditional pmfs $p(w_{i+1}|w_0^i)$ where $w_{i+1} \in V$, $w_0 =$ <s> and $w_1^i \in V^i$, returning an estimate of the probability that a sentence prefix w_0^i is immediately followed by the word w_{i+1}.*

In this dissertation it is assumed that V has a finite size denoted by $|V|$. The sum of CLM probabilities over V should add up to 1:

Property 1 *A CLM over V satisfies*

$$\sum_{w \in V} p(w|w_0^i) = 1$$

for every member pmf $p(\cdot|w_0^i)$ of the CLM.

1.2. Application of statistical language models

Several statistical methods in speech and language applications make use of statistical language models. Language modeling techniques are often applicable in non-language related areas as well, such as the modeling of DNA (deoxyribonucleic acid) strings. In fact, any field in which a probability distribution is needed over a discrete, categorical and high-dimensional space may benefit from the research on statistical language models.
The canonical application of statistical language models is *statistical automatic speech recognition* (ASR). I will now discuss the role of the language model in an ASR system and briefly note its use in other NLP domains.

ASR aims at transcribing an acoustic waveform A (a spoken utterance) by computer to its orthographic form W (a sentence). In a probabilistic formulation of the speech recognition problem, a search procedure selects $\hat{W}$, the W that is most likely to be the correct transcription of A:

$$\hat{W} = \arg\max_W P(W|A). \tag{1.2}$$

Bayesian risk minimization theory proves that (1.2) minimizes the expected sentence error rate (SER).[1] Now (1.2) can be rewritten as

$$\hat{W} = \arg\max_W P(W|A) = \arg\max_W \frac{f(A|W)P(W)}{f(A)} = \arg\max_W f(A|W)P(W). \tag{1.3}$$

The optimization target consists of two factors. The probability density function (pdf) $f(A|W)$ is the acoustic model (commonly, the acoustic space is continuous, hence the notation $f(A|W)$ instead of $P(A|W)$). The second factor $P(W)$ is estimated by the language model. In an HMM(hidden Markov model)-based Viterbi decoder, currently the most common architecture in speech recognition, the acoustic model is a concatenation of phone HMMs, each returning the likelihood of a segment of A given a particular phone. With the help of a pronunciation lexicon the Viterbi procedure rewrites W as the phone sequence that maximizes $f(A|W)$.[2]
In a typical configuration a simple and fast CLM is applied in a first decoding pass. The first pass outputs a word lattice, an efficient representation of a limited number of sufficiently probable sentence hypotheses. In a second pass, a more accurate but slower CLM can be used to re-score the word lattice. If a simple list of n most probable hypotheses is output instead of a word lattice, then any whole-sentence LM can be used in the second pass.

Besides ASR, statistical language models are also needed in other (statistical) natural language applications. Here are a few examples.

- **Document classification** assigns a document class $\hat{c}$ to an input document d. The Bayesian approach would select the class that is most probable given d:

  $$\hat{c} = \arg\max_c P(c|d) = \arg\max_c P(d|c)P(c). \tag{1.4}$$

  The *class-dependent* language models $P(d|c)$ can be trained on a labelled set of documents; the labels can be assigned by hand or obtained through an automatic clustering procedure.

1. For the SER, there are only correct and incorrect sentences. A sentence with two word errors counts as much as a sentence with only one word error. The accuracy of a speech recognition system is, however, mostly measured by word error rate (WER). There are some recent papers on the explicit minimization of the WER with small gains of word accuracy, but this always implies a considerable increase of system complexity. Most speech recognizers take (1.2) as the optimization target and rely on a sufficient correlation between the SER and the WER.
2. Experience shows that summing over all phoneme sequences for a given W, as actually should be done, does not improve the recognition accuracy with respect to the Viterbi (maximizing) approximation.

- **Spelling correction** can be considered as finding the correct orthography $\hat{w}$ of a sentence given the possibly erroneous user input $\tilde{w}$:

$$\hat{w} = \arg\max_w P(w|\tilde{w}) = \arg\max_w P(\tilde{w}|w)P(w). \tag{1.5}$$

  The language model $P(w)$ is trained on correct text, while constraints for the 'human transcriber' model $P(\tilde{w}|w)$ can be obtained from experiments with human subjects.

- **Machine translation** [Brown et al., 1993] is concerned with finding the best translation $\hat{e}$ of a foreign language sentence f :

$$\hat{e} = \arg\max_e P(e|f) = \arg\max_e P(f|e)P(e). \tag{1.6}$$

  The target language model $P(e)$ causes a preference for translations that are well-formed and sound right in the target language.

1.3. Estimating CLMs from observed data

The usual way of estimating LMs and CLMs is to assume a certain parameterized model class and select one member from it by estimating the model parameters from the training sequence.
The training sequence O is a plain text corpus, just a sequence of sentences. It can be represented as a sequence of independent events $x = (h, w)$, where w is a word and h the sequence of words preceding w in the same sentence. Let θ denote the model parameters and $\{q_\theta(w|h)\}_\theta$ the CLM class from which one CLM $q_{\hat{\theta}}(w|h)$ is selected. The estimator Θ is a deterministic function of the training corpus O: $\Theta(O) = \hat{\theta}$. $\hat{\theta}$ is a random variable due to the randomness of O.
Most CLM estimators are based on estimators for the joint pmf $q_\theta(h, w)$ instead of the conditional pmfs $q_\theta(w|h)$. The conditional pmf follows from the joint pmf through normalization:

$$q_\theta(w|h) = \frac{q_\theta(h, w)}{\sum_{v \in V} q_\theta(h, v)}. \tag{1.7}$$

1.3.1. Measuring language model performance

Perplexity

Language model performance can only be measured as the perceived performance of the application in which the language model is applied, e.g. the *average word error rate* in speech recognition.
However, it is useful to have an application-independent measure of model goodness in various circumstances. For instance, when deriving explicit formulas for the estimation of the model parameters; or in the course of an iterative estimation algorithm;

or when comparing the strengths of one model with those of another. For this purpose, *cross-entropy* or *perplexity* is used.
One could, for instance, try to **minimize** the **mean square error** (MMSE) between the estimated set of parameters $\hat{\theta}$ and the optimal set of parameters θ^*:

$$\Theta_{\text{MMSE}} = \arg\min_{\Theta} E[|\theta^* - \Theta(O)|^2]. \tag{1.8}$$

Of course, θ^* is unknown, but one assumes a certain prior distribution $P(\theta^*)$.
The MMSE criterion assigns equal importance to every component of θ. However, it is better to relax the optimization conditions on the parameters that have a small influence in favor of the parameters that have a large influence. The **divergence** or **Kullback-Leibler distance** [Kullback and Leibler, 1951, Kullback, 1959] meets this need and is commonly used for language models. The Kullback-Leibler (KL) distance of the estimated CLM $q_\theta(w|h)$ from the true $P(w|h)$ is written and defined as

$$\begin{aligned} D(P\|q_\theta) &\doteq \sum_{w,h} P(w,h)\log\frac{P(w|h)}{q_\theta(w|h)} \\ &= E[-\log q_\theta(w|h)] - E[-\log P(w|h)] \\ &= H_{q_\theta}[w|h] - H[w|h], \end{aligned} \tag{1.9}$$

where $H_{q_\theta}[w|h]$ is the cross-entropy of w given h according to q_θ, and $H[w|h]$ is the true entropy of w given h. $H[w|h]$ is a fixed term, while $H_{q_\theta}[w|h]$ is bounded from below by $H[w|h]$; it only reaches equality if q_θ equals P. Hence, the KL estimator minimizes the cross-entropy:

$$\Theta_{\text{KL}}(O) = \arg\min_{\theta} H_{q_\theta}[w|h]. \tag{1.10}$$

Most publications on language modeling, however, report the **test set perplexity**, proposed by Jelinek et al. [1983], instead of the cross-entropy:

$$\text{PPL}_{\text{test}} \doteq \exp(-\hat{H}_q[w|h]). \tag{1.11}$$

In this definition $\hat{H}_q[w|h]$ is an empirical average as an estimate of $H_q[w|h]$:

$$\hat{H}_q[w|h] = -\frac{\sum_{w,h} c_{O_{\text{test}}}(w,h)\log q(w|h)}{\sum_{w,h} c_{O_{\text{test}}}(w,h)}. \tag{1.12}$$

where $c_{O_{\text{test}}}(w,h)$ denotes the occurrence frequency of (w,h) in the test corpus O_{test}.
Perplexity was introduced as a more intuitive measure than cross-entropy: imagine a CLM that assigns an equal probability to m words on the (geometric) average and zero probability to the other words in the vocabulary. This model has an expected test set perplexity of exactly m. When this CLM is used in speech decoding, the acoustic model has to distinguish between m equally likely branches on the average. So the

perplexity of the CLM is a direct measure of the difficulty of the disambiguation task that remains for the acoustic model in a speech decoding task.
Note that minimizing the test set perplexity is equivalent with maximizing the likelihood of the test set, and that minimizing KL distance is equivalent with minimizing the *expected* test set perplexity of a random and sufficiently large test set.
Maximizing the likelihood of the *training* set, on the other hand, is often computationally more feasible. At the downside, unconstrained maximum-likelihood estimation leads to *overtraining*, as will be shown in Sec. 1.3.2.

Word error rate

In speech recognition, the **word error rate** (WER) is a customary measure to evaluate the accuracy of the whole recognition system. The set of recognized sentences (the automatic transcription) is aligned with the set of reference sentences (the reference transcription), and then the WER is defined as

$$\mathrm{WER} \doteq \frac{I+D+S}{N_{\mathrm{ref}}}, \tag{1.13}$$

where N_{ref} is the total number of tokens in the reference set, I is the total number of insertion errors, D is the total number of deletion errors, and S is the total number of substitution errors. For each sentence, the alignment that minimizes the total number of insertion, deletion and substitution errors is selected. For instance, if the reference sentence is `have a good day`, and the recognized sentence is `have a good die`, then only 1 word error (the substitution of `day` by `die`) is counted, instead of 2 word errors (the deletion of `day` and the insertion of `die`).

1.3.2. Maximum-likelihood estimation and the need for smoothing

The maximum-likelihood (ML) estimator selects the model that maximizes the likelihood of the training corpus O. It is commonly known that the conditional probabilities, then, are simply relative frequencies:

$$p_{\mathrm{ML}}(w|h) \quad = \quad \frac{c_O(\Phi(w,h)}{\sum_{v\in V} c_O(\Phi(v,h))}, \tag{1.14}$$

where $c_O(e)$ denotes the observation frequency of an event e in O, and Φ is a mapping used to pool observations into equivalence classes. Why pooling observations is necessary, is discussed in Sec. 1.4; it is, of course, clear that no pooling at all would result in most observation frequencies to equal zero.
The ML estimator is only *asymptotically* unbiased. Unfortunately any practical training corpus size is far from the point where the ML estimator reaches saturation. For instance, the 3-gram CLM (cf. Sec. 1.4.1), with a typical vocabulary size $|V| = 20$k, counts about 8×10^{12} free parameters. A huge training corpus of 10^9 word tokens contains at most 10^9 different trigrams, which means that at least 99.9875% of the

trigrams nowhere occur in the training corpus. All the trigrams that did not appear in the training corpus will be assigned a zero probability, while theory, intuition and experience tell us that it is very likely that a significant portion of trigrams in a test corpus was never seen in the training corpus, however large the latter may be. While the probability of zero frequency events are underestimated, the probability mass of the events that did occur in the training corpus is overestimated.

Many of the research papers on language modeling actually were devoted to solving this general problem: assigning reasonable non-zero probabilities to unobserved events. The result is a number of smoothing techniques, which in most cases are more generally applicable than for language modeling only. The theoretic foundation is often rather weak: the question of language modeling is essentially the estimation of probability distributions over *discrete and categorical* spaces, where concepts such as distance and neighborhood are difficult to formulate.
Therefore, smoothing techniques are primarily judged on their observed performance in experiments. In my experience, however, there is no clear winner, although it is possible to formulate a few general guidelines, as given at the end of this chapter. The performance of a smoothing technique depends on factors such as the training corpus size, the vocabulary size, and the discourse style of the task. Even given a specific task, the great number of smoothing techniques and all of their possible combinations makes an exhaustive search for the best combination of smoothing techniques impossible. The development of a language model for a new task is an incremental optimization process with an unknown global optimum.

There are three types of smoothing: clustering, discounting of observed frequencies, and model combination. The next section discusses the most important simple language model classes from the perspective of their characteristic clustering function. Discounting and model combination are discussed in the sections thereafter.

1.4. Simple language model classes

Statistical models are estimated from observation data. In the current discussion, that means that LMs are estimated from tokenized text, the training corpus. LMs and CLMs are defined over an enormous, in fact infinite, discrete space.
It became clear from the previous section that direct estimation of CLM probabilities $P(w_{i+1}|w_0^i = h)$ from observed data is not reliable, due to data sparseness.
The model class and smoothing technique can often be characterized by a **clustering** function Φ that maps joint events $x = (h, w)$ of the immediate occurrence of a word w after the sentence prefix h to a far smaller number of **equivalence classes** $\tilde{x} = (\tilde{h}, \tilde{w})$.
The idea behind Φ is:

1. predicting w from $\tilde{x}$ is easier than from h;
2. $\tilde{h}$ contains all the essential information from h needed to predict $\tilde{w}$;

3. $\tilde{x}$ is observed more often than x, enabling a more reliable estimation of the CLM probabilities.

In general, if Φ is a soft clustering, i.e. a stochastic function with possible outcomes $\tilde{x}$, then $P(w|h)$ and $P(h,w)$ can be factored as

$$P(h,w) = \sum_{\tilde{x}} P(x|\tilde{x})P(\tilde{x}) \tag{1.15}$$

$$P(w|h) = \sum_{\tilde{x}} P(w|\tilde{x})P(\tilde{x}|h) \tag{1.16}$$

over the *hidden* variable $\tilde{x}$, where it is assumed that $\tilde{x}$ provides all essential information from h to predict w. Hidden variables are generally a difficult matter in statistical learning, since they are not observed by definition. In some cases, however, there are efficient estimation algorithms such as the expectation-maximization (EM) algorithm.

Now follows a survey of the most common simple language model classes from the perspective of the clustering function.

1.4.1. Word-based n-grams

It is reasonable to assume that the occurrence of a word is mostly not very much influenced by words that are sufficiently distant. This intuition translates to the choice

$$\Phi(w_0^{i-1}, w_i) = (w_{i-n+1}^{i-1}, w_i). \tag{1.17}$$

The result is an n-gram CLM. For example, a trigram CLM ($n = 3$) approximates $P(w_{i+1}|w_0^i)$ as $P(w_{i+1}|w_{i-1}^i)$.

Note: In order to improve the performance over storage ratio, it is possible to reduce n selectively, resulting into a **variable-length n-gram** or **varigram**. If a certain criterion judges that a context w_{i-n+1}^{i-1} is not sufficiently more informative for any choice of w_{i-n+1}, then only the pmf $P(w_i|w_{i-n+2}^{i-1})$ will be estimated. For example, see [Niesler and Woodland, 1996, Seymore and Rosenfeld, 1996, Van Uytsel and Van Compernolle, 1998, Stolcke, 1998].

Word-based n-grams are the main workhorse of most statistical language models; they are often complemented with other models, but the better part of the language model performance is to be attributed to them. They are very data-hungry, however. For instance, the estimation of a trigram typically needs a training corpus of 10 million tokens; a 4-gram becomes typically only effective if it is estimated on 100 million tokens. Its limited scope precludes any form of generalization. As a consequence, word-based n-gram models are very domain-specific.

1.4.2. Category-based *n*-grams

The central idea of class-based *n*-grams is that words belong to one or more **word categories**. Each word w_i is assumed to be emitted by a word category c_i.
The word categories can either be determined with data-driven clustering algorithms, e.g. [Bahl et al., 1989, Brown et al., 1992, Kneser and Ney, 1993, Ueberla, 1996b,a] or on a linguistic basis: parts-of-speech (e.g. [Niesler and Woodland, 1996, Geutner, 1997]), function/content words (e.g. [Geutner, 1996]), and semantic groupings (e.g. [Kneser and Peters, 1997]).

Here $\Phi(w,h)$ is a function that maps words to categories. There are two kinds: $\Phi(w,h)$ can either be a deterministic or a probabilistic function. $\Phi(w,h)$ is obviously deterministic if it is assumed that each word belongs to exactly one category and each token of (w,h) is mapped to its category; but it is as well deterministic if words can belong to more than one category, but the one selected by Φ is unambiguously determined by w and h, for instance with a decision tree (e.g. [Heeman and Allen, 1997]) or where the mapping is position dependent (e.g. [Pastor et al., 1998]).

If Φ is deterministic, then let c_i denote the mapping of w_i under Φ. The CLM probabilities can be correspondingly written as

$$\begin{aligned} p_{\text{catng}}(w_i|w_0^{i-1}) &= P(w_i, c_i|w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1}) \\ &= P(w_i|w_{i-n+1}^{i-1}, c_{i-n+1}^{i}) P(c_i|w_{i-n+1}^{i-1}, c_{i-n+1}^{i-1}). \end{aligned}$$

The number of parameters can be reduced by assuming independence of w_i resp. c_i of certain conditioning events. For instance, as in [Brown et al., 1992]:

$$p_{\text{catng}}(w_i|w_0^{i-1}) = p_1(w_i|c_i) p_2(c_i|c_{i-n+1}^{i-1}). \tag{1.18}$$

There are, of course, many other parameterizations possible, trading off model accuracy against model reliability given a fixed training set size.
If Φ is stochastic, then c_i has to be treated as a hidden variable. Thus, for a category-based *n*-gram,

$$p_{\text{catng}}(w_i|w_0^{i-1}) = \sum_{c_{i-n+1}^{i}} P(w_i|c_{i-n+1}^{i}, w_{i-n+1}^{i-1}) P(c_{i-n+1}^{i}|w_{i-n+1}^{i-1}), \tag{1.19}$$

where the sum ranges over all category sequences that can possibly underly w_{i-n+1}^{i-1} and w_i. This sum can be computed with dynamic programming using forward probabilities (for instance, see [Geutner, 1997]). However, in a common approximation (for instance, see [Derouault and Mérialdo, 1986]), known as the Viterbi approach [Viterbi, 1967], only one category sequence is tracked, namely the one explaining w_1^i best:

$$p_{\text{catng}}(w_i|w_0^{i-1}) \simeq \sum_{c_i} P(w_i|C_{i-n+1}^{i-1}, c_i, w_{i-n+1}^{i-1}) P(C_{i-n+1}^{i-1}, c_i|w_{i-n+1}^{i-1}), \tag{1.20}$$

where

$$C_i = \arg\max_{c_i} P(w_i|C_{i-n+1}^{i-1}, c_i, w_{i-n+1}^{i-1}) P(C_{i-n+1}^{i-1}, c_i|w_{i-n+1}^{i-1}). \tag{1.21}$$

In the original Viterbi decoding, the summation in (1.20) is replaced by a maximization. This requires less computations, but also reduces the smoothing effect.
Again, depending on the availability of training data, proper parameterizations must be found for $P(w_i|C_{i-n+1}^{i-1}, c_i, w_{i-n+1}^{i-1})$ and $P(C_{i-n+1}^{i-1}, c_i|w_{i-n+1}^{i-1})$. There are no principled approaches to this problem for language modeling; so far it has remained a matter of craftsmanship.

Class-based *n*-grams require less data than word-based *n*-grams for sufficient training. As a stand-alone model, they perform much worse than word-based *n*-grams. They are typically used to complement word-based *n*-grams in order to enlarge the modeling scope. The number of classes can be scaled in proportion with the training corpus size. Automatic clustering generally leads to better performance within the application, but linguistic clustering makes the model less application-specific.

1.4.3. Delayed *n*-grams

Delayed *n*-grams, also called **skip** or **distant** *n*-grams, omit one or more of the most recent words from the context:

$$\Phi(w_1^{i-1}, w_i) = (w_{i-n+1-s}^{i-1-s}, w_i), \tag{1.22}$$

where s is the number of positions skipped. The reason for skipping is to capture dependencies from a larger part of the context than plain *n*-grams do, while avoiding too much data fragmentation.
Delayed *n*-grams are used in combination with another *n*-gram as an approximation of an $(s+n)$-gram, where the directly estimated $(s+n)$-gram would be ineffective due to data sparseness.

1.4.4. Cache-based models and triggers

In this paragraph, a **cache** is a moving window over the test text (which extends beyond the current sentence). Let $T = w_0^N$ denote the test text, where $N+1 = |T|$. A cache-based language model estimates $P(w_i|w_0^{i-1})$ from a cache $w_{\max(0,i-s_i)}^{i-1}$ where the cache length s_i in general may vary, but is often chosen constant (typically 1000... 3000). That is, $w_{\max(0,i-s_i)}^{i-1}$ is treated as the training corpus by the cache-based model. Cache-based word-based uni- and bigrams are rather common components in state-of-the-art language models.
The term 'cache' in language modeling is due to [Kuhn and De Mori, 1990], who actually applied the idea in a more sophisticated way: a POS(parts-of-speech)-based CLM in the style of (1.20) was proposed of which the $P(w_i|c_i)$ component was estimated

as a linear interpolation of an ML model and a cache-based model. The cache-based model maintained separate caches of 200 entries for each of 19 POSs.

Trigger-based language models are a refinement of cache-based models. A trigger pair is a pair of words that have a high mutual information or significantly contribute to another optimization criterion [Tillmann and Ney, 1996]. For instance, the pair (*bank*, *loan*) is likely to be a trigger pair. The trigger-based language model may be soundly implemented as a maximum-entropy model employing trigger pairs as features (cf. Sec. 1.6.4 and [Lau et al., 1993, Rosenfeld, 1994]). It is observed that the most efficient triggers are self-triggers, that is, consisting of two identical words; a trigger-based language model containing only self-triggers is very similar to a unigram cache-based model. This may explain why trigger-based models are not much better than cache-based models.

There are two rationales behind cache-based language modeling and triggers. First, although most information on w_i is on average contained within a short window (as exploited by n-gram CLMs), the rest of the information is still considerable, but spread over a long history. Cache-based models are able to capture dependencies from a long history without augmenting data sparsity.
Second, in any text fragment one can easily spot several idiosyncracies specific for the author, the situation, the topic etc. In particular, it was measured that the probability of a word rises if it has occurred in the recent past. Humans tend to prefer to understand a word if they have heard it recently before, a mechanism that is called 'priming' in psycholinguistics. The unigram cache simulates that behavior — admittedly in a primitive way.

Given the simplicity of unigram cache-based models and the substantial performance gain, they have become an essential component of language models in automatic speech recognition systems. Trigger-based language models are more difficult to implement, and not commercially viable.

1.4.5. Grammar-based language models

Chapter 2 is devoted to grammar-based language models and therefore the exposition here is kept concise.

Grammar-based language models predict a grammatical structure as a hidden variable. If the mapping of a sentence prefix to its grammatical structure were unambiguous, then the clustering function could be written

$$\Phi(w_1^{i-1}, w_i) = ((w_1^{i-1}, t(w_1^{i-1})), w_i). \tag{1.23}$$

The grammatical structure $t(w_1^{i-1})$ can be thought of as a partial parse tree, but in general its form is determined by the chosen (statistical) grammatical framework.

However, in most cases the grammatical structure is ambiguous. Therefore, the CLM probabilities must be computed as a *sum* of probabilities with as many terms as there are non-improbable grammatical structures:

$$P(w_i|w_1^{i-1}) = \sum_{t(w_1^{i-1})} P(w_i|w_1^{i-1}, t(w_1^{i-1}))P(t(w_1^{i-1})|w_1^{i-1}). \tag{1.24}$$

Statistical grammar-based language models are fairly complex, but in combination with other language models, they can boost the performance considerably, as this thesis will show. Further research is needed to relax their space and time complexity, before they can make it into commercial products. A possible disadvantage is their need for a syntactically preprocessed text corpus — but my experiments showed that a good grammar-based language model can still be obtained with an automatically preprocessed corpus.

1.5. Discounting

ML estimation of $P(w|h)$ with a clustering function is only the first step in smoothing. The estimates are further improved by a number of techniques that attempt to compensate the bias of the ML estimate. These techniques can often be described as *discounting* the observation frequencies and then return relative frequencies based on the discounted frequencies.

1.5.1. Laplace's law of succession

Laplace's solution to the problem of data sparsity is to add 1 to every event frequency:

$$q_L(x) = \frac{c_O(x)+1}{|O|+|X|}, \tag{1.25}$$

where O is an observation sequence of events (the training corpus) and X is the set of all distinct events x. (1.25) is also known as Laplace's *law of succession*, or *add-1 smoothing*. This simple formula is actually the Bayesian MMSE estimator where the prior distribution of x is assumed uniform.
The conditional form of (1.25),

$$q_L(w|h) = \frac{c_O(w,h)+1}{\sum_{v\in V} c_O(v,h)+|V|}, \tag{1.26}$$

is obtained using (1.7). Word types not in V are usually mapped to a single word token *<unk>* denoting 'the unknown word'.

Experience shows that Laplace's law is suitable for smoothing unigrams, but less for smoothing $(n>1)$-gram CLMs. The reason is that for estimating an n-gram CLM, the $(n-1)$-gram CLM is probably a better prior distribution than the uniform distribution.

1.5.2. Jelinek-Mercer smoothing (deleted or held-out estimation)

The training corpus O is split into two parts: a development corpus D and a held-out corpus H. H is preferrably drawn from O in a totally random way since it is supposed to be generated by the same source as D. The proportion $|H|/|O|$ is chosen rather small (typically 5 to 10%), because much less parameters are estimated from H than from D.
A pmf of the form

$$q_{\mathrm{JM}}(x) = \begin{cases} q_k & \text{if } c_D(x) = k \text{ and } k = 0,1,\ldots,K \\ \alpha \frac{c_D(x)}{|D|} & \text{if } c_D(x) > K \end{cases} \tag{1.27}$$

was proposed by Jelinek et al. [1983]. K is a design parameter that separates out the events with 'too low' frequencies. The idea is that if $c_D(x)$ is high enough, then the relative frequency estimator is probably not too bad; a constant discount factor α reserves some probability mass for the low frequency events. The probabilities q_k do not directly depend on relative frequencies. q_k and α are chosen such that the likelihood of H is maximized. The solution is analytically found as [Jelinek, 1997, pp. 258–263]

$$q_k = \frac{1}{n_k} \frac{r_k}{|H|} \tag{1.28}$$

$$\alpha = \frac{|D|}{\sum_{j>K} j n_j} \frac{\sum_{j>K} r_j}{|H|} \tag{1.29}$$

where n_k is the number of distinct events that occurred k times in D, and r_k is the frequency in H of events that occurred k times in D:

$$n_k \doteq \sum_x \delta(c_D(x) = k) \tag{1.30}$$

$$r_k \doteq \sum_x \delta(c_D(x) = k) c_H(x), \tag{1.31}$$

and $\delta(e)$ equals 1 if e is true, and 0 otherwise. Note that an event that was not observed in the training corpus gets a probability $r_0/(n_0|H|)$ which is greater than zero in normal circumstances.

This estimator has a class-based interpretation. Each event x belongs to a class that is solely defined by the frequency of its members in the development corpus D: let Φ_k be the class of the events that occur k times in D (hence $|\Phi_k| = n_k$) for $k \leq K$ and let Φ_{K+1} be the class of the events that occur more than K times in D. Then $P(x)$ is written as $P(x|\Phi(x))P(\Phi(x))$. The class probability $P(\Phi(x))$ is the relative frequency of $\Phi(x)$ in the held-out corpus:

$$P(\Phi_k) = \begin{cases} r_k/|H| & \text{for } k \leq K \\ \sum_{j>K} r_j/|H| & \text{for } k = K+1 \end{cases} \tag{1.32}$$

while the member probabilities $P(x|\Phi_k)$ are relative frequencies in D:

$$P(x|\Phi(x)=\Phi_k)=\begin{cases}1/n_k & \text{for } k\leq K,\\ c_D(x)/\sum_{j>K} j\cdot n_j & \text{for } k=K+1.\end{cases} \tag{1.33}$$

In the high-frequency classes, the member probabilities are relative frequencies extracted from D. In the low-frequency classes the probability mass is divided uniformly over its members in absence of any prior knowledge. If another, simpler distribution is known, then it is better to replace the uniform distribution with a scaled version of that other distribution.

Deleted estimation tends to underestimate the probabilities of the more frequent events, probably due to the constant discount factor α. Given a large training corpus, it is inferior to Good-Turing discounting (discussed next). On smaller corpora, however, it is more reliable than Good-Turing. Another strong point is that it can be generalized to handle fractional counts, which occur, for instance, when weighting observation frequencies, or as frequency expectations in reestimation procedures. Deleted estimation is still heavily used in computational linguistics, presumably because training corpora tend to be smaller in this research domain.

1.5.3. Good-Turing and Katz discounting

Jelinek-Mercer smoothing requires an appropriate splitting of the training corpus into a development and a held-out corpus. Good-Turing (GT) smoothing is a more automatic and computationally efficient procedure. The GT smoothing of an event count k is

$$k_{\mathrm{GT}}=(k+1)\frac{n_{k+1}}{n_k}. \tag{1.34}$$

This result can be obtained by significantly different methods [Nádas, 1985],[Jelinek, 1997, pp. 263–265]. A a derivation from the class-based (Bayesian) view is given next.
As in the previous case, $x=(h,w)$ is classified according to its sample count in a development corpus D. In its original form, there is no bound K above which (a scaled version of) the relative frequency estimate is used. Hence an event x belongs to class Φ_k if $c_D(x)=k$. Here is Turing's estimator for the probability that an event x for which $\Phi(x)=\Phi_k$ is emitted (first published by Good [1953], where A.M. Turing is acknowledged):

$$P_{\mathrm{GT}}(\Phi_k)=\frac{(k+1)n_{k+1}}{|D|}, \tag{1.35}$$

where $n_{k+1}=|\Phi_{k+1}|$. This estimator was proven to have small bias for large $|D|$ by McAllester and Schapire [2000] and is meant for low frequency event classes, in

particular for the case $k = 0$: the estimated probability mass of *unseen* events is equal to the relative frequency of *singleton* events in the development corpus.
If no prior knowledge exists, the member probability may be estimated as $1/n_k$, resulting into the well-known formula

$$P_{\mathrm{GT}}(x) = \frac{k+1}{|D|}\frac{n_{k+1}}{n_k}. \tag{1.36}$$

In other words, for $c_D(x) > 0$, $P_{\mathrm{GT}}(x)$ is obtained as a relative frequency estimate $k/|D|$ times a **discount** factor

$$d_k = \frac{(k+1)n_{k+1}}{kn_k}, \tag{1.37}$$

which is easy to recall as the frequency of Φ_{k+1} events divided by the frequency of Φ_k events.

Example 1 *In the first 1M words of the WSJ corpus (non-verbalized punctuation with a closed 20k vocabulary), there are 8634 distinct unigrams. They where categorized according to their observed frequencies and the quantity kd_k was computed to see what a certain observed frequency is worth according to Good-Turing smoothing:*

k	0	1	2	3	4	5	...
n_k	11350	3353	1424	787	548	363	...
kd_k	0.295	0.849	1.658	2.785	3.312	4.612	...

Unseen unigrams get an effective count of .295, while the other counts are discounted with a certain amount between 0 and 1.
After extending the development corpus to the first 4M words, 19,460 distinct unigrams were seen, which is already close to the vocabulary size of 19,984. The situation is now completely different:

k	0	1	2	3	4	5
n_k	294	270	330	368	586	650
kd_k	0.918	2.444	3.345	6.369	5.546	6.055

Note that it is not required that $k \cdot n_k$ be a decreasing sequence in order for Good's method to be consistent. There is no data sparsity in this case, given the size of the training corpus. For low k, the discounted counts are larger than the original counts, as expected.

In the second case of the previous example, a unigram that occurred 3 times is assigned a higher probability than a unigram that occurred 4 times. This 'anomaly' is rather common in the tails of the n_k versus k histogram (in the above case, it is the left tail) and is due to the large variance of n_k. In this region, Good's method is not to be applied without smoothing the n_k themselves.

In the higher range of k (the right tail), Good-Turing discounting becomes too unreliable as well. This problem can be solved as in Jelinek-Mercer smoothing: by pooling the classes $\Phi_{K+1}, \Phi_{K+2}, \ldots$ in one class $\Phi_{>K}$ and distributing the member probabilities proportionally with the relative frequencies. Typically one chooses $K = 5 \ldots 10$. In addition, relative frequencies of events in $\Phi_{>K}$ are considered sufficiently reliable. The result is equivalent with replacing the discount factors $d_{K+1}, d_{K+2}, \ldots$ with one discount factor

$$d_{>K} = 1 - \frac{(K+1)n_{K+1}}{|D| - \sum_{j=1}^{K} jn_j} \tag{1.38}$$

applicable to $c_D(x) = k > K$.

A common and widely used alternative approach was authored by Katz [1987], who leaves out discounting of $\Phi_{>K}$ events altogether ($d'_{>K} = 1$), and adjusts the discount factors $d_1, \ldots, d_K$ such that:

1. the total discounted probability mass that is reserved for unseen events remains $n_1/|D|$: $\sum_{k=1}^{K} kn_k(1-d_k) = n_1$, and
2. the discounts are the original discounts times a constant factor: $1 - d'_k = \mu(1 - d_k)$ for $k = 1 \ldots K$,

which results in the discount factors

$$d'_k = \frac{\frac{(k+1)n_{k+1}}{kn_k} - \frac{Kn_K}{n_1}}{1 - \frac{Kn_K}{n_1}}. \tag{1.39}$$

Good-Turing and Katz discounting are particularly suited for smoothing small sample counts from large corpora, and they are probably most sound from the theoretical perspective. However, on smaller corpora deleted estimation or absolute discounting may be preferrable. On the other extreme, for smoothing probabilities of relatively frequent events, add-1 smoothing or absolute discounting is more reliable.

1.5.4. Nádas smoothing

GT smoothing is a non-parametric empirical Bayesian estimation method. Nádas [1984] proposed another empirical Bayesian estimation method where the event probabilities are assumed to be beta distributed.

Let $f(\theta_k)$ denote the prior density of the probability θ_k of an event that occurred k times in the development corpus. Assuming that $c_D(x)$ is binomially distributed, the

Bayesian estimator that is optimal for the MMSE criterion is

$$\begin{aligned}\hat{\theta}_k &= E[\theta_k|c_D(x)=k] \\ &= \int_0^1 \theta_k f(\theta_k|c_D(x)=k)\mathrm{d}\theta_k \\ &= \frac{\int_0^1 \theta_k P(c_D(x)=k|\theta_k) f(\theta_k)\mathrm{d}\theta_k}{\int_0^1 P(c_D(x)=k|\theta_k) f(\theta_k)\mathrm{d}\theta_k} \\ &= \frac{\int_0^1 \theta_k^{k+1}(1-\theta_k)^{|D|-k} f(\theta_k)\mathrm{d}\theta_k}{\int_0^1 \theta_k^{k}(1-\theta_k)^{|D|-k} f(\theta_k)\mathrm{d}\theta_k}.\end{aligned} \tag{1.40}$$

Good essentially uses a non-parametric estimator of $f(\theta_k)$, while Nádas assumes a beta distribution:

$$f(\theta_k) = \frac{1}{\mathrm{B}(\alpha,\beta)}\theta_k^{\alpha-1}(1-\theta_k)^{\beta-1}. \tag{1.41}$$

The α and β parameters are estimated from an artificial sample comprising maximum-likelihood (ML) estimates of θ_k using the method of moments.

This smoothing method yields model performance that is comparable with JM and GT smoothing, but is more complicated, which is probably the reason why it is hardly ever used. However, it may be interesting to enhance the method by recognizing that the distribution of the vector consisting of all $c_D(x)$ for all x is not a combination of binomial distributions, but actually a multinomial distribution.[3]

1.5.5. Absolute discounting

Experiments show that the true probability of n-gram events is often well approximated by subtracting a small quantity d between 0 and 1 from the original n-gram event count, and redistributing the gained probability mass over all, seen and unseen, events in proportion with a background distribution q_B [Ney et al., 1994]:

$$P_{\mathrm{ABS}}(x) = \frac{\max(c_D(x)-d,0)}{|D|} + d\frac{|X|-n_0}{|D|}q_B(x). \tag{1.42}$$

The resulting method is called **absolute discounting**. d can be estimated by optimizing the likelihood of a small held-out corpus, or analytically [Ney et al., 1994]. As a further refinement, one may define a number of bins over the event frequencies and estimate a separate d for each bin empirically.

It is hard to find a theoretical ground for absolute discounting, but in practice it is frequently used, since it is an easy, robust and reliable smoothing technique. It is preferred in situations where the GT estimator is not to be trusted, e.g. when the n_{k+1}/n_k statistics have a large variance, or on small corpora.

3. Nádas suggests a conjugate n-dimensional Dirichlet prior in this case, but does not elaborate on the matter any further.

1.6. Model combination

1.6.1. Katz back-off

Given two CLMs p_A and p_B, where p_A is induced from a training corpus D. The CLM that arises by **backing off** from p_A to p_B, denoted here by $\mathrm{KBO}(p_A, p_B)$, is defined by

$$\mathrm{KBO}(p_A, p_B)(w|h) \doteq \begin{cases} p_A(w|h) & \text{if } c_D(w,h) > K \\ \lambda(h)\, p_B(w|h) & \text{if } c_D(w,h) \leq K \end{cases} \tag{1.43}$$

where K is a **cut-off** parameter and $\lambda(h)$ is a normalization factor that can be precomputed for each h:

$$\begin{aligned} \lambda(h) &= \frac{1 - \sum_{w \in V} \delta(c_D(w,h) > K)\, p_A(w|h)}{\sum_{w \in V} \delta(c_D(w,h) \leq K)\, p_B(w|h)} \\ &= \frac{1 - \sum_{w \in V} \delta(c_D(w,h) > K)\, p_A(w|h)}{1 - \sum_{w \in V} \delta(c_D(w,h) > K)\, p_B(w|h)}. \end{aligned} \tag{1.44}$$

The rationale of the back-off technique is that $p_A(w|h)$ estimates are considered sufficiently reliable if the event (w,h) was seen more than K times, while in the other case the $p_A(w|h)$ are replaced by scaled $p_B(w|h)$ estimates. p_B is supposed to be a less sophisticated, but more robust model.
As an example, the Katz back-off n-gram [Katz, 1987] is a cascade of smoothed n, $n-1, \ldots,$ 1-gram CLMs combined with back-off:

$$p_{\text{Katz-}n\text{g}} = \begin{cases} \mathrm{KBO}(p_{n\text{g}}, p_{\text{Katz-}(n-1)\text{g}}) & \text{if } n > 1 \\ 1/|V| & \text{if } n = 0. \end{cases} \tag{1.45}$$

In his frequently cited paper [Katz, 1987], Katz proposes to smooth the n-gram CLMs with a modified version of Good-Turing smoothing (Katz smoothing), discounting counts up to about 6 and a cut-off $K = 0$. However, Katz back-off can be applied with other smoothing methods as well, and so can Katz smoothing with other model combination techniques.

1.6.2. Back-off using linear and non-linear interpolation

The Katz back-off technique requires a choice between the original model $p_A(w|h)$ and the back-off model $p_B(w|h)$, i.e. w is either predicted by p_A or by p_B. **Interpolation** differs in the sense that both models influence the next word probability over the complete vocabulary.

There are two variants: **linear** [Jelinek, 1990] and **non-linear** [Ney et al., 1994] interpolation. Linear interpolation is the simplest:

$$\mathrm{LI}(p_A, p_B)(w|h) = (1 - \lambda(h))\, p_A(w|h) + \lambda(h)\, p_B(w|h), \tag{1.46}$$

where $\lambda(h)$ is an interpolation weight between 0 and 1. In general, there is one interpolation weight for each context h (**context-dependent** linear interpolation). In practice it will always be necessary to define bins of contexts that have equal weights in order to limit the number of parameters (this is also known as parameter *tying*). Binning according to context count $c_D(h)$ is quite common practice. If there is only one bin, then the linear interpolation is called **context-independent**.[4]
The estimation of the interpolation weights is done on data that was not used for the estimation of p_A and p_B. One can reserve a small portion of the training data for that purpose. This approach is an application of the deleted estimation method [Duda and Hart, 1973] and is commonly known as **deleted interpolation** [Jelinek and Mercer, 1980]. Alternatively one can simulate $|D|$-fold cross-validation ($|D|$ being the number of events in the training corpus) by creating held-out corpora consisting of one event each and maximizing the joint log-likelihood of the held-out corpora; this method is called **leaving-one-out**. Closed-form formulas for the estimation of the interpolation weights can be obtained in some cases [Ney et al., 1994].

In non-linear interpolation, the weight of p_A does not remain constant over the complete vocabulary V, but depends on $c_D(w,h)$ (actually, on w and h in general). On the other hand, the non-linear interpolation is still linear in p_B, which is different from back-off.

$$\mathrm{NLI}(p_A, p_B)(w|h) = d_{c_D(w,h)}\, p_A(w|h) + \lambda(h)\, p_B(w|h). \tag{1.47}$$

In [Ney et al., 1994] a detailed exposition is given for the case of absolute discounting, but non-linear interpolation is applicable with other discounting methods as well.
The weight of p_B must be chosen so that the interpolated model is still normalized for each h:

$$\lambda(h) = 1 - \sum_{w \in V} d_{c_D(w,h)}\, p_A(w|h). \tag{1.48}$$

It is, in other words, the discounted probability mass, as is the case with linear interpolation (see above).

1.6.3. A special case: Kneser-Ney modified back-off distribution

Both Katz back-off and interpolation back-off are frequently used for *smoothing*. In that setting, $p_A(w|h)$ is the model to be smoothed, while $p_B(w|h)$ is a background model that is trained from the same source as p_A, but is supposed to be more robust and less detailed; hence the term 'back-off'.

4. Rational interpolation [Schukat-Talamazzini et al., 1997] is actually a context-dependent linear interpolation method (although the authors call it non-linear). The idea is to scale the λ's in proportion with the confidence that one has in each of the models, as given by a heuristic confidence function. One could interpret rational interpolation as a parametric alternative to context-dependent λ's tied according to context count.

If $p_A(w|h)$ is a word-based n-gram $p_A(w_i|w_{i-n+1}^{i-1})$, then a common choice for $p_B(w|h)$ is a smoothed $(n-1)$-gram $p_B(w_i|w_{i-n+2}^{i-1})$. The fundamental problem here, observed by Kneser and Ney [1995], is the following: one might estimate $P(w_i|w_{i-n+2}^{i-1})$ as a marginal

$$\begin{aligned} P(w_i|w_{i-n+2}^{i-1}) &= \sum_{w_{i-n+1}} P(w_i, w_{i-n+1}|w_{i-n+2}^{i-1}) \\ &\simeq \sum_{w_{i-n+1}} \mathrm{KBO}(p_A, p_B)(w_i|w_{i-n+1}^{i-1}) P(w_{i-n+1}|w_{i-n+2}^{i-1}), \end{aligned}$$

but this estimate will generally be inconsistent with the training data. For example, assume $n = 2$ and that in the training corpus $\mathtt{Francisco}$ followed $\mathtt{San}$ 10 times, but did not occur after any other token. Then $P(\mathtt{Francisco})$ should not be greater than $P(\mathtt{San})$, but based on KBO, one would estimate

$$P(\mathtt{Francisco}) \simeq P(\mathtt{San}) + p_B(\mathtt{Francisco}) \sum_{w \neq \mathtt{San}} \lambda(w) P(w),$$

where $\lambda(w)$ is the normalization factor. In this case, $\mathrm{KBO}(p_A, p_B)$ clearly overestimates $P(\mathtt{Francisco}|w)$ if $w \neq \mathtt{San}$. The reason is that $p_B(\mathtt{Francisco})$ is too large. A modified background distribution p_B is found by explicitly requiring that

$$p_B(w_i|w_{i-n+2}^{i-1}) = \sum_{w_{i-n+1}} \mathrm{KBO}(p_A, p_B)(w_i|w_{i-n+1}^{i-1}) \cdot P(w_{i-n+1}|w_{i-n+2}^{i-1}),$$

which leads to the solution

$$p_B(w_i|w_{i-n+2}^{i-1}) = p_{\mathrm{KN}}(w_i|w_{i-n+2}^{i-1}) = \frac{n_{>0}(\cdot w_{i-n+2}^{i})}{n_{>0}(\cdot w_{i-n+2}^{i-1}\cdot)}, \tag{1.49}$$

where $n_{>0}(\cdot w_{i-n+2}^{i})$ is the number of distinct observed n-grams ending in w_{i-n+2}^{i} and $n_{>0}(\cdot w_{i-n+2}^{i-1}\cdot)$ is the number of distinct observed n-grams ending in $w_{i-n+2}^{i-1}w$, where w is arbitrary.

The analysis was repeated for linear interpolation by Chen and Goodman [1999] and the same result for p_{KN} was obtained. In a practical implementation, p_{KN} has to be smoothed as well by discounting the quantities $n_{>0}(\ldots)$ and back-off to or linear interpolation with a lower order Kneser-Ney background distribution.

In the literature, the term 'Kneser-Ney back-off' actually means 'Katz back-off to a Kneser-Ney modified lower-order distribution', and 'Kneser-Ney interpolation' means 'linear interpolation back-off to a Kneser-Ney modified lower-order distribution'. As Kneser and Ney [1995] advocate absolute discounting, the use of absolute discounting with Kneser-Ney back-off and interpolation is usually also implied; however, it is possible to combine for instance Good-Turing discounting with Kneser-Ney back-off.

Excellent results from broad-coverage experiments were reported by Chen and Goodman [1999] and Ney et al. [1997] with Kneser-Ney back-off and even better with Kneser-Ney interpolation. Kneser-Ney interpolation is now commonly considered a fair baseline for evaluation of novel smoothing techniques.

1.6.4. Maximum-entropy modeling

Unlike the other combination techniques discussed so far, **maximum-entropy** language models are not constrained to combining information from other language models only, but can combine knowledge of virtually any kind. The knowledge is represented by real functions (including pmfs) $f_i(h,w)$ of the event space, called **features**. Features are mostly **binary**, i.e. they return either 0 or 1. For instance, a trigram feature returns 1 if the last two words of h and w equal a certain trigram (which defines that feature). Or one could define a feature that equals 1 only if w is a noun, etcetera.
Let $p(h,w)$ be the LM to be estimated, subject to a set of linear constraints

$$E_{p(h,w)}[f_i(h,w)] \doteq \sum_{h,w} p(h,w) f_i(h,w) = K_i, \tag{1.50}$$

where $E_{p(h,w)}$ denotes the expectation operator according to the distribution $p(h,w)$. It is assumed that the set of constraints is **consistent**, i.e. there exists at least one solution. A common choice for K_i is the empirical average of $f_i(h,w)$ over a training corpus — the resulting linear constraint is then called *natural*. Further consider a prior model $p_0(h,w)$ to which $p(h,w)$ should be as close as possible in Kullback-Leibler sense under the constraints (1.50).[5] Then Jaynes [1957] proved that $p(h,w)$ is an exponential model:

$$p(h,w) = Z p_0(h,w) \exp\left(\sum_i \lambda_i f_i(h,w)\right), \tag{1.51}$$

with Z a normalization factor. The feature **weights** λ_i have to be determined such that $p(h,w)$ meets the linear constraints. There is no closed form solution in general; usually *generalized iterative scaling* [Darroch and Ratcliff, 1972] or *improved iterative scaling* [Della Pietra et al., 1997] are used. ME training is computationally very expensive, and the more so on large training corpora.
The natural choice of K_i is the expectation of f_i with regard to the *real* $P(h,w)$:

$$K_i = E[f_i(h,w)] = \sum_{h,w} P(h,w) f_i(h,w), \tag{1.52}$$

but $P(h,w)$ is unknown. Instead one could replace P with p_{ML}:

$$K_{i,\mathrm{ML}} = \sum_{h,w} p_{\mathrm{ML}}(h,w) f_i(h,w). \tag{1.53}$$

However, the only exponential model that satisfies these constraints *is* in fact p_{ML}! Therefore, constraint *smoothing* is crucial [Rosenfeld, 1996, Martin et al., 1999, Chen and Rosenfeld, 2000]. An interesting alternative is *fuzzy* ME smoothing: this method,

5. If p_0 is uniform (no prior knowledge), then the minimum divergence solution has maximum entropy.

developed in [Della Pietra and Della Pietra, 1993], replaces the constraints with a sum of squared errors in the target function, which now becomes

$$D(p\|p_0) + \sum_i \left(E_p[f_i(h,w)] - E_{p_{\mathrm{ML}}}[f_i(h,w)]\right)^2 .$$

Very favorable results with a fuzzy ME smoothed n-gram CLM were reported by Chen and Rosenfeld [2000].

1.6.5. Log-linear interpolation

Log-linear interpolation [Klakow, 1998] is closely related to maximum-entropy models, but training is much easier when there are already a number of other CLMs available. Let the CLMs be $p_i(w|h)$, $i = 1 \ldots N$. Suppose that one knows the Kullback-Leibler distance of the result of the interpolation, denoted here by $\mathrm{LLI}(p_1^N)$, to each of the given models: $D(\mathrm{LLI}(p_1^N)\|p_i) = d_i$. Furthermore, $\mathrm{LLI}(p_1^N)$ is required to be as close as possible (in KL sense) to the uniform distribution. Using Lagrange multipliers, it is then easy to prove that its format must be

$$\mathrm{LLI}(p_1^N)(w|h) = Z(h,\lambda_1^N) \prod_{i=1}^{N} p_i(w|h)^{\lambda_i}, \tag{1.54}$$

where $Z(h,\lambda_1^N)$ is a normalization factor that depends on h and λ_1^N. In terms of log-probabilities, the interpolation is linear, hence the name 'log-linear' interpolation. The λ_i are estimated by an iterative maximization of the likelihood of a held-out corpus that can be small because of the small number of parameters to be estimated. The estimation of the d_i is implicitly done by the estimation of the λ_i.

In [Klakow, 1998] it is further claimed and experimentally shown that the LLI technique is superior to context-independent linear interpolation with an identical number of free parameters, identical training data and identical p_1^N. Ignoring the fact that the training process consumes more computing time due to the computation of the normalization factor, LLI seems preferrable to LI in some cases.

1.7. Empirical comparison of language modeling techniques

Most publications that introduce a novel language modeling technique typically report experiments showing a small but significant gain with the technique over a trigram baseline. However, the practical value of a technique depends on the size of the training corpus and the language style of the target application. The influence of these factors is typically not measured in such experiments. Also, a novel technique should be complementary to other techniques: its benefit should remain visible in combination with other techniques.

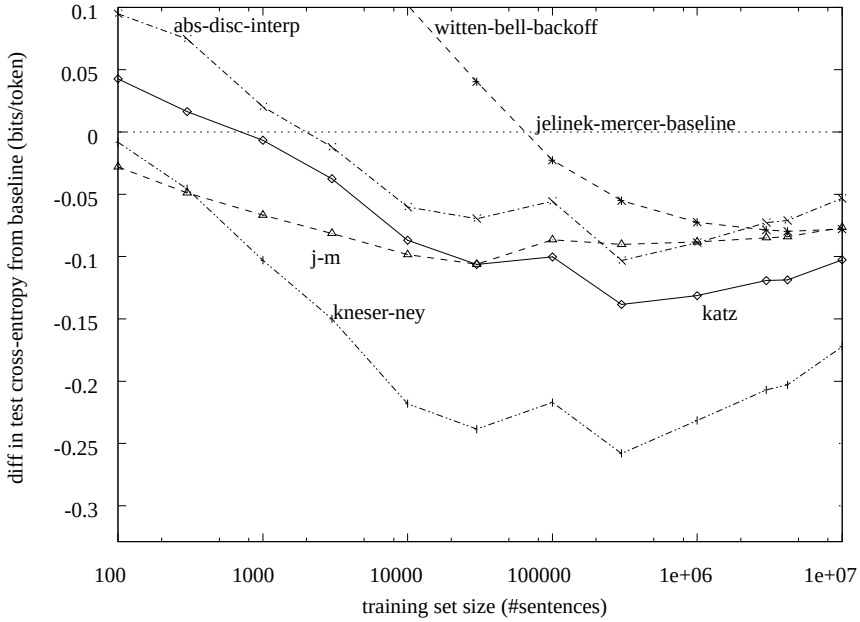

Figure 1.1. Relative cross-entropies of trigrams using different smoothing techniques at varying training data size. Reproduced from [Goodman, 2001a, Fig. 1].

Publications that systematically compare several smoothing techniques and combinations thereof, with different training corpus sizes and different applications, are rare. I will now discuss a relatively comprehensive set of experiments by Goodman [2001a].

Goodman's paper does not compare smoothing techniques across domains. The training and test data are all in the North American Business News domain [Stern, 1996]. Importantly however, it contains results with a very large training corpus of 284M tokens. The experiments indicate that interpolated Kneser-Ney back-off with absolute discounting consistently outperforms Katz-style back-off with Good-Turing discounting, especially in estimating 4-grams from a large training corpus ($>$ 10M–100M tokens). Cache models yield worthwile entropy gains at training corpus sizes less than 100M tokens, but remained ineffective in the speech recognition experiments (the recognition test set consisted of sentences randomly drawn from different articles).

Fig. 1.1 compares trigram models smoothed with different combinations of discount and back-off techniques, for varying training corpus sizes. The difference of each model's empirical cross-entropy with the one of the baseline model is taken as the quality measure. The baseline trigram is smoothed with linear interpolation back-off without discounting. The 'j-m' model is smoothed with Jelinek-Mercer discounting and linear interpolation back-off. The 'katz' model is smoothed with Katz discounting and Katz back-off. The 'abs-disc-interp' uses absolute discounting with linear

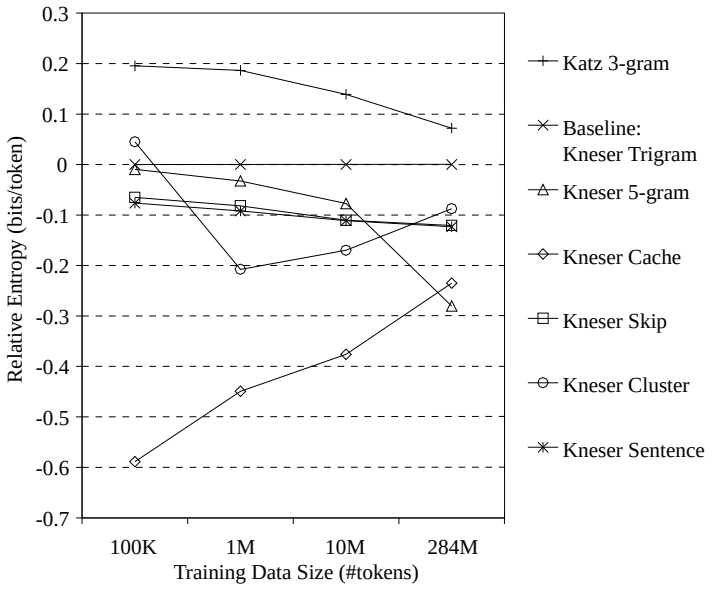

Figure 1.2. Relative cross-entropies of different language model classes using Kneser-Ney linear interpolation back-off smoothing. Reproduced from [Goodman, 2001a, Fig. 9].

interpolation back-off. The 'kneser-ney' model is smoothed with absolute discounting and Kneser-Ney linear interpolation back-off. The plot shows the superiority of the 'kneser-ney' model over the 'katz' model and the 'j-m' model at training corpus sizes from 100k sentences (about 2M tokens).

Fig. 1.2 is adapted from [Goodman, 2001a, Fig. 9]. It compares the difference of empirical cross entropies of a number of language models with a baseline 'Kneser Trigram' model. With the exception of the 'Katz Trigram' model, all of these models are smoothed with interpolated Kneser-Ney back-off distributions and absolute discounting. The 'Katz trigram' is smoothed with Katz back-off and Katz discounting. The 'Kneser Cache' model is a linear interpolation of a smoothed unigram cache model, a trigram cache model and the baseline model. The 'Kneser Skip' model is a linear interpolation of number of smoothed skipping 4-grams (in the style of delayed *n*-grams) with the baseline model. The 'Kneser Cluster' is a smoothed category-based 5-gram with automatically obtained word categories. The 'Kneser Sentence' model is a sentence mixture model with automatically obtained sentence types.

Fig. 1.2 shows that the 'Kneser 5-gram' is not really worthwile at a training corpus size of 10M tokens, but becomes powerful at the full training corpus size of 284M tokens. The remarkable power of cache models decreases with larger training corpus sizes.

Finally, Fig. 1.3 shows results from speech recognition experiments with different

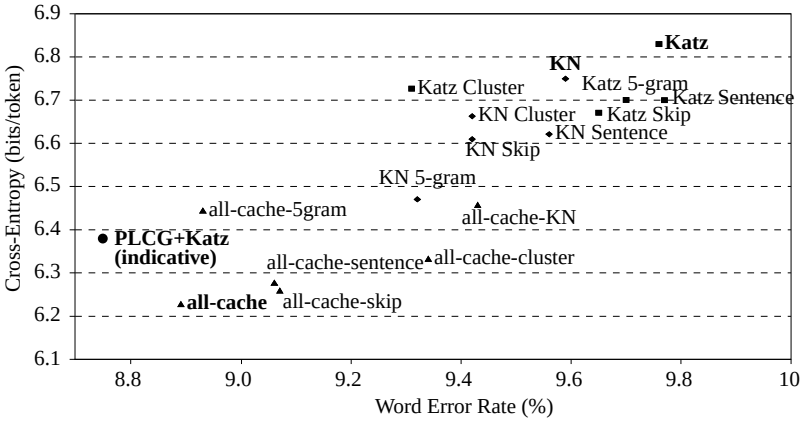

Figure 1.3. Word error rate versus cross-entropy. Reproduced from [Goodman, 2001a, Fig. 12]. The point corresponding with the PLCG-based model is added by us and only indicative, since training and test conditions differ.

model classes and smoothing techniques. The 'Katz' model is a Katz trigram; the 'KN' model is a trigram smoothed with Kneser-Ney linear interpolation back-off. The 'Katz . . . ' models use Katz back-off and discounting and interpolation with the 'Katz' model. The 'KN . . . ' models use Kneser-Ney linear interpolation back-off and interpolation with the 'KN' model. The 'all-cache' model combines all the techniques dealt with in the paper, except cache-based modeling — the latter turned out to harm the speech recognition. The other 'all-cache-. . . ' models are equivalent with the 'all-cache' model, but with one more technique left out: for instance, the 'all-cache-cluster' model does not have the 'cluster' component (a category-based 5-gram).

I have also superposed a 'PLCG+Katz' point, at -12% in word error rate and $-.46$ bits/token from the 'Katz' point; these were the best obtained results with the PLCG-based language model relative to a trigram baseline. The point is indicative, because the training and test conditions of this thesis differ from Goodman's.

1.8. Conclusion

This chapter defined the research field of statistical language modeling, discussed the role of language models in natural language applications and gave a literature overview of common language model estimation techniques.

The estimation of a statistical language model is difficult: in the categorical space of words and sentence prefixes there is no obvious definition of a distance measure, which hampers generalizing from training data. Therefore statistical language models typically contain a very large number of parameters, and consequently face data sparseness throughout.

Language modeling research has come up with a number of observation clustering techniques to mitigate data sparseness, discounting techniques to compensate for the bias of relative frequencies of small samples, and model combination techniques to improve the robustness against rare events or to relax the specificity of the model. A systematic overview of the most common of these techniques was given, and their specific strengths and weaknesses were mentioned. Appropriate smoothing is important, but there is no single best combination of smoothing techniques; at most there are a few heuristic rules of thumb that may speed up the finding of a satisfying smoothing strategy.

The results from an experiment by Goodman were included, comparing combinations of several language model techniques trained and tested on the North American Business news corpus. Kneser-Ney back-off turned out consistently better than Katz back-off.

Jelinek-Mercer smoothing seems less powerful than Katz and Kneser-Ney back-off, but can be used with non-integer observation frequencies (they will show up in the reestimation procedure of the PLCG-based language model).

CHAPTER 2

Grammar-based language models

This chapter motivates grammar-based language modeling and contains an overview of the relevant literature in order to situate the PLCG-based language model, to show in which ways it contributes to the research domain, and to serve as a collection of references for those who want to become familiar with grammar-based language modeling.

2.1. Motivation

Human language is obviously *structured*. It is only logical that language modeling researchers have always attempted to exploit that fact. Language models that predict language structure explicitly will be called *grammar-based* in this chapter, since it is precisely *grammar* that describes sentence structure. The language model components discussed in Chapter 1 can be used in combination to form powerful and robust LMs/CLMs. All of these, however, fail to recognize *syntactically structured* dependencies within a sentence.
N-grams always assume the same simple structure, while cache and trigger models do not assume any structure at all. Consider, for instance, the following sentences:

(a) The news agent has been running this story.
(b) The news agent has been running this boring story.

A human reader would assume that the probability of ‘story’ in sentence (b) should be comparable with the one in sentence (a). However, a trigram CLM would only see ‘running this’ when predicting ‘story’ in (a), and ‘this boring’ in (b); it fails to generalize the trigram ‘running this story’ to any slightest variant of it. Cache- and trigger-based models may account for ‘news agent’ and ‘running’ boosting the probability of ‘story’, but almost equally they would estimate the probability of ‘all’ in the next sentence:

(c) The news agent has been running this story all week.

which they should not.
While it cannot be denied that there are many non-structural and local patterns in a language, I believe that grammar is a substantial factor in the sentence generation process. Grammar-based CLMs are therefore expected to become a powerful *complement* to the other conventional language modeling techniques.
In the long term, my study is also motivated by the intuition that preference patterns of syntactic structure are relatively invariant over different discourse contexts. Proper identification of the invariant parts of language will permit language models to be more adaptive, since it would allow to reduce the number of parameters to be adapted. Adaptation is certainly still a weak point of the conventional, mostly word-based, LM techniques currently available.

2.2. Grammatical theories and generalization

Grammar-based language models have gone through an evolution in various aspects: the underlying grammatical theory, parsing techniques, application domain, model development and system integration. These characteristics are quite related; this section is structured according to the formal characteristics of the grammar and discuss their repercussion on the other aspects in the go.
The evolution of the grammatical theories *underlying language models for speech decoding* can be described as climbing up the **Chomsky hierarchy** [Chomsky, 1959] (see also [Jurafsky and Martin, 2000, pp. 478–481] and [Hopcroft and Ullman, 1979, Ch. 9]). The latter is a well-known classification of formal grammars according to their *generative* power due to Chomsky: in decreasing order of generative power, the following classes are discerned:

- Recursively enumerable grammars (REG) or Turing machines: rules are of the format $\alpha \rightarrow \beta$. This is the most general format.
- Context-sensitive grammars (CSG): the format of the rules is $\alpha A \beta \rightarrow \alpha \delta \beta$; a CSG rule must rewrite a non-terminal A as a sequence δ of terminals and non-terminals, where a context, given by α and β, may be specified.
- Context-free grammars (CFG): a CFG rule rewrites a non-terminal to a sequence δ of terminals and non-terminals, but no context can be specified (this is the context-freeness assumption). So the format of the rules is $A \rightarrow \delta$.
- Finite-state grammmars (FSG) or regular grammars: the rule format is $A \rightarrow a\delta$ (for a right-branching FSG): every FSG rule must generate a terminal symbol.

The Chomsky hierarchy is a classification of grammars according to both *weak* and *strong* generative power. The *weak* generative power of a grammar is the set of sen-

tences that can be generated, while *strong* generative power is the set of sentences *and* their corresonding structures.

From the perspective of computational learning, it is the *strong* generative power that matters, since it is strongly related with the capability of a computational model to induce correct, elegant generalizations. For instance, the weak generative power of a CFG can be arbitrarily closely approximated with a FSG, but it takes a lot more rules and the grammatical structure would be implausible in most cases (namely, exclusively right-branching for a right-branching FSG).

While the earliest speech recognition systems were based on FSGs, the attention shifted to CFGs. Since a few years, researchers propose models that violate the context-freeness assumption, e.g. various unification-based grammars; these are specific types of CSGs. A *generic* treatment for CSGs — let alone REGs — is not yet within reach, but probably not immediately useful for human natural language.

The evolution from FSG- to CFG-based language modeling and beyond is motivated by:

1. *Advances in hardware technology*: The lesser the constraints on the language that can be handled, the more complex algorithms grow. However, the computing resources typically available to researchers, in terms of processing speed and memory, have tremendously grown and keep growing.

2. *Striving to (more) correct modeling in wider application domains*. For instance, research on automatic speech recognition evolves to less constrained domains implying gradually more 'natural' language.

3. *The limited usability of grammar transforms*: algorithms have been developed to convert or approximate a grammar with another one of a lower strong generative power. It is often easier and more intuitive to specify or hand-tune a grammar of a higher strong generative power (perhaps because it is closer to human cognition). A CFG can be approximated with a FSG with an algorithm by Pereira and Wright [1991]. Stolcke and Segal [1994] describe a method to compute precise *n*-gram probabilities from a PCFG. At a higher level, one can mention the Gemini system by Moore et al. [1997], which compiles a subset of unification-based grammars (cf. below, Sec. 2.5.1) into CFGs. However, typically the compiled grammar is very large, and it loses the long-distance modeling capacity of the higher-level grammar; hence there is a need for algorithms and language models that make it easier to integrate higher-level grammars in their native forms into the speech decoding process.

4. *Advances in statistical large-scale parsing*: The more recent work in grammar-based language modeling profits from significant advances in automatic parsing of general text [Manning and Schütze, 1999, Ch. 12], both on the algorithmic and the training data side. The large-scale parsing trend started with IBM's statistical parsing effort [Black et al., 1993, Magerman, 1994]. The Penn Treebank corpus [Marcus et al., 1993], primarily consisting of 49,000 human-annotated sentences from Wall Street Journal articles, made it possible to induce large-coverage probabilistic grammars and

estimate probabilities for each of their parameters [Magerman, 1994, Collins, 1997, Ratnaparkhi, 1997, Charniak, 2000]. Furthermore, the PARSEVAL metric [Black et al., 1991] provided a cheap and automatic way to compare the performance of possibly entirely different parsing systems. The best current parsing systems boast an accuracy of about 90% in terms of labeled precision and recall. Although there is discussion about the relevance of the PARSEVAL metric and the lack of detail in the Penn Treebank [Caroll et al., 1998, Manning and Carpenter, 1997], the basic message for the language modeling community is that statistical parsing with even a very simplistic grammar is still able to provide useful cues on the expected structural dependencies between words within one sentence.

Language models based on FSG and CFG are discussed in the next two sections, respectively. Then developments beyond CFG are described.

2.3. Finite-state grammars

Finite-state grammars are equivalent with finite-state automata, which are very well understood. In the Chomsky hierarchy they have the least generative power. The probabilistic extension of FSG (PFSG) can be implemented as a *weighted* finite-state automaton.

FSG-based language models can be compiled into the search network of a speech decoder and generally lead to a highly efficient single-step speech decoder. However, for robustness it may sometimes be preferred to postpone the grammar constraints to a second pass. Loose coupling of the acoustic match and the language module allows a greater flexibility. For instance, in [Ward et al., 1988], a class-based trigram is applied on the lattice output using an island-driven search strategy. On the Resource Management task, a small-vocabulary speech recognition test suite [Price et al., 1988], robustness against missing words was improved with respect to left-to-right search. Jackson [1992] proposed a hybrid approach: an FSG, essentially a template matcher, spots phrase islands in a recognized sentence, and a CFG then checks whether they form a meaningful sentence. The approach was only tested for language understanding in ATIS, a travel information domain serving as a test for small-vocabulary automatic speech recognition and understanding. Interestingly, Ward and Young [1993] proposed the opposite, i.e. recognizing phrases with a CFG and scoring those with a phrase-level trigram. Word error rate on the ATIS task was reduced from a 23.5% bigram baseline to 18.2%.

While FSG-based language models are typically associated with the early automatic speech recognition systems [Bahl et al., 1978, Lowerre and Reddy, 1980, Chow et al., 1987], they continue to be used because of their computational efficiency or just because the target language is, in fact, a regular language. For some restricted domains (e.g. database query systems), finite-state grammars are a good choice. This is the case, for instance, if user input is adequately modeled with a limited number of *tem-*

plate sentences such as '(want-ticket) from (airport) to (airport) on (day-of-week)'.
In LVCSR applications, such as general dictation, FSGs need many rules, do not generalize well and have difficulty in integrating prior linguistic knowledge; they need a very large training corpus and are very specific to that corpus. These are precisely the disadvantages of the *n*-gram model, which is a PFSG: each history w_{i-n+1}^{i-1} corresponds with a state and at the transition (taken with a certain probability conditional on w_{i-n+1}^{i-1}) from a state corresponding with w_{i-n+1}^{i-1} to a state corresponding with w_{i-n+2}^{i}, a symbol w_i is emitted. The word-pair model, on the other hand, is an example of an FSG (non-stochastic bigram).
FSGs are also used as a 'compiled' format derived from (mostly handwritten) CFGs. Automatic procedures have been described to approximate PCFGs by *n*-grams [Stolcke and Segal, 1994] or more general PFSGs [Pereira and Wright, 1991]. This approach is quite common in current speech recognition APIs. It is typically used for small-vocabulary applications where the language is rather controlled, such as natural language interfaces with central services.

2.4. Context-free grammars

It was stated earlier that, for modeling natural language, CFGs are preferrable to FSGs with regard to their strong generative power, but are more difficult to integrate in a speech decoder and considerably more complex to parse. CFGs can describe language structure in a way that can be interpreted by humans, except for a few phenomena such as cross-serial dependencies in Dutch.[1] For limited domains, CFGs may be sufficiently powerful.
A CFG is defined by a set of terminals (word vocabulary), non-terminals one of which is the start symbol or the axiom, and a set of production rules. A sentence is generated, starting from the axiom, by rewriting a non-terminal (the lefthand side of the rule) as a string of terminals and non-terminals (the righthand side of the rule). This process is repeated until no non-terminals remain. The *context-freeness* assumption entails that the rewriting of a non-terminal is independent of what comes before or after that non-terminal.
For example, the following rules constitute a simple CFG (see Appendix D for the meaning of the symbols):

S → *NP VP*	*VP* → *VBZ NP*
NP → *the NN*	*NN* → *market*
NN → *industry*	*VBZ* → *supports*
VBZ → *creates*	

Specific CFGs that are written by hand for limited tasks are often expressed in terms of task-oriented semantic category labels, such as *AMOUNT* or *DESTINATION*. Attempts

1. As, for example, in: 'Ik heb hem zijn vader het dak zien helpen herstellen', where 3 object-subject dependencies cross.

to write CFGs for general language have met with problems of sufficient coverage. Large CFGs can alternatively be derived from treebanks (i.e. parsed text corpora) such as the Penn Treebank [Marcus et al., 1993] with better coverage; however, treebank grammars tend to overgenerate extremely [Charniak, 1996, p.7]. Therefore they only make sense in a stochastic framework.

A probabilistic or stochastic CFG (PCFG, SCFG) is a CFG equipped with production rule probabilities, indicating the conditional probability that a given non-terminal is rewritten by a certain rule, and not by other rules for which the lefthand side matches. While the CFG can only impose *hard constraints* (allow or disallow word sequences), the PCFG is more versatile since it produces *scores*. These scores can be combined with scores from other knowledge sources and afterwards compared with a pruning threshold.

According to a PCFG, the probability of generating a sentence by a specific sequence of productions is simply the product of the probabilities of the rules that were applied in generating that sentence. This probability can be written as a joint probability $P(W,T)$, where W is the sentence and T is the analysis tree. The probability of a sentence W, then, is the sum of $P(W,T)$ over all T for the same W; algorithms that compute this sum efficiently are known [Jelinek and Lafferty, 1991]. The rule probabilities are bootstrapped from a small treebank and maximum-likelihood trained in an unsupervised mode on plain text using the inside-outside technique [Baker, 1979], actually a generalization of forward-backward training of HMMs and PFSGs, and an instance of the expectation-maximization (EM) algorithm [Dempster et al., 1977]. Unfortunately, the likelihood function usually contains many local maxima.

Both CFG and PCFG have been proposed as language models in speech decoding since the mid 1980's. In 1985, Derouault and Merialdo [1985] use a phoneme-based hand-written PCFG to rescore n-best lists of phoneme strings. A paper by Matsunaga et al. [1990] is complementary in that it presents parsing of a phoneme lattice (resulting into a word/phrase lattice, that is ultimately rescored with a dependency grammar), but does not use probabilities.

In 1986, Tomita adapted his 'left-right' (LR) parsing technique to select the grammatical sentences from word lattices resulting from a first speech decoding pass. The LR parser is basically a shift/reduce parser and is very efficient due to a look-up table that is computed beforehand. LR parsing has frequently been used in similar settings [Ward et al., 1988, Su et al., 1991] and extended in various ways: Kita and Ward [1991] propose a tight integration within the acoustic match by associating an LR stack with each search path. As already mentioned earlier, combined use of a CFG and a FSG is also possible. For instance, Ward and Young [1993] use a CFG on a phrase level only and the sequence of phrases is scored with a phrase trigram afterwards for reasons of robustness. GLR and GLR* lattice parsing [Lavie and Tomita, 1993] are extensions of LR, able to skip ungrammaticalities in the input. An

adaptation of Tomita's parser to handle PCFGs is due to Wright [1990].[2] A simpler alternative is provided by Goddeau and Zue [1992], who use state and shift probabilities instead of the usual PCFG production rule probabilities. Actually this practice of closely linking parser actions with probabilities is central to the general idea of history-based stochastic grammars (see further), and inspired the more recent line of context-sensitive grammar-based stochastic language models, which will be surveyed below.

Other CFG parsing algorithms than GLR have been used as well in CFG based language models. In [Nakagawa, 1987, Kai and Nakagawa, 1992], an *Earley parser* is used in a single pass recognizer to predict the set of possible next words, which are examined by a word spotter. An Earley chart parser was equally used in Broca, a speech recognizer that integrates the language model and the pronunciation lexicon in one CFG [Howells et al., 1992]. Stolcke [1995] proposed a language model based on an Earley parser with a CFG that can compute sentence prefix probabilities as well. It applies the principles of generative stochastic grammar right away: a prefix probability is computed as an (implicit) sum of probabilities of partial derivations. A derivation is a sequence of stochastic Earley parser moves (shift, complete, produce). The computation of the sum is made efficient with dynamic programming (DP). The language model is successfully combined with a word bigram in the forward search pass of the speech recognizer of the BeRP dialog system (a restaurant guide) [Jurafsky et al., 1995].

The CKY (Cocke-Kasami-Younger) algorithm is another well-known (P)CFG parsing algorithm that applies DP. It was used in various early speech recognition systems, for instance to extract the grammatical sentences from a lattice [Young et al., 1988, Chow and Roukos, 1989] or in a single pass recognizer [Ney, 1987, 1991].

As a final comment regarding the integration of CFG constraints,[3] one has to choose between a multi-pass or single-pass search. A multi-pass search involves rescoring a lattice or an n-best list. A single-pass search may be more efficient due to early pruning of ungrammatical partial hypotheses, but also less efficient because parsing effort is expended on hypotheses that would have been excluded from the lattice in a multi-pass search scheme.

In a single-pass search strategy, unlike FSGs, CFGs cannot be compiled into a finite static search network. Alternative strategies are dynamic network expansion (e.g. [Murveit and Moore, 1990, Brown and Glinski, 1994]) or interleaved grammar checking, such as [Ney, 1987, Nakagawa, 1987]. The grammaticality is only checked *after* the acoustic match in [Ney, 1987]. In contrast, a language model based on an Earley parser such as [Nakagawa, 1987] only hypothesizes a next word if it can lead to a grammatical partial hypothesis.

2. In further work, Wright describes a procedure to construct PCFG parse trees in an optimal order such that a chain of first-order conditional transition probabilities approximates the true joint probability as closely as possible [Wright, 1991, Wright et al., 1992].

3. A literature overview on this subtopic was presented by Chappelier et al. [1999].

2.5. Beyond context-free

There are no parsing algorithms (yet) that can handle all CSGs, but some specific classes can be handled reasonably efficiently with dedicated algorithms. Specifically stochastic phrase-structure grammars that incorporate non-local probabilistic dependencies have led to powerful stochastic language models — among which the PLCG-based language model.

2.5.1. Unification-based grammars

Unification-based grammars (UBGs)[4] assign *typed feature structures* to constituents instead of simple category labels. Features provide an elegant way to specify detailed syntactic and semantic constraints, such as agreement (person, number, tense) and subcategorization (the list of expected complement categories), while keeping the grammar concise and intuitive. Unification of feature structures is a key element in almost all modern grammatical frameworks, including lexical-functional grammar [Bresnan, 1982, 2001], head-driven phrase structure grammar [Pollard and Sag, 1994], construction grammar [Kay and Fillmore, 1999] and unification categorial grammar [Uszkoreit, 1986].
Consider, for example, the following CFG rule: $S \rightarrow NP\ VP$. This rule generates combinations of an NP and a VP that one would wish to exclude, for example combinations of a singular NP with a plural VP. It is possible to replace the rule with two rules: $S \rightarrow NPsg\ VPsg$ and $S \rightarrow NPpl\ VPpl$. However, this process has to be repeated for the person agreement. One realizes that this approach cannot be maintained, because a combinatorial explosion of category labels and CFG rules will result. A more elegant solution is obtained by assigning feature structures to constituents, for example:

$$VP \begin{bmatrix} PERSON & 3rd \\ NUMBER & singular \end{bmatrix},$$

and *augmenting* the CFG rule with a number agreement constraint:

$$S \rightarrow NP\ VP,\ NP{:}NUMBER = VP{:}NUMBER.$$

The operator by which two feature structures are combined or marked incompatible is called *unification*. Long-distance dependencies can be modeled because unification can propagate features values from daughter nodes to mother nodes or vice versa.
The unification principle has been applied to various grammar-based language models. An early example is a finite-state speech recognizer by Hemphill and Picone [1989], which parses with augmented stochastic regular grammars on both the sentence and word level. The word-level grammar emulates a pronunciation lexicon, where phonetic feature constraints ensure that assimilation rules are followed within and across

4. A detailed account of unification and a literature overview can be found in [Jurafsky and Martin, 2000, Ch. 11], [Shieber, 1986] and [Knight, 1989].

word boundaries. The parsing algorithm is an augmented chart-based Earley parser, extended to handle feature structures and probabilities.
Several other publications deal with language models based on augmented CFGs. There are basically two approaches:

1. The UBG is transformed (exactly or approximately) into a large CFG, and the original software can be used. See, for instance, [Black et al., 1993, Rayner et al., 2000]. The scalability of this approach is quite limited.

2. A CFG parser is extended to deal with features. Then, again there are two options: either the features are used as (categorical) filters on the otherwise unchanged operation[5] of the probabilistic or non-probabilistic CFG parser, or the features are considered part of the probabilistic sentence generation. Language models following the former option include [Derouault and Merialdo, 1985, Chow and Roukos, 1989, Seneff, 1989, Okada, 1991]; the latter option is followed by Goodman [1997], among other more recent publications on stochastic grammar-based language models.

UBG-based language models as cited above allow handcrafting a fairly accurate grammar for small natural language applications in a reasonable time. However, application in large-vocabulary applications, such as LVCSR, is impeded by the lack of training data: hand-parsing a sufficiently large text corpus with a UBG is too costly. With the help of a morphological lexicon and/or analyzer, it seems possible to fill the feature value structures automatically, once a coarse annotation in the style of the Penn Treebank is available; but this track is not yet explored for language modeling purposes.
A special feature, that actually *can* be annotated automatically fairly simply — at least in English — is the 'lexical head' feature, giving rise to lexicalized grammars, which are discussed now.

Lexicalized grammars

The first step in analyzing a word in a sentence often entails a generalization to a morpho-syntactic category, such as a part-of-speech. However, the words themselves contain much more information about *syntactic* preferences and cooccurrences than only their parts-of-speech. Capturing this kind of information is important for reducing syntactic ambiguity. Consider, for example, the following sentences:

(a) I donated books to him.
(b) *I donated him books.
(c) I gave books to him.

5. The combination of non-probabilistic unification of features with probabilistic CFG rules leads, if one is not careful, to an incorrect estimate of derivation probabilities. Briscoe and Carroll have discussed probabilistic GLR UBG parsing extensively [Briscoe and Carroll, 1993], but their model is incorrect. A probabilistically sound model is described by Inui et al. [1997].

(d) I gave him books.

Sentence (a) is more likely than (b), because 'donate' does not allow the indirect object to appear before the direct object. However, 'give', although a synonym, allows both forms (c) and (d).
The syntactic preference in the above example is commonly described with the *subcategorization* mechanism: the idea of subcategorization is that each lemma requires ('subcategorizes for') a certain set of complements, and these are listed in the lexicon.[6] Explicit use of a subcategorization feature in a grammar is hard for large-vocabulary purposes, by lack of an extensive detailed subcategorization lexicon. My attempt is to model the *phenomenon* of subcategorization — admittedly in a suboptimal way — with the less knowledge-based lexicalization method of headword percolation (cf. below).
Besides syntactic disambiguation, lexicalization is also important for *semantic* disambiguation. For example:

(a) She beat him with her oboe.
(a) She beat him with her piano.

Although both sentences are syntactically equivalent and correct, and both an oboe and a piano are musical instruments, (a) is expectedly more likely than (b) for obvious reasons. A grammar that reduces 'beat' to a tag 'verb', or 'piano' to 'noun' or even 'musical-instrument', has no means to assign (a) a higher probability than (b).
Most often the lexical feature identifies with a lexical property of the **head** of the constituent. The head imposes constraints on the structure of the constituent. The notion of a head that combines with complements and adjuncts is common to most modern formal grammar theories. The best statistical large-scale natural language parsers currently available [Collins, 1997, Charniak, 2000] choose the **headword** (the form of the head as it appears in the sentence) as the lexical category of a constituent. In this (most frequent) case, the unification of the lexical feature is often called **headword percolation**. For English, a set of deterministic headword percolation rules by Magerman [Magerman, 1994] are in popular use.

Example 2 *The parse tree in Fig. 2.1 was taken from the Penn Treebank corpus [Marcus et al., 1993]. It was lexicalized with Magerman's headword percolation rules.*

2.5.2. Dependency and link grammars

Phrase structure grammars represent the structure of a sentence with a tree graph. In contrast, **dependency grammars** use a **dependency graph** that has just as many

6. The concept of subcategorization is strongly related, but the term 'syntactic preference' is more appropriate in the statistical setting, in that it suggests the use of soft constraints as opposed with hard selection (allow or disallow). It is also more general; for instance, the fact that *NN* very probably rewrites to *Friday* in the context of *Thank God it's ...*, cannot be described as a subcategorization phenomenon.

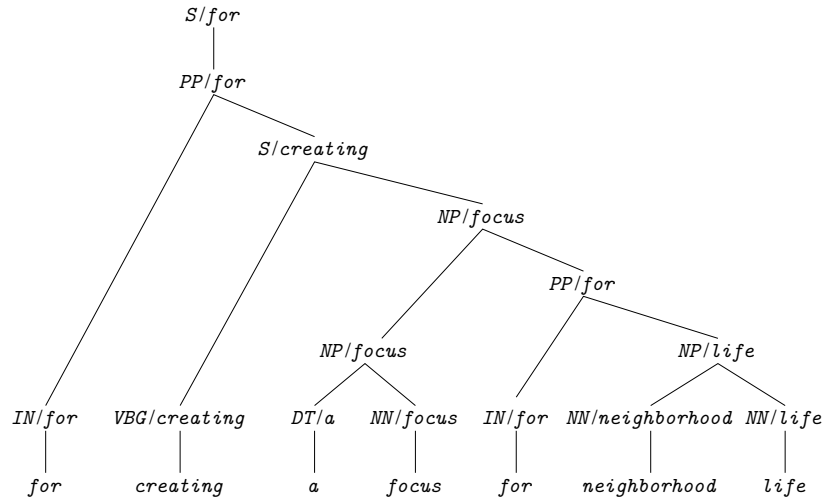

Figure 2.1. Example of a headword annotated sentence.

Figure 2.2. A local tree and its corresponding dependency representation.

nodes as there are words in the sentence. A dependency corresponds with an arc connecting two words; depending on the specific type of dependency grammar, arcs can be directed and/or labeled in order to specify a functional relationship (e.g. object-verb).

Collins' 1996 parser uses a dependency grammar that is extracted from the Penn Treebank. This parser expects a POS-tagged sentence. Before the actual parsing, a baseNP[7] chunking is performed and each baseNP is replaced with its headword. For the extraction of the grammar, the original tree representation is converted into a dependency representation in two steps. First, the tree is headword percolated. Second, each n-ary local tree is converted into $n-1$ dependencies as in Fig. 2.2.

The probability of a parse T given a POS-tagged sentence S is computed as $P(B|S) \cdot P(T|B,S)$, where B is the baseNP segmentation and T is the dependency graph, which is the sequence of outgoing dependency arcs (as many as there are words in the sentence). The parsing algorithm itself is a bottom-up chart parser.

7. A baseNP is a noun phrase that does not contain other noun phrases.

For the statistical parsing community, Collins' parser showed that Magerman's results with a complex history-based grammar [Magerman, 1994] could be obtained with a much simpler grammar. For the language modeling community, it was an important source of inspiration and a development tool for the dependency language modeling project at the Johns Hopkins CSLU 1996 workshop [Stolcke et al., 1997, Chelba et al., 1997]. The goal of this project was to develop a dependency language model that would approximate $P(S)$ as $\sum_{K^*} P(S,K^*)$, where K^* is the linkage (dependency graph) obtained from Collins' 1996 parser. K^* is represented as a sequence of disjuncts $d_0 \ldots d_n$. A disjunct d_i is the set of all incoming and outgoing dependency arcs (links). The dependency language model would compute $P(S,K^*)$ as $\prod_i P(w_i, d_i | w_0, d_0, \ldots, w_{i-1}, d_{i-1})$ where the history is reduced to a trigram context (w_{i-1}, w_{i-2}) plus the open links. All in all, various other shortcuts and approximations had to be introduced and the results obtained were not quite encouraging. A part of the problem was that the Switchboard corpus was chosen as a testbench for 'political' reasons (sic), which is notoriously difficult.

Strongly related with dependency grammar is **link grammar** [Sleator and Temperley, 1991, 1993]. It can be considered as a dependency grammar with undirected and labeled arcs, but differs from the classical dependency framework in three aspects: the link graph must be planar, there can be cycles in a link graph and the order for each link pair is specified (i.e., whether one word can precede or follow the other, or do both).

A probabilistic language model based on link grammar, called 'grammatical trigram', is described by Lafferty et al. [1992]. Moderate results with a simplified version of the model only using automatically obtained word pairs were reported in [Della Pietra et al., 1994]. The dependency language model [Stolcke et al., 1997] was linguistically more sound, but experimental results were not convincing either. From a methodological perspective, however, I believe that dependency language modeling has greatly contributed to later grammar-based language models in two ways:

1. *Stress on lexicalization.* Most often, the best predictors for words are other words, as proven by the good performance of the word-based trigram language model. Lexicalization is important for both statistical parsing and statistical language modeling.

2. *Probabilistic context-sensitivity.* Link grammars are only context-free as far as their categorical base is concerned; that is, the rule probabilities are not. The probability of a dependency is not only conditioned on features that are local to the dependency itself (such as source and target node), but on more distant (1 or 2 dependency arcs further) features too. The practice of heavy probabilistic conditioning is later found essential for both statistical parsing and statistical language modeling.

2.5.3. History-based grammars

History-based grammars (HBGs) arose as an attempt of IBM researchers [Black et al., 1993, Magerman, 1994] to apply statistical n-gram language model techniques to statistical parsing of natural language. HBG starts with decomposing the generation of a sentence in a sequence of elementary parser actions, called a **derivation**. There must be a one-to-one correspondence between derivations and pairs (S,T), where S is the sentence and T a particular analysis of that sentence (according to some grammatical theory). Therefore the generation probability of (S,T) is the probability of the derivation, which can be estimated using conventional language modeling techniques: the probability of the next parser action is conditioned on the previous ones.
A HBG-based language model follows naturally since the generation probability of the sentence is the sum of the probabilities of the derivations that yield that sentence.
The grammar used in [Black et al., 1993] is a PCFG obtained from a hand-built UBG by clustering certain features and feature-value pairs; other features apart from the syntactic category are added, such as a semantic category, a primary lexical head, a secondary lexical head and a daughter index; the features are predicted one by one using the chain rule. Since there is far too much conditioning information, decision trees are used to reduce (i.e. cluster) the context. Although the resulting parser was particularly heavy and complex, it showed that statistical parsing had the potential to outperform any hand-built parser. The modeling method promoted by the experimental results was starting with a simple, large coverage grammar and letting a statistical parameter estimation procedure (on a treebank or plain text if possible) do the rest.

2.5.4. Current state of the art: grammar-based LMs for LVCSR

In restricted application domains syntax-based (C)LMs are commonly used, since it is possible to write a grammar by hand in these cases. However, for LVCSR, syntax-based (C)LMs were for a long time outperformed by simple word-based trigram models. There are at least three causes of failure:

1. The grammars used were context-free.
2. There was no use of fine-grained, particularly lexical, features.
3. Grammars were hand-built and had insufficient coverage.

The former two points cause bad probability estimates; the latter causes zero probability estimates for perfectly acceptable sentences.
A first encouraging advance was realized when Chelba and Jelinek (henceforth abbreviated C&J) proposed a CLM based on a stochastic shift-reduce parser [Chelba and Jelinek, 1999]. This work was particularly encouraging because it was the first to report a significant enhancement of speech recognition accuracy in a read newspaper task by introducing syntactic constraints into a large-scale language model. The statistical syntactic constraints were assembled from a hand-parsed (e.g. the Penn Treebank) or machine-parsed text corpus and could be reestimated on plain text afterwards.

The C&J model's design and its strong emphasis on lexical dependencies are reminiscent of the dependency LM [Stolcke et al., 1997, Chelba et al., 1997]; it can be considered as a generalization of the word-based trigram CLM. Problems 1 and 2 mentioned above are avoided by non-local stochastic conditioning on lexical features: the parser uses probabilistic shift and reduce actions that are conditioned on the two most recent 'exposed heads' (an exposed head is the lexical head of a phrase covered by a subtree that is not yet a subtree in another tree). Problem 3 is tackled by initializing conditional shift and reduce probabilities directly from a treebank (a hand- or machine-parsed text collection) and applying conventional statistical smoothing techniques.

Other syntax-based LVCSR language models have been proposed since. While Chelba and Jelinek [1999] deploy a shift-reduce parser, Roark [2001] and Charniak [2001] do similar things with a top-down parser. Bod [2000] proposed a language model based on data-oriented parsing and proposes a maximum-likelihood estimation procedure for it.

Before discussing some of these models in detail, here are some common characteristics:

1. All of the above cited grammar-based language models are based on a generative probabilistic grammar — a tradition that started with history-based grammars.[8]

2. In contrast with HBGs, the probability of a next parse move is not conditioned on the preceding parse moves, but on their *result*, i.e. a partial parse. Actually for probabilistic conditioning only certain features of the partial parse are selected (context clustering by feature selection). For instance, these features are the lexical and syntactic heads of the two most recent unattached constituents (exposed heads) in [Chelba and Jelinek, 1999]. In [Roark, 2001] a decision tree (a 'tree-walking' function) is used for more sophisticated feature selection. Generally there is a tendency to include lots of features, thereby increasing data fragmentation and necessitating statistical smoothing techniques.

3. The grammars are highly lexicalized.

4. The language models are computationally very expensive to train and to apply. Furthermore, they cannot be tightly coupled with the speech decoder, they can only be used for rescoring a severely pruned lattice or an n-best list.

8. A generative probabilistic grammar Γ leads to a joint probability distribution $P_\Gamma(W,T)$ over sentences W and linguistic analyses T. The associated language model $P_\Gamma(W)$ is simply the marginal of $P_\Gamma(W,T)$. An important question is how to compute this marginal both efficiently and accurately, since the joint (W,T) space is infinite when dealing with natural language. Also, it is often desirable that prefix probabilities can be computed.

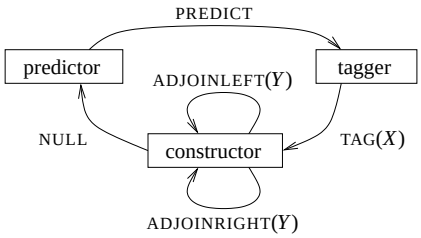

Figure 2.3. C&J shift-reduce parser. (Reproduced from [Chelba and Jelinek, 2000, Fig. 7].)

Chelba (C&J)

C&J's model (called 'structured language model' by its authors, a name that is probably too generic in this text) is conceived as a generalization of a trigram. Consider predicting the word *after* in the following sentence:

> *the contract ended with a loss of 7 cents after trading as low as 9 cents .*

The trigram would predict *after* from *7* and *cents*. These are probably not the two words that contain most information; in this case *contract* and *ended* seem to be more predictive. C&J observed that, at least in English, the two words from a sentence prefix that are most predictive can often be found as the two most recent 'exposed headwords' in the partial parse generated by a probabilistic shift-reduce parser. In order to explain the concept of exposed headwords, it is necessary to see how partial parses look like. Therefore a more detailed description of the probabilistic shift-reduce parser is inserted here.

The shift-reduce parser is described as a probabilistic automaton that has three states (predictor, tagger, constructor), as shown in Fig. 2.3. It starts in the predictor state, predicts a next word (through PREDICT, essentially a SHIFT) and arrives in the tagger state. Then it must execute a TAG(X) operation, which predicts a part-of-speech label X for the word just predicted. In the constructor state, the parser can repeatedly choose from either ADJOINLEFT or ADJOINRIGHT, or return to the predictor state with a NULL operation. The ADJOINLEFT(Y) operation combines (i.e., REDUCEs) the two most recent unconnected trees to one tree with syntactic label Y, and percolates the headword of the left daughter tree to the new tree. For instance, in Fig. 2.4, if the parser had done an ADJOINLEFT(S) instead of a NULL and a PREDICT (of *after*), the result would have been one S/*contract* tree. The ADJOINRIGHT(S) would have produced a similar tree, but with headword *ended*. The new combined tree that results from an ADJOINRIGHT or ADJOINLEFT then becomes the most recent unconnected tree; the reduction process is repeated, unless the parser decides that the most recent unconnected tree should become a *left* daughter of another tree instead of a *right*

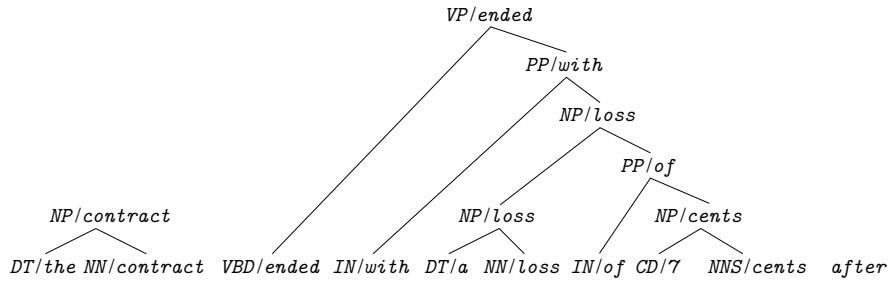

Figure 2.4. A partial parse in C&J. (Reproduced from [Chelba and Jelinek, 2000, Fig. 1].)

daughter. In the latter case it will return to the predictor state by taking a NULL transition.

Let us now return to the example given above. At the moment that the parser predicts *after*, one can now see that the most probable — or at least, most straightforward to a human reader — partial parse tree consists of two trees, the first covering *the contract* with headword *contract* and the second covering *ended with a loss of 7 cents* with headword *ended* (Fig. 2.4). These trees are not yet subtrees of another tree, which explains the term *exposed* headwords.

Generating the partial parse trees is ambiguous. In the example, another partial parse of the same sentence prefix is obtained by an ADJOINLEFT(*NP*) before the NULL transition that precedes PREDICT(*after*), leaving only *NP*/*contract* as an exposed headword.

The language model tracks and grows all probable partial parse trees up to the point that the next word is to be predicted; at that moment, the probability of the next word output by the language model, is computed as a weighted average of PREDICT probabilities, where the weights are the probabilities of the corresponding partial parses.

Due to the considerable ambiguity in parsing, the language model needs to discard less probable sequences early in order to keep computing time within reasonable limits. In order to get an idea of this ambiguity, it may be instructive to look at the number of distinct partial parse tree contexts considered by the language model at predicting the next word. Fig. 4.6 displays this quantity for the PLCG-based language model on page 91; 100 different partial tree contexts at predicting the 10th word in a sentence is typical. Chelba did not report similar figures, but I expect his model to behave similarly.

Roark

Roark's language model [Roark, 2001] is based on a stochastic version of a left-to-right top-down PCFG parser described by Aho et al. [1986].

The model builds *candidate analyses* $C = (d, P(d), F(d), \beta, w_i^n)$ consisting of a left-most partial derivation d (explained in Sec. 3.2.1), its figure-of-merit $F(d)$ (explained

next) and its probability $P(d)$, a sequence β of non-terminal symbols remaining to be rewritten, and the sentence suffix w_i^n remaining to be generated.
The figure-of-merit $F(d)$ is defined as the product of $P(d)$ and a so-called *look-ahead probability* $\mathrm{LAP}(\beta, w_i)$, which is the likelihood that β rewrites as some word string starting with w_i. Through $F(d)$, the look-ahead word functions as a soft bottom-up filter on the extension of partial parses; a candidate analysis is discarded if its figure-of-merit falls below a certain offset from the best figure-of-merit in the same priority queue. The offset is dynamically adapted in order to control the number of candidate analyses in the same priority queue.
$F(d)$ is used as the score fuction in a synchronous beam search: candidate analyses are organized in priority queues $\mathscr{H}_0, \ldots, \mathscr{H}_{n+1}$, where $\mathscr{H}_i$ contains all surviving candidate analyses that use w_i as the look-ahead word. Candidate analyses in H_i are extended by the parser in parallel by repeated PCFG rule productions, until ultimately the look-ahead word w_i itself is produced. If C' is a candidate analysis created from $C \in \mathscr{H}_i$ by producing w_i, then C' is placed in H_{i+1}, as well as in another queue $\mathscr{H}_{i+1}^{\text{init}}$. The sentence prefix probability $P(w_0^i)$ is available as

$$P(w_0^i) = \sum_{d \in \mathscr{H}_{i+1}^{\text{init}}} P(d) \tag{2.1}$$

and consequently conditional language model probabilities are obtained as

$$P(w_i | w_0^{i-1}) = \frac{\sum_{d \in \mathscr{H}_{i+1}^{\text{init}}} P(d)}{\sum_{d \in \mathscr{H}_i^{\text{init}}} P(d)}. \tag{2.2}$$

An interesting aspect of the parser is the fact that rewrite probabilities can be conditioned using a completely connected partial parse tree; exactly *which* features from a partial parse tree are used, is determined with a hand-tuned decision tree [Roark, 2001, Fig. 4].

Charniak

Charniak's language model, described in [Charniak, 2001], is a modified version of his 'maximum-entropy inspired parser' [Charniak, 2000], which is regarded as one of the best large-scale statistical parsers published so far. This parser needs two passes: in a first pass, candidate parse trees are generated with a bottom-up best-first probabilistic chart parser [Charniak et al., 1998]; in the second pass, these candidate parse trees are rescored from the top down using extensive probabilistic top-down conditioning. Daughter nodes are rescored only after their mother node has been rescored, and in the following order:

1. head daughter node;
2. daughter nodes left to the head daughter, from right to left — this is a 2nd order Markov process;

Table 2.1. Test set perplexities (PPL) (Penn Treebank section 23–24 test set) of a baseline trigram and various grammar-based language models. See also Table 4.6 on page 94.

Model	PPL stand-alone	PPL interpolated with trigram
trigram	167	167
C&J	141	130
Roark	152	137
Charniak	130	126
PLCG	133	126

3. daughter nodes right to the head daughter, from left to right — this is a 2nd order Markov process as well.

The detail of the probabilistic conditioning makes some clever smoothing necessary. Charniak proposed a smoothing method inspired on maximum-entropy models. However, the resulting smoothed predictors are probabilistically improper. For parsing, where probability is only used for ranking candidate parse trees, this is not really a problem. Charniak [2001] shortly noted that some modifications to the smoothing scheme had to be made for statistical language modeling, but he gave no precise description of these modifications nor a probabilistic analysis.

Assuming that Charniak's language model is probabilistically sound, its excellent performance in terms of test set perplexity is remarkable, as compared with C&J's and Roark's models (Table 2.1). On the other hand, Charniak's model cannot be used as a left-to-right conditional language model, which excludes interpolation with other language models on the word level.

The probabilistic left-corner parsing LM

The main achievement and the subject of my thesis is the development of the PLCG-based language model, a grammar-based language model that is more efficient than the other cited models while offering at least the same flexibility and modeling performance in terms of word perplexity and word recognition accuracy. Efficiency is primarily needed to make training on large corpora feasible (comparable to sizes of corpora to train 3- and 4-grams, for example).

This was obtained with a statistical left-corner parsing scheme in combination with dynamic programming (DP). DP could be used in combination with non-local probabilistic conditioning by selecting features at fixed relative positions in the partial parse tree — like [Chelba and Jelinek, 1999] but unlike [Roark, 2001] — and incorporating the non-local features in a state network representation.

The grammatical framework is identical to C&J's, Roark's and Charniak's: a simple lexicalized phrase structure grammar. This has two reasons:

1. Comparability: evaluation of language models is necessarily empirical, and small differences in experimental conditions may significantly influence the outcomes.

2. Availability of training data: the framework of the Penn Treebank is a simple phrase structure grammar. There are no other treebanks available of a comparable size and parsed with a more detailed grammar, such as a UBG.

Refining the grammar, while maintaining the large-scale scope of the language model, is a topic for future research.

2.6. Conclusion

The strength of grammar-based language models relies on their generalization power derived from recognizing hidden syntactic structural patterns. The area of grammar-based language models has seen an extension in various (related) ways. The PLCG-based model is a logical continuation of this evolution.
From the formal aspect, the early language models were nothing more than network-compiled finite-state grammars and context-free grammar parsers. Several provisions in the grammar or the parser have been introduced since: on the one hand, concepts and extensions, such as unification of syntactic and lexical features, were adopted from more modern computational linguistics; on the other hand, dependency and link grammars, as well as history-based grammars stressed the importance of non-local probabilistic conditioning, particularly on lexical information.
As to parsing techniques, not so much has changed: variants of the popular CKY, Earley (with or without lookahead), LR and Left Corner parsing algorithms are being used in the latest grammar-based language models, although the introduction of probability has slightly shifted the attention to other issues. For instance, cycles in a CFG cause a naive parser to enter an infinite loop; this problem is mitigated by a PCFG, because each additional run through a cycle will render a parse more improbable (as it should) — although some factorization techniques may remain useful for efficiency. However, probabilistic grammars are typically large — 'nothing is impossible'. A probabilistic parsing algorithm for a language model should scale well with sentence length and grammar size. This necessitates intelligent pruning and an effective search technique, such as beam search with or without dynamic programming. Dynamic programming (DP) restructures the search space of a parser (a tree) in a more compact network, thus allows a faster evaluation of equally many derivations. DP is commonly associated with context-free chart parsing, but this thesis shows how it can be applied to left-corner parsing in the PLCG-based language model.
While the first grammar-based language models were integrated in applications with a small vocabulary and restricted language use, recent grammar-based language models can improve on word-based trigrams and 4-grams in large-vocabulary speech recognition applications such as dictation and transcription of read newspaper articles. This

coincides with a shift from small hand-written intelligent grammars to a more corpus-based learning of the grammar, in both supervised and unsupervised modes. The resulting grammars tend to be large and unsophisticated, but are sufficiently accurate to do automatic parsing, due to probabilistic conditioning.

The PLCG-based language model uses a simple phrase-structure grammar extracted from the Penn Treebank corpus, which allows a fair empirical evaluation against other competing grammar-based language models. More detailed (for instance, unification-based) grammars are not yet within the reach of large-vocabulary language modeling, because of the lack of large copora parsed with more detailed grammars.

As grammar-based language models grow more complex, they become more difficult to tightly integrate in the search engine of traditional speech decoders. The solution of this issue is beyond the scope of this thesis; it may involve a rethinking of the complete search engine.

CHAPTER 3

A language model based on probabilistic left corner parsing

Traditional language modeling techniques, described in Chapter 1, do not use prior knowledge of grammatical structure. In this chapter, my aim is to improve language modeling for statistical large-vocabulary continuous speech recognition LVCSR) by *grammatically structuring* the language model probabilities. In Sec. 2.5.4, a number of recent grammar-based language models, applicable in LVCSR, were presented. This chapter describes a novel grammar-based language model using left-corner parsing, that was found to be a competitive alternative to the other grammar-based language models. The description of experiments is deferred to the next chapter.

3.1. Problem formulation and methods

This chapter proposes a novel language model, which is based on probabilistic left corner parsing. It is an answer to the principal question of this thesis: in which way can simple prior knowledge about grammatical structure contribute to better statistical language models?

Statistical language models actually model the probabilistic generation of a sentence; therefore it is straightforward to start from a generative view on grammatical parsing: parsing a sentence is, in fact, understood as generating the grammatical structure that yields the sentence to be analysed.

The first step of the solution consists of the direct estimation of the probabilities of the elementary parsing moves (i.e. generation moves) from a treebank, which is a large set of pre-analysed sentences. This way, one avoids the manual specification of grammatical prior knowledge in the form of a limited set of non-probabilistic rules. This data-driven approach appeared successful in recent, comparable research, especially for large-vocabulary speech recognition.

The next question is, how the elementary parsing moves should be defined, which parsing strategy is optimal. I chose left corner parsing, which tightly couples expectations about the global structure of the sentence with partial analyses of the sentence itself (Sec. 3.3). This parsing strategy avoids the inefficiency that stems from building partial analyses of the sentence which afterwards turn out to be incompatible with the global structure, or building global structures that afterwards turn out incompatible with the actual sentence. A PLCG (probabilistic left corner grammar) is defined as the ensemble of the treebank-derived probabilities of the elementary left corner parsing moves.

The strength of the data-driven approach lies in the possibility to condition the move probabilities on very detailed features of partial analyses. However, this inflates the number of possible (partial) analyses of the sentence considerably; hence, the efficiency of the parsing algorithm, which accumulates the probabilities of all these analyses, is crucial. To this purpose I developed a recursive computation strategy that is based on the principle of dynamic programming (Sec. 3.4). The resulting parsing algorithm can be adapted to a conditional language model — the PLCG-based language model — virtually without extra cost (Sec. 3.5). It also facilitates reestimating PLCG-probabilities from plain, unparsed training text.

Inefficiency of other approaches

My aim is to combine the strengths of C&J's and Roark's models using stochastic left corner parsing.

The C&J model is inefficient in both memory and time, which restricts its use to rescoring n-best lists or very thin word lattices. Perhaps more importantly, available computational resources severely limit the amount of training text used to reestimate the model. There are two sources of inefficiency:

1. *Following paths leading to improbable derivations.* C&J's parser grows isolated subtrees from the bottom up without (stochastic) top-down filtering. In the equivalent derivation tree representation, too many paths are extended that lead to improbable paths in the end; they should have been pruned in an earlier stage.

2. *Searching a tree-shaped derivation space.* The derivation space is tree-structured. Hence identical paths occurring at different places in the tree are treated separately. Efficiency could be gained by a representation that avoids repetition of identical paths. A dynamic programming approach for C&J was presented by Jelinek and Chelba [1999], but this text did not contain empirical results. It was later improved and implemented by Van Aelten and Hogenhout [2000].

Roark's top-down parsing LM [Roark, 2001] does not suffer from inefficiency 1.: parse trees are grown from the top down with look-ahead to the next word. So it can decide in an earlier stage which rules should not be expanded. However, the *rule probabilities themselves* cannot be conditioned on the lookahead-word, since that would violate the chosen chain-rule decomposition of the derivation probability; there-

fore aggressive pruning of partial paths by comparison of their probabilities is likely to introduce too many search errors. A disadvantage of Roark's model w.r.t. C&J's model is that next-word probabilities are systematically underestimated, while C&J are able to renormalize them.
Roark's derivation space representation is similar to C&J's, sharing the same inefficiency. Besides, this representation makes it difficult — if not impossible — to find an accurate EM reestimation algorithm.[1]
On the other hand the tree representation of derivations gives Roark's model flexibility in extracting conditioning information from the path prefix using 'tree-walking' functions.

3.2. Definitions and notation

3.2.1. Stochastic grammars and parsing

A **context-free grammar** (CFG) Γ is a 4-tuple (N,V,P,S), where N is a finite set of non-terminal category labels, V is a finite set of terminal category labels (the vocabulary), $P = \{(A \rightarrow \alpha) | A \in N, \alpha \in (N \cup V)^*\}$ is a finite set of context-free production rules and $S \in N$ is the axiom or start symbol. Latin capitals $(A,B,C,\ldots)$ represent non-terminals, Latin lowercase letters $(a,b,c,\ldots)$ represent terminals, while lowercase Greek letters $(\alpha,\beta,\gamma,\ldots)$ represent *sequences* of terminals and/or non-terminals.
The following discussion also needs the **left corner relation**. Given a CFG $\Gamma = (N,V,P,S)$, X is a left corner of Y if and only if there is some α for which there is a rule $Y \rightarrow X\alpha$ in P. It is written as $X \angle Y$. The reflexive and transitive closure of $\angle$ is written as $\angle^*$.
The **derives** relation, written $\overset{\Gamma}{\Rightarrow}$ or $\Rightarrow$ if Γ is clear from the context, is defined as follows: $X_0^n \Rightarrow X_0^{i-1}\alpha X_{i+1}^n$ if and only if there is a rule $(X_i \rightarrow \alpha) \in P$. A top-down derivation sequence can now be written as

$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow w_1^n.$$

The reflexive and transitive closure of $\overset{\Gamma}{\Rightarrow}$ is written as $\overset{\Gamma}{\Rightarrow}^*$. The proposition $S \overset{\Gamma}{\Rightarrow}^* w_1^n$ states that the word string w_1^n parses as a sentence with the grammar Γ.
Let a CFG be given. A **local tree** $t = (t_1^m)_X$ is either an empty sequence or a sequence of pointers to other local trees $t_1,\ldots,t_m$, labeled with a category $X \in (N \cup V)$. The function $R(t) = X$ returns the label of t, and by extension, $R(t_1^m) = R(t_1)\ldots R(t_m)$. Let $t \triangle_i t_i$ denote that t_i is the i-th daughter node of t. If $m \geq 1$, the local tree $t = (t_1^m)_X$ is only valid if there is a grammar rule $R(t) \rightarrow R(t_1^m)$. Otherwise t is a terminal node and $R(t)$ must be a terminal category.
A **parse tree** is a set of local trees in which there is exactly one local tree, called the **top** of T, that is not pointed to by the other local trees, and each other local tree

1. Chelba however found and used a working *approximative* EM reestimation algorithm [Chelba, 2000].

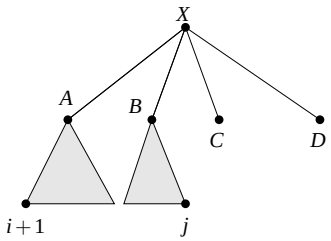

Figure 3.1. A graphical representation of a constituent $q = [_i AB_j CD]_X$.

is pointed to by exactly one other local tree. The **yield** of T, denoted by $Y(T)$, is the sequence of the labels of the terminal nodes in T, read in depth-first, left-to-right order. T is a **full** parse tree if $Y(T)$ contains terminals only; otherwise it is a **partial** parse tree.

A parse tree T is a representation of a top-down derivation sequence: consider the top-down derivation sequence $S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_p = w_1^n$, then each α_i corresponds with a parse tree T_i and each step $\alpha_i \Rightarrow \alpha_{i+1}$ corresponds with expanding one leaf node of T_i resulting into T_{i+1} such that $Y(T_i) = \alpha_i$.

Note that there are multiple top-down derivations corresponding with one parse tree, since the order in which non-terminals are rewritten is arbitrary. The **leftmost derivation** is the top-down derivation that always rewrites the *leftmost* non-terminal (the corresponding relation is written $\Rightarrow_L$). There is a unique correspondence between a parse tree and its leftmost derivation.

The concept of a **constituent** links a local tree with its yield. It can be understood as a local tree of which the yield is partially or completely verified to correspond with a part of the input sentence. In this text, a **constituent** $q = [_i \alpha_j \beta]_X$ of a parse tree T for which $Y(T) = w_1^n$, is equivalent with the proposition that there is a local tree $(t_1^m)_X \in T$ for which $R(t_1^k) = \alpha$, $R(t_{k+1}^m) = \beta$ and $Y(t_1) \cdots Y(t_k) = w_{i+1}^j$. Fig. 3.1 gives a graphical representation of a constituent $q = [_i AB_j CD]_X$.

A constituent q is said to be a constituent of w_1^n if there is at least 1 parse tree T where $Y(T) = w_1^n$ and q is a constituent of T. If $\beta = \varepsilon$, q is called **resolved**. Otherwise q is **unresolved**.

For notational convenience, I define two functions $\mathrm{pos}(q)$ and $\mathrm{cat}(q)$, returning the position and the category of q, respectively. For instance, if $q = [_i \alpha_j \beta]_X$, then $\mathrm{pos}(q) = j$ and $\mathrm{cat}(q) = X$.

3.2.2. Parsing and language models

In natural language grammars, the generation of a sentence is initiated by a start symbol, usually called S. The sentence is complete as soon as a parse tree is found with S as the top category and the input sentence as its yield. Language modeling how-

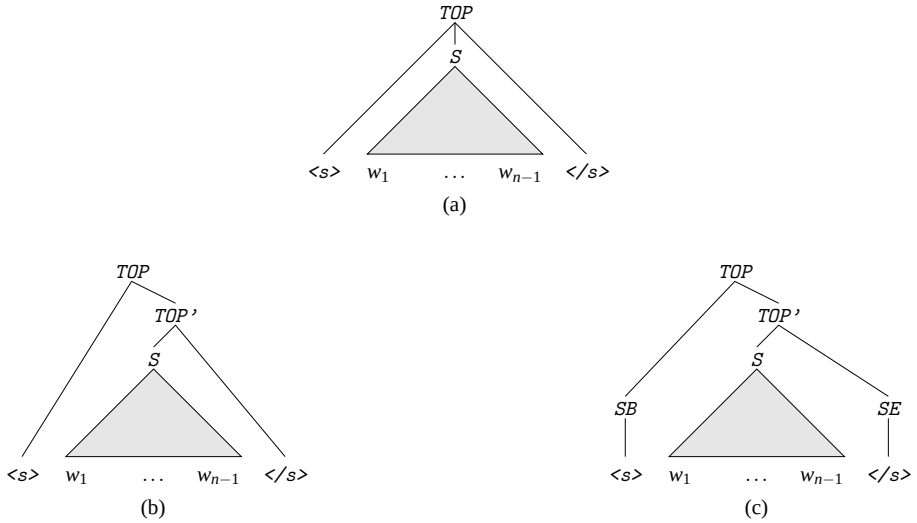

Figure 3.2. Marking sentence boundaries in parsing-based language modeling.

ever, traditionally initiates a sentence with a start-of-sentence symbol, e.g. <s>, and completes it with the emission of an end-of-sentence symbol, e.g. </s>.
These two different views can be made compatible by making the original local parse tree $t_S = (\ldots)_S$ a daughter of a local tree $(t_I, t_S, t_M)_{TOP}$ (cf. Fig. 3.2(a)). The yield of the resulting tree is w_0^n, where $w_0 =$ <s>, $w_n =$ </s> and the original input sentence is w_1^{n-1}. If the grammar or parser only handles binary trees, this can be accounted for with a TOP' tree as in Fig. 3.2(b). If the framework requires pre-terminals (e.g. parts-of-speech) between the input words and the actual parse tree, two pre-terminals SB (sentence begin) and SE (sentence end) can also be added, as in Fig. 3.2(c).

3.2.3. Push-down automata

I will use left corner push-down automata (PDA) to develop the concept of PLCG-based language modeling. The notation of PDA transitions follows Hopcroft et al. [2001, Ch. 6], except that the state from the instantaneous description is left out (cf. below) since the left corner PDAs are stateless. A **stateless PDA** is a 4-tuple $(V, \mathscr{S}, m, q_0)$ where V is a finite alphabet of input symbols, $\mathscr{S}$ is a finite alphabet of stack symbols, m is a transition function and q_0 is the initial stack symbol. The transition function m is defined by a set of rules that describe possible transitions from one instantaneous description to another one, formalized as a relation $\vdash$. An instantaneous description of the PDA is written as $\left(w_i^j, \pi\right)$, where w_i^j denotes the input sequence left to be consumed, and π denotes the contents of the stack. The stack is written as a sequence with the stack top at its left end and the stack bottom at its right end. The

operation

$$\left(w_i^j, \alpha\pi\right) \vdash \left(w_{i+1}^j, \beta\pi\right)$$

consumes the next input symbol (w_i) and replaces α on the stack with β. Note that w_i can be empty, in which case the operation does not consume input. Also α and/or β can be empty.

3.3. Probabilistic left corner parsing

This section gradually develops the concept of a probabilistic left corner grammar (PLCG). After a brief review of left corner derivation and introduction of the left corner automaton, its stochastic generalization is described, followed by its extension with lexicalized categories and extensive probabilistic conditioning of the move probabilities. Finally the concept of PLCG submodels is introduced and the PLCG is defined.

3.3.1. Non-probabilistic left corner parsing

Non-probabilistic left corner parsing is known as an efficient CFG parsing technique. It is often attributed to Rosenkrantz and Lewis II [1970], although a similar idea was already used by the SBT parser [Griffiths and Petrick, 1965]. Several enhancements and extensions of the original method were proposed later [Matsumoto et al., 1983, Wirén, 1987, Nederhof, 1993, Moore, 2000].

In Sec. 3.2.1 we saw that a parse tree uniquely corresponds with its leftmost derivation. However, there are many other canonical derivation schemes, *left corner derivation* being one of them. In left corner derivation, the generation of a full parse tree of a given category is initiated by predicting (*shifting*) the next word token (the leaf node at its bottom left), which is considered a (trivial) full parse tree. A full parse tree can be used for *projection* or be *attached* to another partial parse tree. Projecting from a full parse tree means constructing a local tree on top of the full parse tree, such that the full parse tree is the leftmost daughter node of the local tree; the other daughter nodes have a label but no daughters. The resulting partial tree is completed by 'plugging in' (*attaching*) other full parse trees of the appropriate categories at the daughter nodes. These full parse trees have each been generated by recursive application of this strategy.

Left corner derivation is a very plausible hypothesis of how humans predict next words. Projection models the expectation of the type of following constituents, based on the type of constituents already observed, and on hypothesized global structure; attachment reflects the confirmation that a certain projection is correct; and shifting corresponds with listening to the next word, and ruling out certain expectations about the sentence structure that conflict with it. For instance, consider the sentence prefix

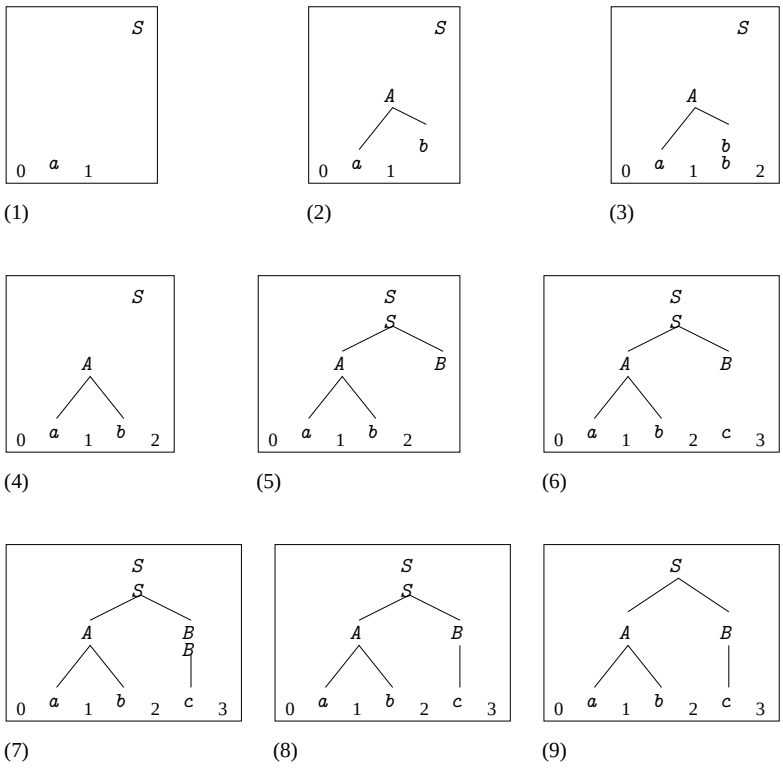

Figure 3.3. Left corner derivation of a small parse tree.

<s> analysts expect sharp ...

A human reader would believe it reasonably probable that a noun follows. Why? Probably because the adjective *sharp* signals the start of an *NP*; the construction of the *NP* seems likely because it would complete the verb *expect* to form a *VP*, which would complete a *S* together with the *NP analysts*. Each of these expectations are actually modeled as projections: a *VP* that completes a *S* is projected from the *NP analysts*, an *NP* that completes a *VP* is projected from *expect*, a noun or noun phrase that completes an *NP* is projected from *sharp*. Each projection is influenced by previous projections. If, instead, the sentence prefix had been

<s> analysts say sharp ...

then one can imagine that the verb *say* projects a sentence rather than a noun phrase. The projection of a noun (phrase) that would complete an *NP* from *sharp*, is provoked by the expectation of a sentence, and not by an expectation of a *VP*, as in the previous example.

To get a more formal feel for left corner derivation, I detail the left corner derivation steps of a small parse tree displayed in Fig. 3.3 (this time in terms of constituents instead of parse trees):

1. Start from $[_0\, \varepsilon_0\, S]$ and shift a: obtain $[_0\, a_1\,]_a$.
2. Project $[_0\, a_1\, b]_A$.
3. Shift b: obtain $[_1\, b_2\,]_b$.
4. Attach $[_1\, b_2\,]_b$ to $[_0\, a_1\, b]_A$: obtain $[_0\, ab_2\,]_A$.
5. Project $[_0\, A_2\, B]_S$.
6. Shift c, and obtain $[_2\, c_3\,]_c$.
7. Project $[_2\, c_3\,]_B$.
8. Attach $[_2\, c_3\,]_B$ to $[_0\, A_2\, B]_S$: obtain $[_0\, AB_3\,]_S$.
9. Attach $[_0\, AB_3\,]_S$ to $[_0\, \varepsilon_0\, S]$: obtain $[_0\, S_3\,]$, which completes the derivation.

A left corner parser can be elegantly and concisely described as a non-deterministic stateless PDA $(V, \mathscr{S}, m, q_I)$, where

1. V is the set of terminals (a vocabulary consisting of word types);
2. $\mathscr{S}$ is the set of constituents;
3. $q_I = [_0\, \varepsilon_0\, S]$ is the initial stack element;
4. m is defined by the following operations:

   a) The SHIFT operation:

$$\left(w_{j+1}^n, ([_i\, \alpha_j\, Y\, \beta]_X, \pi)\right) \quad \vdash \quad \left(w_{j+2}^n, ([_j\, w_{j+1}\ _{j+1}\,]_{w_{j+1}}, [_i\, \alpha_j\, Y\, \beta]_X, \pi)\right) \tag{3.1}$$

   for any $i, j, \alpha, \beta, X, Y$ and π (which denotes the rest of the stack). In words, the shift operation takes the next word w_{j+1}, builds a new constituent with it and pushes it on the stack. It is applicable whenever the stack top is an unresolved constituent.

   b) A number of PROJECT(U, δ) operations:

$$\left(w_{k+1}^n, ([_j\, \alpha_k\,]_X, [_i\, \beta_j\, Y\, \gamma]_Z, \pi)\right) \quad \vdash \quad \left(w_{k+1}^n, ([_j\, X_k\, \delta]_U, [_i\, \beta_j\, Y\, \gamma]_Z, \pi)\right) \tag{3.2}$$

   for any $i, j, k, \alpha, \beta, \gamma, X, Y, Z$ and π. In words, a PROJECT(U, δ) operation consumes no input and replaces the stack top with a new constituent. The stack top must be a resolved constituent and figures as the leftmost daughter of the new constituent.

   c) The ATTACH operation:

$$\left(w_{k+1}^n, ([_j\, \alpha_k\,]_X, [_i\, \beta_j\, X\, \gamma]_Z, \pi)\right) \quad \vdash \quad \left(w_{k+1}^n, ([_i\, \beta X_k\, \gamma]_Z, \pi)\right) \tag{3.3}$$

for any $i, j, k, \alpha, \beta, \gamma, X, Z$, and π. In words, the ATTACH operation is applicable if the category of the stack top X equals the first unresolved daughter of the constituent below it. The stack top is popped, and the constituent below it is modified by marking the daughter X as resolved.

d) The PDA terminates successfully by empty stack if all input is consumed and the stack only contains the final constituent $q_F = [_0 S_n]$:

$$(\varepsilon, ([_0 S_n])) \quad \vdash \quad (\varepsilon, \varepsilon) \tag{3.4}$$

A deterministic left corner parser can be constructed from the PDA described above with a general method described by Lang [1974].
Note that dead paths can be abandoned early by checking whether $w \angle^* Y$ when shifting w (cf. (3.1)), and whether $U \angle^* Y$ when projecting U in order to ultimately obtain a constituent of category Y (cf. (3.2)). These checks constitute **top-down filtering**, and can be implemented as a simple look-up in a pre-compiled table.

3.3.2. Probabilistic left corner parsing: previous art

In a probabilistic CFG (PCFG), each production rule $A \rightarrow \alpha$ is annotated with the conditional probability $P(\alpha|A)$ that the given lefthand side A produces the righthand side α. Although this probability is successfully handled in various PCFG parsers, its definition is most logical from the viewpoint of top-down derivation. Starting from a more general view on derivation, conceptually simpler probabilistic parsers are obtained as follows.
Consider a derivation m_1^p, which is a sequence of p moves $m_1, m_2, \ldots, m_p$. Then $P(m_1^p) = \prod_{i=1}^{p} P(m_i|m_1^{i-1})$. $P(m_1^p)$ is a **derivation probability** or **path probability** and $P(m_i|m_1^{i-1})$ a **move probability** or **transition probability**. The space of derivations can be organized as a prefix tree in which a path from the root to one of the leaves represents one derivation. Simple probabilistic parsing is realized by searching the most probable path (or all 'sufficiently' probable paths) in the derivation prefix tree.
If a derivation uniquely corresponds with a parse tree, then the probability of the parse tree can be computed as the probability of its derivation and $P(m_i|m_1^{i-1})$ is equivalent with $P(m_i|t(m_1^{i-1}))$ where $t(m_1^{i-1})$ is the partial parse tree generated with m_1^{i-1}.
A *probabilistic* left corner automaton (PLCA) is now obtained straightforwardly by annotating the original stack rules with conditional move probabilities

$$P(m_i = \text{SHIFT}(w)|m_1^{i-1}),$$
$$P(m_i = \text{PROJECT}(U, \delta)|m_1^{i-1}), \text{ and}$$
$$P(m_i = \text{ATTACH}|m_1^{i-1}),$$

where m_1^{i-1} denote the preceding moves. If one is only interested in the best parse tree of a given sentence, there is no need to treat the SHIFT(w) move probabilistically

where w is the next word to be shifted, since it is possible when and only when the top of the stack is unresolved. However, in order to obtain an estimate of the probability of the input sentence (instead of a parse tree) the SHIFT(w) move has to be predicted probabilistically. Of course, the probabilities of all next moves, given the preceding ones, should add up to 1:

$$\sum_{w \in V} P(\text{SHIFT}(w) | m_1^{i-1}) + \sum_{\substack{U \in N \\ \delta \in (N \cup V)^*}} P(\text{PROJECT}(U, \delta) | m_1^{i-1}) + P(\text{ATTACH} | m_1^{i-1}) = 1. \tag{3.5}$$

Previously described PLCAs can be recognized as simplifications of the above general description. For example, the PLCA described by Manning and Carpenter [1997] can be regarded as the result of three simplification steps of m_1^{i-1}:

1. The derivation prefix m_1^{i-1} is replaced by the stack it produces.

2. The prediction of a SHIFT(w) is conditioned only on the top of the stack, while a PROJECT(U, δ) and an ATTACH are conditioned only on the top and the subtop.

3. Referring to the Eqs. 3.1–3.3, the SHIFT(w) move is conditioned only on Y. The PROJECT(U, δ) move is conditioned only on X and Y. The ATTACH move is conditioned only on X.

These simplifications implement a context clustering function. They imply assumption of probabilistic independence from all information that is contained in the stack but not captured by the context clustering function. This assumption is, of course, at least partly invalid, but it allows estimation of the conditional move probabilities from a limited number of training examples.

3.3.3. Probabilistic left corner parsing: extension

The standard PLCA described in the previous section is now extended in two ways.

Constituent context

Model accuracy is improved by weakening the independence assumptions on the move probabilities, since they are false. At the same time however, it is my intention to simplify the PLCA rules Eqs. 3.1–3.3 to rules that only specify the top constituent on the stack, which will allow easy model reestimation in the spirit of the Baum-Welch algorithm. Both goals are realized by integrating all conditioning features in the top stack element:

- A **local tree context** $\vec{g}$ is defined as a set of features of a parse tree with respect to a local tree in that parse tree.

- The definition of a constituent is extended with a local tree context. From now on, a constituent $x = [_i \alpha_j \beta | \vec{g}]_X$ is defined as the set of parse trees that contain a local tree $t = (t_1^m)_X$ for which $R(t_1^m) = \alpha\beta$ and $\alpha \Rightarrow^* w_{i+1}^j$ *and this local tree has a local tree context* $\vec{g}$.

For a local tree context to be useful in parsing, it should be recognizable from the part of the parse tree that is already constructed at the time the local tree which it conditions is being constructed.

The local tree context used in the C&J model consists of the categories of the two rightmost isolated trees (i.e., they are not yet a subtree of another tree). An alternative definition is given here, based on the full parse tree (see Fig. 3.4(a)), because it will show the equivalence of C&J with the local tree context in the PLCA-based model.
The local tree context of the PLCA-based model consists of 3 features $(g_1, g_2, g_3) = \vec{g}$. Going up the tree from a given local tree t, g_1 is found as the category of the first node, say r, that is not a leftmost daughter. The category of the leftmost sister of r, say s, is g_2. Again going up the tree from s until the first node, say r', that is not a leftmost daughter, s' is found as the leftmost sister of r', and g_3 is found as the category of s'.
More formally, the local tree context feature g_2 is found by calling $t = r_0$ and considering the sequence $(r_0 = t, r_1, ..., r_p)$ for which $r_i \triangle_1 r_{i-1}$ for $i = 1 \ldots p-1$ but $r_p \triangle_q r_{p-1}$ with $q > 1$. If $r_p \triangle_1 s$, then $g_2 = R(s)$. g_3 is defined as the g_2 context feature of s: $g_3 = R(s')$, where s' is found in the same way from s as s was found from r_0. Finallly g_1 is defined as $R(r_{p-1})$. As shown in Fig. 3.4(c), r_{p-1} is not yet available when constructing r_0; however, its label $R(r_{p-1})$ has already been predicted by the projection of r_p from s.
Fig. 3.4(b) shows the equivalence with the probabilistic conditioning of the shift-reduce parsing mechanism used by the C&J model. This parser creates binary parse trees only. The part of the parse tree that has not yet been predicted, at the time r_0 is being constructed, is drawn with dashed lines. Only s and s' are available, there is no such node as r_{p-1} yet. This leads to an essential difference with the PLCA-based model: there is no context feature g_1. (In the original C&J model, $g_3 = R(s')$ and $g_2 = R(s)$ condition the construction of r_0 only if r_0 is a leaf node (shift move). C&J's reduce steps are independent from s', presumably for practical reasons, but it is theoretically possible to include s' too.)
Now follows a redefinition of the PLCA with context-conditioned constituents as stack elements. Given an input sentence w_0^n:

- The stack is initialized with (cf. Fig. 3.2(c))

  $$q_I = [_0 \, \mathit{SB}_1 \, \mathit{TOP'} | \mathit{TOP}, \mathit{SB}, \mathit{SB}]_{\mathit{TOP}}$$

  . The second and third element of the context do not correspond with the local tree context definition; they are just used to avoid additional symbols.

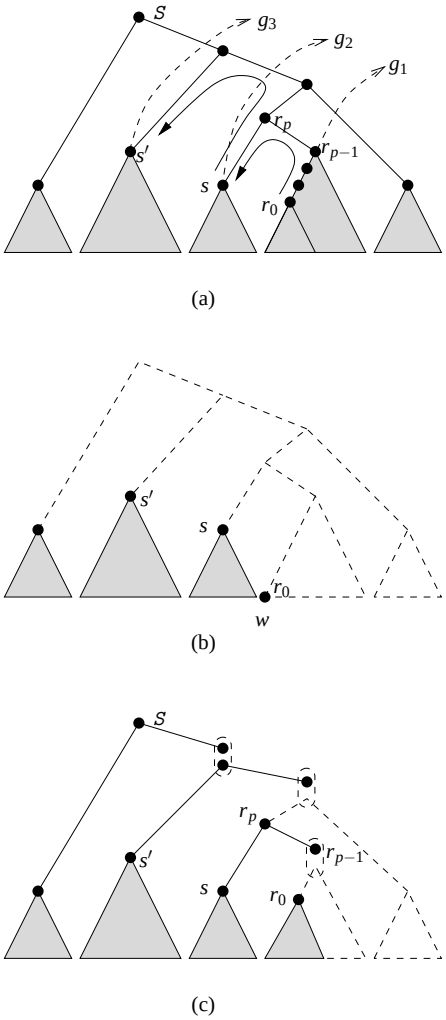

Figure 3.4. (g_1, g_2, g_3) context of a local tree $t = r_0$. (a) Definition of g_1, g_2 and g_3 on a full parse tree. (b) In the C&J model: at the time w is shifted, only s and s' are known. (c) In the PLCA model: at the time r_0 is constructed, s and s' are available; r_{p-1} is not yet known, but $R(r_{p-1})$ is. (Dashed lines indicate parts of the parse tree that await completion. The dashed boxes indicate pending attach moves.)

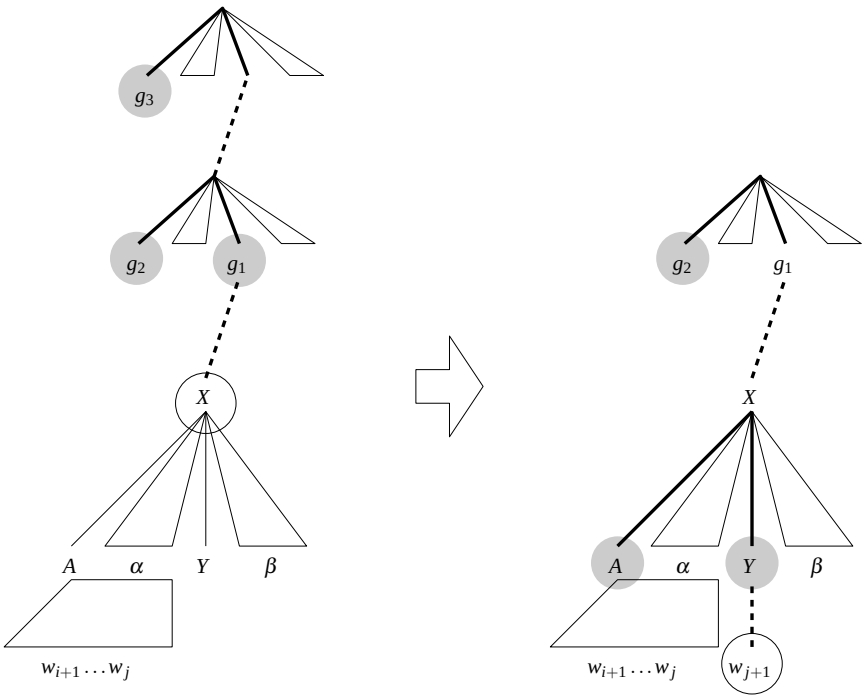

Figure 3.5. Context inheritance after a SHIFT.

- The stack is emptied and the PLCA terminates successfully if it only contains (cf. Fig. 3.2(c))

$$q_F = [_0\, \mathit{SB} \quad \mathit{TOP'}_n \,|\, \mathit{TOP}, \mathit{SB}, \mathit{SB}]_{\mathit{TOP}}.$$

- The original SHIFT operation (3.1) is modified as:

$$\left(w_{j+1}^n, ([_i A\alpha\ _j Y\beta|\vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{j+2}^n, ([_j w_{j+1}\ _{j+1}|\vec{h}]_{w_{j+1}}, [_i A\alpha\ _j Y\beta|\vec{g}]_X, \pi)\right),$$

  where $\vec{g} = (g_1, g_2, g_3)$ and $\vec{h} = (Y, A, g_2)$. The construction of $\vec{h}$ can be understood by looking at Fig. 3.4(c): a new constituent, $[_j w_{j+1}\ _{j+1}|\vec{h}]_{w_{j+1}}$, corresponding with r_0, is created and pushed on the stack when an unresolved constituent, $[_i A\alpha\ _j Y\beta|\vec{g}]_X$, corresponding with r_p, was encountered on top of the stack. The first unresolved daughter constituent of r_p is h_1 and equals Y. h_2 is found as the category of the leftmost daughter of r_p, which is A. The g_2 context feature is inherited as h_3: it corresponds with the category of s' on the same figure. Fig. 3.5 is a more literal graphical interpretation of the above formulation of SHIFT.

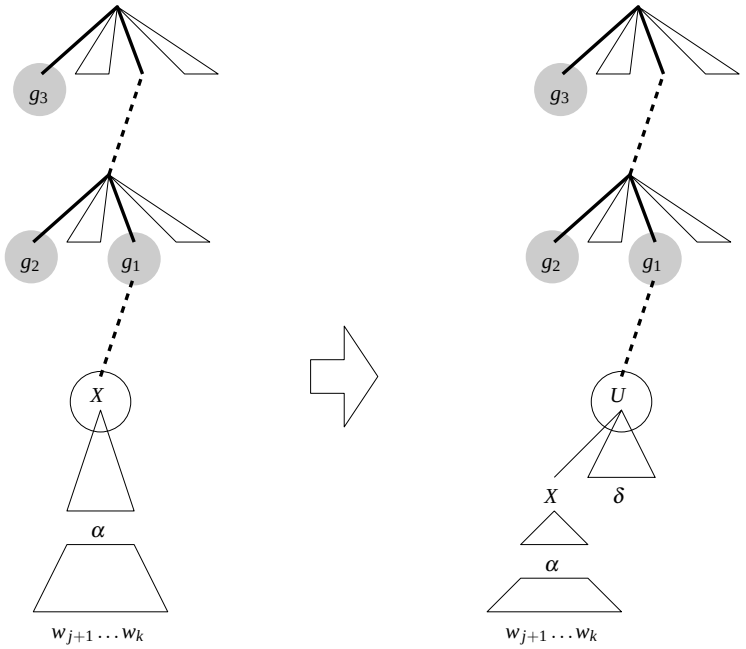

Figure 3.6. Context inheritance after a PROJECT.

- The original form of a PROJECT(U, δ) move (3.2) becomes:

$$\left(w_{k+1}^n, ([_j\, \alpha_k\, |\vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{k+1}^n, ([_j\, X_k\, \delta |\vec{g}]_U, \pi)\right),$$

  where $\vec{g} = (Y, B, h_2)$. Again referring to Fig. 3.4(c), if $[_i\, \alpha_j\, |\vec{g}]_X$ would correspond with r_0, then $[_i\, X_j\, \delta |\vec{g}]_U$ would correspond with r_1 and $p > 1$. Hence $\vec{g}$ is simply inherited. Fig. 3.6 shows a more literal graphical interpretation of the above formulation of PROJECT.

  Note that in case $X = Y$, the ATTACH operation is applicable too (see next).

- Finally, the original ATTACH operation (3.3) becomes:

$$\left(w_{k+1}^n, ([_j\, \gamma_k\, |\vec{h}]_Y, [_i A\alpha_j\, Y\beta |\vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{k+1}^n, ([_i A\alpha Y_k\, \beta |\vec{g}]_X, \pi)\right),$$

  where $\vec{h} = (Y, A, g_2)$. Fig. 3.7 shows a graphical interpretation of the above formulation of ATTACH.

  For convenience, attachment is considered as a special projection: ATTACH $\equiv$ PROJECT(ATT) and require $P(\text{PROJECT}(\mathit{ATT})|[_j\, \gamma_k\, |\vec{h}]_Y) = 0$ if $h_1 \neq Y$.

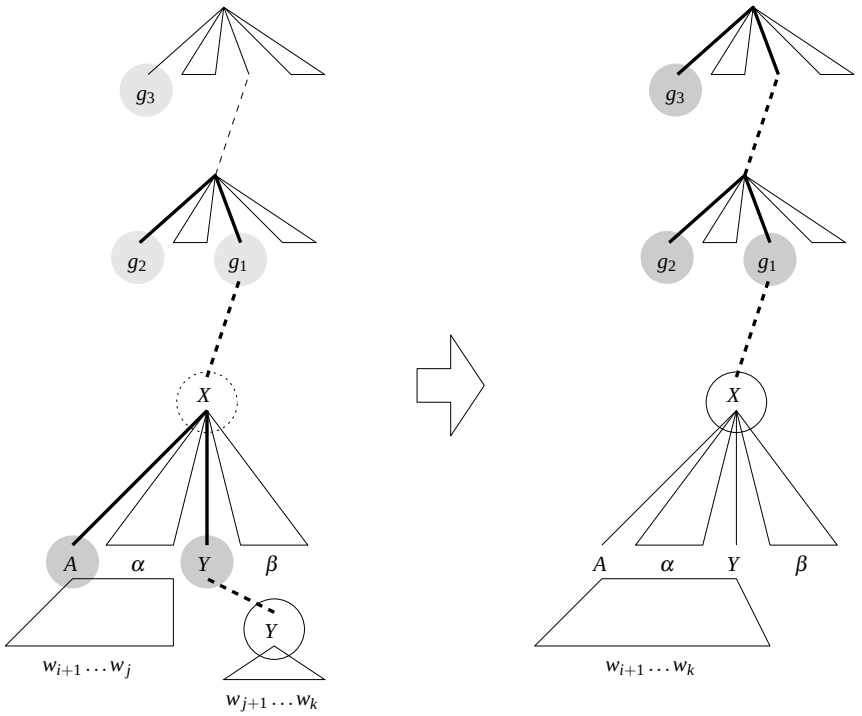

Figure 3.7. Context inheritance after an ATTACH.

Lexicalized categories

It is empirically found that syntactic categories, however fine-grained they are, lose important information that statistically influences certain structural pattern preferences. A common and simple approach is to keep record of a lexical feature besides a syntactic category feature, leading to **lexicalized** grammars. For instance, in a typical lexicalized grammar, the lexical feature of the noun phrase 'an evil monster' would be 'monster'.

The same principle is applied to the PLCA, and some more notation is needed for this purpose: underlined Latin capitals denote a composite category label consisting of a syntactic category and a lexical category separated with a slash (/); underlined Greek letters denote sequences of composite category labels. A terminal w is replaced with a composite $\underline{W}/w$ using a dummy $\underline{W}$ syntactic feature.

It is also necessary to specify how the lexical feature is determined: it is assumed that the lexical feature of the mother constituent can be determined as a function of its syntactic category and the composite categories of its daughters, and call this function head$(\cdot)$.[2] In the experiments described in the next chapter, it is even assumed that

2. This assumption seems sufficient for English, but it may be too restrictive for other languages. For

the lexical category of a constituent is equal to the lexical category of its daughter at head position, and that this head position can be precomputed for each grammar rule using a set of 'headword percolation' rules by [Magerman, 1994]. There is a headword percolation rule for each mother category. For instance, there is a rule that says: "if the mother category is an *NP*, then the first daughter, starting from the rightmost daughter, that has one of the following categories, is the head: *NNP*, *NNPS*, *NP*, *NN*, *NNS*, *NX*, *CD*, *QP*, *PRP*, or *VBG*."
Given the input sentence w_0^n, the format of the stack rules of the extended PLCA is then again adapted to the following final form.

- The stack is initialized with

$$q_I = [_0\, \mathit{SB/<s>}\,{}_1\, \mathit{TOP'} | \mathit{TOP}, \mathit{SB/<s>}, \mathit{SB/<s>}]_{\mathit{TOP}} \tag{3.6}$$

- The stack is emptied and the PLCA terminates successfully if it only contains

$$q_F = [_0\, \mathit{SB/<s>} \quad \mathit{TOP'}\,{}_n | \mathit{TOP}, \mathit{SB/<s>}, \mathit{SB/<s>}]_{\mathit{TOP}} \tag{3.7}$$

- A SHIFT(w_{j+1}) move applies

$$\left(w_{j+1}^n, ([_i \underline{A\alpha}\,{}_j\, Y\beta | \vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{j+2}^n, ([_j \underline{W}\,{}_{j+1} | \vec{h}]_W, [_i \underline{A\alpha}\,{}_j\, Y\beta | \vec{g}]_X, \pi)\right) \tag{3.8}$$

with a probability $P(\text{SHIFT}(w_{j+1}) | [_i \underline{A\alpha}\,{}_j\, Y\beta | \vec{g}]_X)$ where

$$\vec{h} = (Y, \underline{A}, \underline{g}_2)$$
$$\underline{W} = W/w_{j+1}$$
$$\sum_{w \in V} P(\text{SHIFT}(w) | [_i \underline{A\alpha}\,{}_j\, Y\beta | \vec{g}]_X) = 1$$

- A move PROJECT(U, δ) applies

$$\left(w_{k+1}^n, ([_j \underline{\alpha}\,{}_k | \vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{k+1}^n, ([_j \underline{X}\,{}_k\, \delta | \vec{g}]_U, \pi)\right) \tag{3.9}$$

with a probability $P(\text{PROJECT}(U, \delta) | [_j \underline{\alpha}\,{}_k | \vec{g}]_X)$ where

$$\underline{X} = X/\text{head}(X, \underline{\alpha})$$
$$\sum_{U, \delta} P(\text{PROJECT}(U, \delta) | [_j \underline{\alpha}\,{}_k | \vec{g}]_X) = 1 - P(\text{ATTACH} | [_j \underline{\alpha}\,{}_k | \vec{g}]_X)$$

example, in Dutch, the head of 'een paar schoenen' is 'paar', while the head of 'een paar hemden' is 'hemden', but their local trees are identical.

- The ATTACH move applies

$$\left(w_{k+1}^{n}, ([_j \underline{\gamma}_k | \vec{h}]_Y, [_i \underline{A\alpha}_j Y\beta | \vec{g}]_X, \pi)\right) \quad \vdash \quad \left(w_{k+1}^{n}, ([_i \underline{A\alpha Y}_k \beta | \vec{g}]_X, \pi)\right) \tag{3.10}$$

with a probability $P(\text{ATTACH} | [_j \underline{\gamma}_k | \vec{h}]_Y)$ where

$$\begin{aligned} \vec{h} &= (Y, \underline{A}, \underline{g}_2) \\ \underline{Y} &= Y/\text{head}(Y, \underline{\gamma}) \\ P(\text{ATTACH} | [_j \underline{\gamma}_k | \vec{h}]_Y) &= 0 \quad \text{if } Y \neq h_1 \end{aligned}$$

Again, one may opt to regard ATTACH equivalent with PROJECT(ATT).

Note: It will be henceforth assumed that conditional moves are statistically independent from the $\underline{\alpha}$ feature (the categories of the resolved daughters except the first one). Its role in the evaluation of head$(X, \underline{\alpha})$ can be accounted for in other ways if one can assume that the head$(\cdot)$ function selects a head position and returns the lexical feature of the head daughter: for instance, the head position is looked up when the syntactic features of the mother and the sisters are projected, and the lexical head of the sister at head position is propagated as soon as it is being attached. So a wildcard '$*$' can replace $\underline{\alpha}$, which enhances the time and space efficiency of the parser.

3.3.4. Inducing a probabilistic left corner grammar from a treebank

Submodels

The move probabilities are conditioned on the topmost constituent of the PLCA stack. In this thesis, these probabilities are estimated from training data. Given that there is always a limit to available matching training data, the PLCA moves are assumed independent from most of the features of the topmost constituent. Only a few of them can effectively serve for probabilistic conditioning. A typical parameterization is (cf. Eqs. 4.1–4.2):

$$P(\text{SHIFT}(w) | [_i \underline{A} *_j Y\beta | \vec{h}]_X) = p_s(w | Y, \underline{A}, \underline{h}_2), \tag{3.11}$$

$$P(\text{PROJECT}(U, \delta) | [_j \underline{A} *_k | \vec{g}]_X) = p_p(U, \delta | g_1, X, \underline{A}, \underline{g}_2), \tag{3.12}$$

$$P(\text{ATTACH} | [_j \underline{A} *_k | \vec{g}]_X) = p_a(ATT | g_1, X, \underline{A}, \underline{g}_2). \tag{3.13}$$

p_s, p_p and p_a are ensembles of conditional *pmf*s and will be called **submodels** in this text.

After fixing the parameterization, the submodels are initialized using methods similar with C&J, as explained next. The training corpus is a human-annotated or machine-generated treebank. Each tree is decomposed into its PLCA derivation steps. That is,

given the above parameterization, the treebank is transformed in a stream of independent joint events of the types

$$(\text{SHIFT}(w), Y, \underline{A}, \underline{h}_2),$$
$$(\text{PROJECT}(U, \delta), g_1, X, \underline{A}, \underline{g}_2),$$
$$(\text{ATTACH}, g_1, X, \underline{A}, \underline{g}_2).$$

The shift events observed by the C&J model would rather look as $(\text{SHIFT}(w), \underline{A}, \underline{h}_2)$, that is, without Y.
Then, submodels are estimated from relative frequencies, using the standard language modeling techniques such as smoothing, interpolation and back-off. An overview of these techniques was given in Chapter 1. For instance, a p_s smoothed by linear interpolation with a lower order version of it, say $\tilde{p}_s$, may look like:

$$\tilde{p}_s(w|Y, \underline{A}, \underline{h}_2) = d \times \frac{C(\text{SHIFT}(w), Y, \underline{A}, \underline{h}_2)}{\sum_{v \in V} C(\text{SHIFT}(v), Y, \underline{A}, \underline{h}_2)} + (1-d) \times \tilde{p}_s(w|Y, \underline{A}),$$

where $C(x)$ is the observed frequency of the event x in the training corpus, and d is an interpolation factor.
Note: Linguistically speaking it does not make sense to assign an attach probability greater than 0 if $X \neq g_1$; in the experiments, attachment was always disallowed if $X \neq g_1$. In afterthought however, for the language model it may be useful to still allow attachment in this case; one would rely on the reestimation procedure (cf. Sec. 3.5.3) to find the optimal estimate, which may be greater than 0.
Note: It may be useful to have different parameterizations for projections at different levels in the parse tree.

Probabilistic Left Corner Grammar

Now a probabilistic left corner grammar (PLCG) can be defined as a 5-tuple $\Gamma = (N, V, p_s, p_p, p_a)$ where N is a finite set of non-terminal category labels including $\mathtt{W}$, V is a finite set of terminal category labels (the vocabulary), p_s is a shift submodel, p_t is a project submodel and p_a is an attach submodel.
Note that there is no need to specify a finite set of rules in a PLCG: all necessary information is given by the submodels. Given a PLCG, the construction of a corresponding PLCA is trivial.
The submodels are naturally induced from a treebank using statistical estimation methods. Statistical estimation avoids problems that arise with other methods:

- The submodels could be computed from a PCFG with a method similar to Stolcke's method to compute a probabilistic transitive left corner relation from a PCFG Stolcke [1995]. It is however impractical for large PCFGs; in my experiments, lexicalization and encoding of context in the category labels would lead to huge PCFGs.

Table 3.1. A PLCA trace generating the sentence *<s> ann likes john </s>*.

constituent at stack top	move	prob
$[_0\ SB/\texttt{<s>}_1\ TOP'\,|\,TOP, SB/\texttt{<s>}, SB/\texttt{<s>}]_{TOP}$	SHIFT(*ann*)	0.1
$[_1\ W/ann_2\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{W}$	PROJECT(*NNP*, ε)	1.0
$[_1\ W/ann_2\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{NNP}$	PROJECT(*S*, *VP*)	0.7
$[_1\ NNP/ann_2\ VP\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{S}$	SHIFT(*likes*)	0.2
$[_2\ W/likes_3\,|\,VP, NNP/ann, SB/\texttt{<s>}]_{W}$	PROJECT(*VB*, ε)	1.0
$[_2\ W/likes_3\,|\,VP, NNP/ann, SB/\texttt{<s>}]_{VB}$	PROJECT(*VP*, *NNP*)	0.5
$[_2\ VB/likes_3\ NNP\,|\,VP, NNP/ann, SB/\texttt{<s>}]_{VP}$	SHIFT(*john*)	0.1
$[_3\ W/john_4\,|\,NNP, VB/likes, NNP/ann]_{W}$	PROJECT(*NNP*, ε)	1.0
$[_3\ W/john_4\,|\,NNP, VB/likes, NNP/ann]_{NNP}$	ATTACH	1.0
$[_2\ VB/likes\ NNP/john_4\,|\,VP, NNP/ann, SB/\texttt{<s>}]_{VP}$	ATTACH	1.0
$[_1\ NNP/ann\ VP/likes_4\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{S}$	PROJECT(*TOP'*, *SE*)	0.8
$[_1\ S/likes_4\ SE\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{TOP'}$	SHIFT(*</s>*)	1.0
$[_4\ W/\texttt{</s>}_5\,|\,SE, S/likes, SB/\texttt{<s>}]_{W}$	PROJECT(*SE*, ε)	1.0
$[_4\ W/\texttt{</s>}_5\,|\,SE, S/likes, SB/\texttt{<s>}]_{SE}$	ATTACH	1.0
$[_1\ S/likes\ SE/\texttt{</s>}_5\,|\,TOP', SB/\texttt{<s>}, SB/\texttt{<s>}]_{TOP'}$	ATTACH	1.0
$[_0\ SB/\texttt{<s>}\ TOP'/\texttt{</s>}_5\,|\,TOP, SB/\texttt{<s>}, SB/\texttt{<s>}]_{TOP}$		
Total derivation probability		0.00056

- Left corner parsing can be simulated by a top-down, left-to-right parser constructed from the *left corner transform* of a CFG [Rosenkrantz and Lewis II, 1970]. Unfortunately, the left corner transform blows up the original CFG although Johnson and Roark [2000] provide a way to slightly alleviate that problem.

- One can also transform the *treebank* instead of the treebank-induced PCFG; the PCFG induced from the transformed treebank, then, is equivalent with the PLCG induced from the original treebank (see for instance [Johnson, 1998, Roark and Johnson, 1999, Roark, 2001]). This approach does not seem tenable with detailed contexts and lexicalized categories, however.

An additional advantage of estimating the submodels directly from a treebank is the flexibility in the statistical estimation process; for instance, various smoothing methods can be applied straightforwardly without complicating the estimation algorithm.

3.3.5. A toy example

Table 3.1 shows one possible execution trace of an artificial PLCA generating the sentence *<s> ann likes john </s>*. The 3rd column is the conditional probability of the move (in the 2nd column), given the constituent at the stack top (in the 1st column).

3.3.6. Summary

The stochastic left corner parser constructs subtrees from left to right and from bottom to top like C&J's shift-reduce parser, but the bottom-up process is conditioned on top-down information (soft top-down filtering), namely the goal category; one could interpret top-down filtering in bottom-up parsing as the equivalent of 1-word lookahead in top-down parsing, as in Roark's model.
Probabilities of left corner parser moves can be estimated directly from a treebank; they are conditioned on non-local and partly lexicalized features from the partial analyses. The probabilistic left-corner grammar is defined as the ensemble of all the left corner parsing move probabilities. The non-deterministic PLCG parsing is simulated by a probabilistic push-down automaton, of which all the rules and their probabilities are unambiguously specified by the PLCG.
The constituent was defined as the container of all the information of a partial analysis that is relevant for the next parser move. It is the basis of the knowledge presentation used by the deterministic PLCG parsing algorithm, developed in the next section.

3.4. PLCG parsing in a compact network

This section develops an efficient synchronous PLCG parsing algorithm. Being ultimately interested in language modeling, special attention is paid to how the sum of the probabilities of distinct parsing paths can be obtained efficiently. Not only probability sums of full paths, but also those of partial paths are needed to compute conditional language model probabilities.

3.4.1. Efficient representation of PLCG derivations

It may be expected that there are many different PLCG derivations with a non-zero probability that produce the same input sentence (prefix). An efficient representation of these (partial) derivations lessens memory requirements and facilitates a time-efficient traversal of the derivations. Stated otherwise, given fixed memory and time resources, a more efficient representation will allow to evaluate more derivations — since in practice pruning is always needed.
The least efficient representation one can think of, lists each (partial) derivation next to each other; at each point of indeterminism, the derivation prefix is copied for as many choices there are.
In a first step, it can be realized that the derivation prefix does not have to be copied at a point of indeterminism, but that forking from the same node suffices. In effect one obtains a tree representation of the derivation space (the 'derivation prefix tree'), as is used in [Chelba and Jelinek, 1999] and [Roark, 2001].
However, many inefficiencies remain. For instance, whenever an unresolved constituent calls for the construction of a daughter constituent of a certain category, and

that daughter constituent was realized in N different ways, then it should not be necessary to proceed with N derivations, since the details of the generation of the daughter constituent does not influence the rest of the derivation.

As another example, when two separate derivations arrive at unresolved constituents both shifting the next word, then identical constituents consisting of that word and identical contexts may result. What follows after the shifts is identical in both derivations — until the attachments that respectively correspond with the shifts.

These inefficiencies can be generally described as follows: there are still identical parts in the derivation prefix tree that occur in memory multiple times, and are separately traversed. Each time when two or more separate partial derivations happen to arrive at identical constituents at their stack tops, an identical choice of moves is to be made.

A network-based representation that eliminates this inefficiency is developed in the following section. The network results from 'joining' identical nodes in the derivation prefix tree. Consequently, a network node represents multiple partial derivations; Sec. 3.4.3 discusses how to compute the sum of their probabilities, leading to a synchronous parsing algorithm described in Sec. 3.4.4.

3.4.2. The PLCG network

The discussion now moves from a PLCA view on the PLCG to a network view. First, it is important to realize that the contents of the PLCA stack can be reconstructed at any time if the preceding moves are known and that in the previous section the PLCA was designed such that the next moves are conditioned on the current top of the stack only. This leads to a linear directed graph representation of the generation of a parse tree by a PLCA where nodes are labeled with the top of the stack (instead of the complete stack) and arcs are labeled with a move. Henceforth, this linear graph is called a *path*. The probability of the path is the product of the conditional probabilities of its moves, so it equals the probability of the corresponding parse tree.

Given a PLCG, the associated network is the union graph of all paths with a probability greater than 0. This network could be a path prefix tree, and as such it would be equivalent with a derivation tree. However, it is my aim to make this network as compact as possible. This is obtained by the requirement that there are no two nodes with the same label:

A **PLCG network** $\mathscr{G} = (\mathscr{G}^n, \mathscr{G}^a)$ is a directed acyclic graph containing nodes and directed arcs between these nodes. Each node $q \in \mathscr{G}$ is labeled with a constituent and there are no two nodes with the same label. Each arc is labeled with a move and the conditional probability of that move given the label of the source node. The w_0^n-**constrained** network $\mathscr{G}_{w_0^n}$ is the subgraph of $\mathscr{G}$ that includes all paths that generate parse trees that yield w_0^n with a probability greater than 0, but no other paths. There is at most one arc between two nodes.

For convenience of notation, nodes and arcs are treated as completely equivalent with their labels. For instance, $q = [_i Y *_j \beta|\vec{h}]_X$ denotes that the label of the node q is $[_i Y *_j \beta|\vec{h}]_X$. And $(q_i, q_j) = \text{ATTACH}(q_k)$ denotes that the arc from q_i to q_j is labeled $\text{ATTACH}(q_k)$. The conditional move probability is written as $P(q_j|q_i)$.
Using a parse tree representation adapted for language modeling (cf. Fig. 3.2(c)), the **initial node** is q_I (cf. (3.6)), and the **final node** is q_F (cf. (3.7)), where n is the number of input words, excluding `<s>` but including `</s>`.
A **path** is a linear subgraph of $\mathscr{G}$, represented as a sequence of moves or a sequence of constituents. The **path set** denoted by $\langle q_i, q_j \rangle$ is the set of all paths starting from q_i and arriving in q_j. ℓ is a **complete path** if $\ell \in \langle q_I, q_F \rangle$. The **precedes** relation is denoted by $\prec$. $q_i \prec q_j$ if and only if $\langle q_i, q_j \rangle \neq \emptyset$.
A **path concatenation** operator is defined now for convenience. The concatenation of two paths ℓ_1 and ℓ_2, written as $\ell_1\ell_2$, is the union graph of ℓ_1 and ℓ_2. It is only defined if ℓ_1 arrives in the node where ℓ_2 departs. For path sets,

$$\langle q_1, q_2 \rangle \langle q_2, q_3 \rangle \doteq \{((q_1, q_2))\ell | \ell \in \langle q_2, q_3 \rangle\}, \tag{3.14}$$

$$\langle q_1, q_2 \rangle (q_2, q_3) \doteq \{\ell((q_2, q_3)) | \ell \in \langle q_1, q_2 \rangle\}, \tag{3.15}$$

$$\langle q_1, q_2 \rangle \langle q_2, q_3 \rangle \doteq \{\ell_1\ell_2 | \ell_1 \in \langle q_1, q_2 \rangle, \ell_2 \in \langle q_2, q_3 \rangle\}. \tag{3.16}$$

Note that if $\ell_1\ell_2$ is defined, ℓ_1 and ℓ_2 have only one node in common; if there were another common node, $\mathscr{G}$ would contain a cycle, which is in contradiction with the definition.
As an illustration, Fig. 3.8 shows a fragment of a severely pruned PLCG network. It contains 4 partial paths.

3.4.3. Computing sums of path probabilities

One node in the PLCG network corresponds with multiple partial derivations. This section explains how the sum of their probabilities — which will be called the *forward probability* of the node — can be computed efficiently.
Efficiency is in principle obtained by the *dynamic programming* (DP) principle of storing (tabulating) and maximally re-using intermediate results. In PCFG chart parsing, a well-known DP parsing algorithm, the 'intermediate result' is stored as a chart item annotated with its probability. The item is constructed from previously constructed daughter items, and its probability is computed from the probabilities of its daughter items. In the PLCG network, the 'intermediate result' is constructed as a network node annotated with its forward probability. The node and its forward probability is constructed from previously constructed nodes and their probabilities; there must be a valid PLCG move between the node and each of its predecessors. Duplication of work is avoided by requiring that each node be unique.
Actually, the computation of yet another probability, the *inner probability*, is necessary in the PLCG network, because of the 'attach constraint', which is explained first.

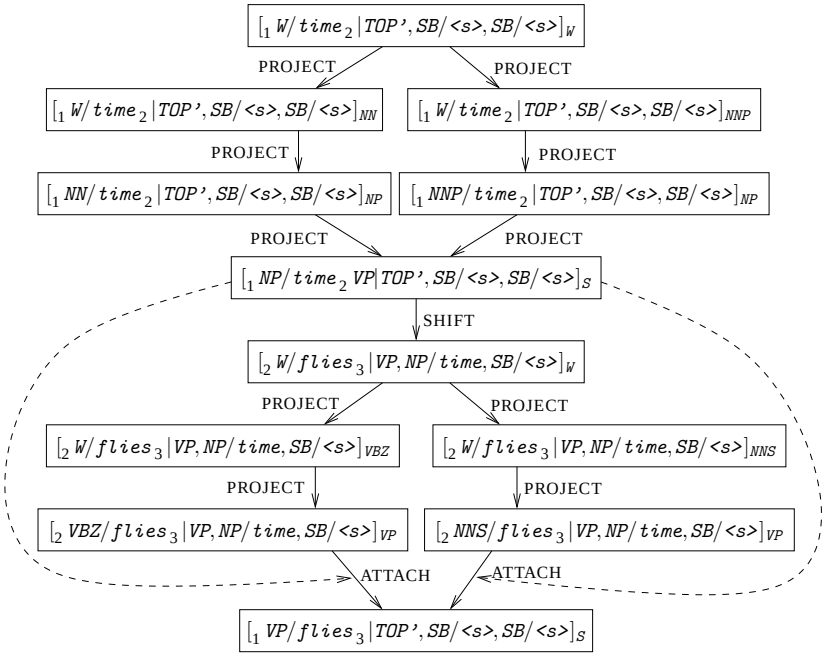

Figure 3.8. Fragment of a (severely pruned) PLCG network constrained to the sentence *<s> time flies </s>*. A dashed arrow connects an ATTACH with its argument.

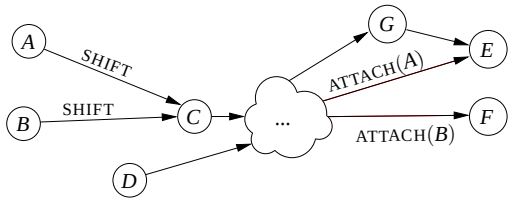

Figure 3.9. The PLCG network is not Markov due to the constraint that a path containing ATTACH(q) must visit q. The sequence $(A,C,\ldots,F)$ does not constitute a valid path. Neither does $(B,C,\ldots,E)$. On the other hand $(B,C,\ldots,F)$ and $(B,C,\ldots,G,E)$ are valid paths.

The attach constraint

A move probability is conditioned only on its source node q. However, there is one subtlety to this. Within one single path, an ATTACH move refers implicitly, but unambiguously, to an *attachment* node: the node labeled with the constituent that is being completed by the constituent at the source node of the ATTACH move. However, in a graph, the ATTACH move cannot extend a path prefix that does not contain a corresponding attachment node. I refer to this non-Markov property as the *attach constraint*. In order to be able to decide whether the attach constraint is satisfied, the

ATTACH move is henceforth augmented with the corresponding attachment node as its argument. Fig. 3.9 should make this more clear.
Fortunately, within the same path attach constraints are *nested*, which means that any other legal subpath between q and $\text{ATTACH}(q)$ found in the PLCG network can be substituted to form another legal full path. This property is formally expressed as follows. The proof is given in Appendix A.

Lemma 1 *Let t be a partial path in a PLCG network $\mathscr{G}$ and $(q,q') = \text{ATTACH}(q'')$ be a move in t. If $\text{ATTACH}(s)$ is a move in a path $u \in \langle q'', q\rangle$ then $q'' \prec s$.*

The following lemma identifies a subgraph $\langle q^o, q\rangle$ that will play a role in the definition of the inner probability. The proof is also given in Appendix A.

Lemma 2 *If $q = [_i \underline{X} *_j \beta|\vec{h}]_Z$, then on every w_0^i-constrained path in $\langle q_I, q\rangle$ there is a node $q^o = [_i w_{i+1\ i+1}|\vec{h}]_W$.*

The superscript 'o' will henceforth be used in this meaning.

Forward and inner probability

A language model returns an estimate of the probability of an input sentence w_0^n with $w_0 = \texttt{<s>}, w_n = \texttt{</s>}$. A path $t \in \langle q_I, q_F\rangle$ uniquely corresponds with a derivation (w_0^n, T); therefore $P(w_0^n) = \sum_T P(w_0^n, T) = \sum_{t\in\langle q_I,q_F\rangle} P(t)$. It is impractical, if not infeasible, to enumerate all t. However, by defining forward and inner probabilities, the sum can be obtained implicitly. The following treatment based on forward and inner probabilities is inspired by work on PCFG parsing by Stolcke [1995].

Definition 3 (forward probability) *Given a constrained PLCG network $\mathscr{G}_{w_0^n}$. The* forward probability *of a node q is*

$$\mu(q) \doteq \sum_{t\in\langle q_I,q\rangle} P(t).$$

Definition 4 (inner probability) *Given a constrained PLCG network $\mathscr{G}_{w_0^n}$. Consider a node $q = [_i X *_j \beta|\vec{h}]_Z$, $Z \neq \texttt{W}$. The inner probability of q is defined as*

$$\nu(q) \doteq \sum_{t\in\langle q^o,q\rangle} P(t).$$

If $Z = \texttt{W}$, $q^o = q$ and $\nu(q) \doteq 1$.

Using forward probabilities, the problem of finding $P(w_0^n)$ is reformulated as the problem of finding $\mu(q_F)$. If $\mathscr{G}_{w_0^n}$ were Markov, the edges (q,q') for a given q only depend on q itself and are independent of the path that led to q. Thus

$$\langle q_I, q'\rangle = \bigcup_{q:(q,q')\in\mathscr{G}_{w_0^n}} \langle q_I, q\rangle\,(q,q'),$$

and, since the path sets under the union operator are disjunct,

$$\mu(q') = \sum_{q:(q,q')\in\mathscr{G}_{w_0^n}} \mu(q)P((q,q')).$$

So $\mu(q_F)$ could be obtained as follows: first initialize $\mu(q_I) = 1$, then visit all $q' \in \mathscr{G}_{w_0^n}$ in topological order, computing $\mu(q')$ from all $\mu(q)$ and $P((q,q'))$ for $(q,q') \in \mathscr{G}_{w_0^n}$. The PLCG network, however, is not Markov due to the attach constraint, which complicates the computation of forward probabilities. The recursion formula involves inner probabilities:

$$\mu(q') = \sum_{q,q''} \mu(q'')P(q^o|q'')\nu(q)P(q'|q) + \sum_{q} \mu(q)P(q'|q), \tag{3.17}$$

$$\nu(q') = \sum_{q,q''} \nu(q'')P(q^o|q'')\nu(q)P(q'|q) + \sum_{q} \nu(q)P(q'|q), \tag{3.18}$$

where — in both equations — the first sum is over all q for which $(q,q') \in \mathscr{G}_{w_0^n}$ and $(q,q') = \text{ATTACH}(q'')$, the second sum is over all q for which $(q,q') \in \mathscr{G}_{w_0^n}$ and (q,q') is a PROJECT move. A derivation of (3.17) and (3.18) is given in Appendix B.

3.4.4. Synchronous parsing algorithm

The essential task of a PLCG parser is computing the inner and forward probabilities of the nodes in the constrained network. Conceptually, the PLCG network is static for a given PLCG, and only the forward and inner probabilities have to be recomputed for each new input sentence. However in practice it is only feasible to allocate nodes dynamically the first time they get visited. A node that does not exist in main memory is assumed to have a zero or 'negligible' forward probability for the current input sentence.

The recursion formulas (3.17) and (3.18) for the computation of the inner and forward probabilities are now translated into three parsing subroutines: PROJECT-NODE, ATTACH-NODE and SHIFT-NODE. These routines take a source node and a move as input. If the target node does not exist yet in main memory, they allocate it and initialize the inner and forward probabilities to 0, otherwise they just update the probabilities according to (3.17) and (3.18). They return the target node. Assume a source node $q = [_i \underline{Y} *_j \alpha|\vec{h}]_X$. Then

1. If α is not empty, say $\alpha = Z\beta$, SHIFT-NODE(q,w) retrieves or initializes q' and an arc (q,q'), with $q' = [_j w/w_{j+1\ j+1}|\vec{g}]_w$ where $\vec{g} = (Z,\underline{Y},\underline{h}_2)$. $\nu(q')$ is initialized to 1, $\mu(q')$ is incremented with $\mu(q)p_s(w|q)$.

2. If α is empty, PROJECT-NODE(q,U,δ) retrieves or initializes q' and an arc (q,q'), with $q' = [_i \underline{X}_j \delta|\vec{h}]_U$. $\mu(q')$ is incremented with $f\mu(q)$ and $\nu(q')$ is incremented with $f\nu(q)$, with $f = p_p(U,\delta|q)(1-p_a(\mathit{ATT}|q))$.

initialize q_I
for each $j = 1, 2, \ldots, n$:
  for each $i = j-1, j-2, \ldots, 0$:
    for each q for which $\text{start}(q) = i, \text{pos}(q) = j$:
      for each U, δ for which $p_p(U, \delta | q) > 0$:
        $q' \leftarrow$ PROJECT-NODE(q, U, δ)
        if q' is resolved:
          schedule q' for further PROJECT-NODE/ATTACH-NODE
      if q is attachable:
        for each matching node m:
          $q' \leftarrow$ ATTACH-NODE(q, m)
          if q' is resolved:
            schedule q' for further PROJECT-NODE/ATTACH-NODE
  if $j = n$, return
  for each $i = 0, 1, \ldots, j-1$:
    for each q for which $\text{start}(q) = i, \text{pos}(q) = j$:
      $q' \leftarrow$ SHIFT-NODE(q, w_j)

Figure 3.10. Word-synchronous PLCG parsing algorithm.

3. Suppose α is empty and a matching $q'' = [_k \underline{h}_2 *_i h_1 \beta | \vec{g}]_Z$, $\underline{g}_2 = \underline{h}_3$ is found for some $Z, k, \beta, g_1, \underline{g}_3$. Then ATTACH-NODE$(q, q'')$ retrieves or initializes q' and an arc (q, q') with $q' = [_k \underline{U} *_j \beta | \vec{g}]_Z$. Furthermore, according to (3.17), $\mu(q')$ is incremented with $\mu(q'') p_s(q^o | q'') \nu(q) p_a(\mathit{ATT} | q)$. According to (3.18), $\nu(q')$ is incremented with $\nu(q'') p_s(q^o | q'') \nu(q) p_a(\mathit{ATT} | q)$.

In order to allow the PLCG-based language model to operate in a conditional mode (i.e. to return the probability of a next word given the words preceding it) I developed a word-synchronous parsing algorithm: partial analyses, constrained by a left context w_0^i, are extended in parallel observing the constraints imposed by w_{i+1}. Fig. 3.10 outlines the parsing algorithm used by the PLCG-based language model.

3.4.5. Pruning

Dynamic programming allows to account for more parsing paths than beam search with the same amount of computer time and memory. Still it usually remains necessary to prune the constrained network quite severely. The following fairly simple pruning scheme proved to provide a satisfying time/performance trade-off in my experiments:

1. Consider nodes $\mathscr{Q}_{ij} = \{q|\text{start}(q) = i, \text{pos}(q) = j\}$ and let $M_{ij} = \max_{q \in \mathscr{Q}_{ij}} \mu(q)$. Then $q \in \mathscr{Q}_{ij}$ is pruned if $\mu(q) \cdot \rho < M_{ij}$.
2. ρ is an *adapted beamwidth*. If N_{ij} is the number of nodes in $\mathscr{Q}_{ij}$, then $\rho = \rho^o N_{ij}^{-\sigma}$. ρ^o is the *maximum beam width*, σ is the *beam narrowing factor*.
3. ρ^o and σ are specified by the user. Typical values are $\rho^o = 10^4$, $\sigma = 0.3$.

The beam narrowing factor guards the parser against a combinatorial explosion of execution time in situations where too many nodes have forward probabilities close to each other.[3]

3.5. The PLCG-based language model

In this section, a LM and a CLM is derived from the word-synchronous PLCG parsing algorithm. The CLM offers a few advantages over the LM. First of all, it can be combined with other CLMs, thereby providing a rough but usually effective tool for smoothing and mixing in other knowledge sources.
Another advantage is that a CLM can be applied earlier in the search process than a LM, reducing the input/output delay and potentially increasing search efficiency — although in practice a multi-pass approach is still preferred where a simple CLM (for instance, a trigram) is applied in a first pass and the advanced CLM or LM rescores the remaining hypotheses in the second pass. This is because the limited reduction of the search effort by advanced CLMs in the first pass would not compensate for the added computational complexity.

3.5.1. The PLCG-based language model

The PLCG-based LM straightforwardly returns the forward probability of the final node after parsing: for an input sentence W

$$\mu(q_F) = \sum_{t \in \langle q_I, q_F \rangle} P(t) = \sum_{W,T} P(W,T) = P(W), \tag{3.19}$$

since each path through the W-constrained network corresponds with 1 PLCG derivation (W,T).
In practice, pruning will cause $\mu(q_F)$ to systematically underestimate the sentence probability. Alternatively, one can compose the sentence probability with conditional probabilities using the chain rule $P(W) = \sum_{i=1}^{n} P(w_i|w_0^{i-1})$ where $W = w_0^n$. The conditional probabilities are emitted by the CLM, explained in the following section.

3. The beam narrowing factor can be considered a generalization of Roark's pruning method in [Roark, 2001], which would correspond with $\sigma = 3$, or in [Roark and Johnson, 1999], which would correspond with $\sigma = 1$. One important difference, though, is that I have one beam per group of nodes $\mathscr{Q}_{ij}$, while Roark's approach would rather correspond with pruning globally over $\bigcup_i \mathscr{Q}_{ij}$.

3.5.2. The PLCG-based conditional language model

In this section, expressions for the prefix probabilities are derived, which will lead to an efficient calculation of conditional probabilities as the ratio of prefix probabilities. Due to pruning, the prefix probabilities lose probability mass, which would lead to an underestimate of the conditional probabilities. However, it will be shown how a particular partitioning of the constrained PLCG graph allows the ratio of prefix probabilities to be rewritten as a weighted, normalized average of shift probabilites, which is guaranteed to be normalized in the face of pruning.
As a first step, the prefix probability $P(w_0^j)$ is expressed as the sum of probabilities of all partial paths that generate w_0^j and end just before shifting w_{j+1}. This sum is efficiently computed as a sum of partial sums, where each partial sum is available as the forward probability of a node where a subset of the considered partial paths arrive. The set of these nodes is called $\mathscr{H}_j$ below.
One can then see that, in the absence of pruning, the total probability of all partial paths that end just after shifting w_j equals the total probability of all partial paths that end just before shifting w_{j+1}: the probability mass is preserved, since it is only re-distributed. In other words, $P(w_0^j)$ is alternatively expressed as a sum of forward probabilities of nodes where these partial paths arrive. The set of these nodes is called $\mathscr{W}_j$ below.
The following lemma allows a more formal treatment of the above intuitive reasoning. (The proof is trivial and therefore omitted.)

Lemma 3 *Assume the following two subsets of nodes of the constrained PLCG graph* $\mathscr{G}_{w_0^n}$*:*

$$\mathscr{H}_j \doteq \{q \in \mathscr{G}_{w_0^n}^n | pos(q) = j, q \textit{ is unresolved}\}$$
$$\mathscr{W}_j \doteq \{q \in \mathscr{G}_{w_0^n}^n | pos(q) = j, cat(q) = \texttt{W}\}.$$

The sets $L_q = \{t \in \langle q_I, q_F \rangle : q \in t\}, q \in \mathscr{H}_j$*, form a partition of* $\langle q_I, q_F \rangle$ *since each path in* $\langle q_I, q_F \rangle$ *traverses exactly one* $q \in \mathscr{H}_j$*. In the same way the sets* $L_q = \{t \in \langle q_I, q_F \rangle : q \in t\}, q \in \mathscr{W}_j$*, form a partition of* $\langle q_I, q_F \rangle$*.*

The forward probability of a node $q \in \mathscr{H}_j$ can be interpreted as a real probability. Each path in $\langle q_I, q \rangle$ uniquely corresponds with one way of generating w_0^j and arriving in q. In other words $\mu(q)$, being the sum of probabilities of these paths, is the joint probability of w_0^j and q:

$$\mu(q) = P(w_0^j, q) \tag{3.20}$$

With a similar reasoning on $\mathscr{W}_j$ and due to Lemma 3 one now obtains two simple formulas to compute *prefix probabilities*:

$$P(w_0^j) = \sum_{q \in \mathscr{H}_j} \mu(q) = \sum_{q \in \mathscr{W}_j} \mu(q), \tag{3.21}$$

and conditional probabilities:

$$P(w_{j+1}|w_0^j) = \frac{P(w_0^{j+1})}{P(w_0^j)} = \frac{\sum_{q\in\mathscr{W}_{j+1}} \mu(q)}{\sum_{q\in\mathscr{H}_j} \mu(q)} = \frac{\sum_{q\in\mathscr{H}_j} \mu(q) P_s(w_j|q)}{\sum_{q\in\mathscr{H}_j} \mu(q)}. \tag{3.22}$$

Thus follows the intuitively appealing conclusion that $P(w_{j+1}|w_0^j)$ is a normalized and weighted average of the shift probabilities $p_s(w_{j+1}|q)$ with weights $\mu(q)$.
Contrary to the computed whole-sentence probability (3.19) and the prefix probabilities (3.21), (3.22) remains a proper probability under path pruning. Another important advantage is that the conditional probabilities can be emitted on-line during parsing, namely at the end of the main loop (Fig. 3.10).

3.5.3. Maximum-likelihood training

The PLCG-based language model belongs to the class of 'parsing' syntax-based language models. All parsing language models (PLMs) have in common that they build a grammatical parse structure T as a *hidden event* in order to predict a sentence W, the *observed event*. In most training scenarios, the PLM is initialized on a parse-annotated corpus Ω^* using the maximum-likelihood criterion. In other words, the initial PLM maximizes likelihood of the joint event (W,T). So in that scenario the initial PLM is optimal as a parser, because $\arg\max_T P(W,T) = \arg\max_T P(T|W)$ since W is considered given. A language model however should maximize the expected probability of a test corpus, regardless of the accuracy of the intermediate hidden structure. Hence it makes sense to maximize the likelihood of the unannotated training corpus Ω instead of Ω^*.
An EM-training procedure was proposed in [Bod, 2000] for the case of a language model based on a data-oriented lexical-functional grammar parser (LFG-DOP). However, in the latter case the EM updates are less complicated because the LFG-DOP graph satisfies the Markov property (like HMMs), while the PLCG graph does not, due to the attach constraint.
Another example is found in [Chelba, 2000] where an approximation of an EM training procedure is proposed for the structured language model; later Van Aelten and Hogenhout [2000] developed, implemented and experimented with a full EM training procedure based on the dynamic programming version of the structured language model [Jelinek and Chelba, 1999].
In this section, I derive an Expectation-Maximization training procedure as an instance of the general treatment in [Dempster et al., 1977].

EM updates

Let Ω be the unannotated training corpus. Let Ω^* denote Ω annotated with hidden structure generated by a model θ, and suppose one wants to update an old model θ^o. The E-step of the EM algorithm involves the construction of an auxiliary function

$Q(\theta,\theta^o)$, the θ^o-expectation of the logarithm of the θ-likelihood of Ω^* given Ω. In the context of this thesis it can be written as

$$\begin{aligned} Q(\theta,\theta^o) &= \sum_{W\in\Omega}\sum_{T} P_{\theta^o}(T|W)\log P_\theta(W,T) \\ &= \sum_{W\in\Omega}\sum_{t\in\langle q_I,q_F\rangle} P_{\theta^o}(t|W)P_\theta(t) \\ &= \sum_{W\in\Omega}\sum_{t\in\langle q_I,q_F\rangle} P_{\theta^o}(t|W)\sum_{m\in t} P_\theta(m) \\ &= \sum_{m}\sum_{W\in\Omega}\sum_{t\in\langle q_I,q_F\rangle} \mathrm{F}_t(m)P_{\theta^o}(t|W)P_\theta(m), \end{aligned}$$

where m now ranges over all possible (move|context) events and $\mathrm{F}_t(m)$ is the frequency of the m event in path t. Using the definition of the *expected frequency* of m in the generation of W by θ^o as

$$\mathrm{EF}_W(m|\theta^o) \doteq \sum_{m}\sum_{t\in\langle q_I,q_F\rangle} P_{\theta^o}(t|W)\mathrm{F}_t(m), \tag{3.23}$$

and the total expected frequency of m in the generation of Ω as θ^o as

$$\mathrm{EF}_\Omega(m|\theta^o) \doteq \sum_{W\in\Omega} \mathrm{EF}_W(m|\theta^o). \tag{3.24}$$

$Q(\theta,\theta^o)$ can be rewritten as

$$Q(\theta,\theta^o) = \sum_m \mathrm{EF}_\Omega(m|\theta^o)\log P_\theta(m). \tag{3.25}$$

If one chooses (M-step)

$$\hat{\theta} = \arg\max_\theta Q(\theta,\theta^o), \tag{3.26}$$

then Dempster et al. [1977] proved that $P_{\hat{\theta}}(\Omega) \geq P_{\theta^o}(\Omega)$, so that the EM algorithm is guaranteed to converge.
The conditional probabilities $P_\theta(m)$ are subject to $\sum_{m:H(m)=h} P_\theta(m) = 1$ for each transition context h, where $H(m)$ denotes the conditioning context of m. Under these constraints the maximum is found in

$$P_{\hat{\theta}}(m) = \frac{\mathrm{EF}_\Omega(m|\theta^o)}{\sum_{n:H(n)=H(m)} \mathrm{EF}_\Omega(n|\theta^o)}. \tag{3.27}$$

Expected frequency of a move

The key of the EM-update is the expected frequency of a move given a context. In the constrained PLCG graph of a sentence W, each move (q_i,q_j) corresponds with such a

conditional event, written here as $M(q_i, q_j)$. The expected frequency of an event m can be rewritten as a sum of *visit* probabilities of the move (q_i, q_j) for which $M(q_i, q_j) = m$ in the W-constrained graph:

$$\mathrm{EF}_W(m) = \sum_{\substack{(q_i,q_j):\\ M(q_i,q_j)=m}} P((q_i,q_j)|W) \tag{3.28}$$

$$= \frac{1}{P(W)} \sum_{\substack{(q_i,q_j):\\ M(q_i,q_j)=m}} P((q_i,q_j),W). \tag{3.29}$$

$P((q_i,q_j),W)$ can be further decomposed as a sum of probabilities of paths t in $\langle q_I, q_F \rangle$ containing the (q_i,q_j) move. This sum can be efficiently computed from the constrained graph after the computation of *outer* probabilities.

Definition 5 (outer probability) *Let $q = [_i \underline{X} *_j \beta|\vec{h}]_Z$ be a node in a constrained PLCG network and consider all full paths that traverse q (and therefore also traverse q^o (cf. Lemma 2). The* ***outer probability*** *of q is now defined as*

$$\xi(q) = \sum_{t \in T(q)} \frac{P(t)}{P(q^o, \ldots, q)},$$

where $P(q^o, \ldots, q)$ is the probability of the path segment of t between q^o and q.

Let $T(q_i,q_j)$ be the set of full paths that visit both the adjacent nodes q_i and q_j. Then

$$P((q_i,q_j),W) = \sum_{t \in T(q_i,q_j)} P(t). \tag{3.30}$$

This sum can now be factorized as

$$P((q_i,q_j),W) = \begin{cases} \nu(q_i)\xi(q_i) & \text{if } (q_i,q_j) \text{ is a SHIFT} \\ \nu(q_i)P(q_j|q_i)\xi(q_j) & \text{if } (q_i,q_j) \text{ is a PROJECT} \\ \xi(q_j)\nu(q'')P(q_i^o|q'')\nu(q_i)P(q_j|q_i) & \text{if } (q_i,q_j) = \text{ATTACH}(q''). \end{cases} \tag{3.31}$$

A derivation of (3.31) is given in Appendix C.

Backward computation of ξ

The question, how to obtain the outer probabilities efficiently, still needs to be solved. It turns out that one can find a recursive formula in the style of (3.17) and (3.18). Assume $\xi(q')$ of all successors q' are given. Then

$$\xi(q) = \begin{cases} \sum_{q' \in A(q)} \xi(q')P(q'|q) + \cdots & \\ \cdots \sum_{(q',q'') \in B(q)} \xi(q')\nu(q'')P(q^o|q'')P(q'|q) & \text{if } q \text{ is resolved} \\ \sum_{(q_1,q_2) \in C(q)} \xi(q_2)\nu(q_1)P(q_2|q_1) & \text{else,} \end{cases} \tag{3.32}$$

where

$$A(q) \doteq \{q' | (q,q') \text{ is a } \textsc{project}\}$$
$$B(q) \doteq \{(q',q'') | (q,q') = \textsc{attach}(q'')\}$$
$$C(q) \doteq \{(q_1,q_2) | (q_1,q_2) = \textsc{attach}(q)\}.$$

A derivation of (3.32) is given in Appendix B.

Summary

In this section a maximum-likelihood training method for a PLCG-based language model based on the EM-algorithm was derived. It is summarized as follows:

1. Choose submodel parameterizations (cf. Sec. 3.3.4).
2. Initialize the submodels on the annotated corpus Ω^*: write each parse tree as a PLCG derivation and initialize the parameters using (3.27) where the EF_Ω are replaced with real frequencies F_{Ω^*}. Usually a statistical smoothing method is applied to alleviate the problem of data sparsity.
3. For each sentence in Ω:
   a) Forward pass: build the constrained PLCG graph using θ^o and update forward and inner probabilities synchronously with word-synchronous PLCG parsing algorithm (Fig. 3.10).
   b) Backward pass: find a reverse topological ordering so that a node q is only visited after all other nodes q' for which $q \prec q'$. Initialize $\xi(q_F) = 1$ and apply an update strategy based on (3.32).
   c) For each move in the constrained PLCG graph, compute the visit probability using (3.31); compute the sums (3.29) and update the corresponding counters (3.24).
4. Calculate the parameters of the reestimated model θ with (3.27). In practice a smoothing method is hereby applied to counteract overfitting.
5. Check convergence of the training corpus probability and monitor the probability of a held-out part of the training corpus according to θ; if convergence is reached or overtraining is detected, stop here. Otherwise let $\theta^o = \theta$ and reiterate from step 3.

3.6. Summary

This chapter developed the PLCG-based LM from an efficient synchronous dynamic-programming PLCG parser using rich context nodes. The model emits next word

probabilities in one single left-to-right pass, so it can be used as a CLM. It is initialized on a treebank using common LM techniques. I also presented an algorithm to optimize the PLCG-based LM further on plain text.

CHAPTER 4

Experiments with the PLCG-based language model

Experiments were conducted in two stages. In the first stage, small models were trained on the Penn Treebank corpus. These models were used to test the software, find appropriate submodel parameterizations and compare test set perplexities with other competing models. In the second stage I worked towards a more realistic large-vocabulary speech recognition setting: large models were trained and reestimated on the BLLIP-WSJ corpus, and their performance in n-best list rescoring was tested.

4.1. Parameterization and optimization of the submodels

4.1.1. Modeling

Data

The Penn Treebank [Marcus et al., 1993] (PTB) is a collection of text available from the LDC including material from the ATIS domain, the Wall Street Journal (WSJ), the Brown and Switchboard corpus. The text is annotated with hand-corrected labeled parse trees, part-of-speech tags, function labels (such as subject, location, etc.), anaphora and disfluency markers (the latter for Switchboard only). An example of an annotated sentence is shown in Fig. 4.1. A list of the labels and their meanings is reproduced from [Marcus et al., 1993] as Appendix D.

In my experiments, I only used the WSJ portion of the Penn Treebank (version 3). All information in the parse trees, other than the syntactic constituent labels was discarded (e.g. function markers, anaphora references). Sections 00–20 were used for training, while sections 21–22 were reserved for testing during development and sections 23–24 were used for final testing. The training set contains 42,073 sentences worth of

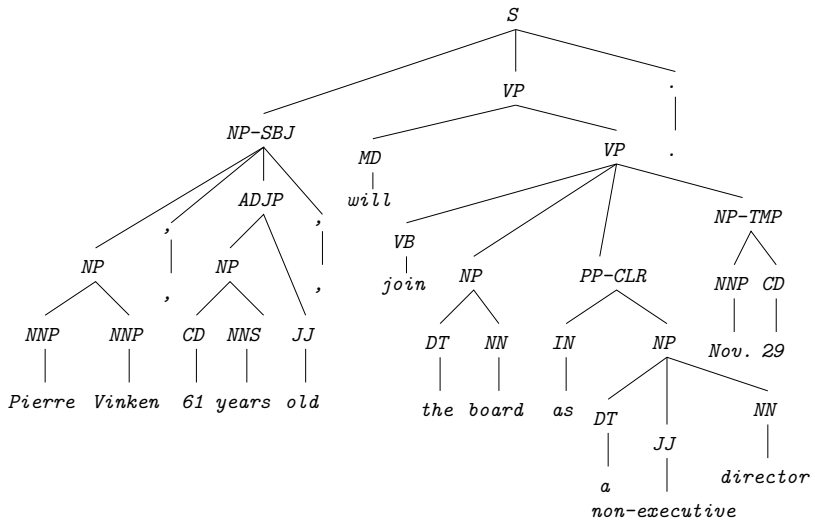

Figure 4.1. Example sentence from the PTB (first sentence of the WSJ portion, section 00). A list of the labels and their meanings is included as Appendix D.

1,046,082 word tokens (after tokenization, preserving punctuation). The development set contains 3,371 sentences (83,524 tokens) and the test set contains 3,759 sentences (93,185 tokens).
The labeled parse trees went through a number of preprocessing stages. I hereby attempted to match the preprocessing used in Chelba's experiments [Chelba, 2000] as closely as possible, in order to allow a reasonable comparison of performance results with Chelba and Jelinek's structured language model (C&J).

1. All terminals are converted to lowercase. Two different batches of experiments were run: 1. using non-verbalized punctuation (nvp), where punctuation characters are removed altogether; 2. using verbalized punctuation (vp), where punctuation characters are treated as regular terminals.

2. Numbers in Arabic digits are replaced by a token *N*.

3. A list of the 10,000 most frequent terminals (in sections 00–20) is collected. The terminals that are not in the word list are replaced with *<unk>*.

4. The original parse tree is encapsulated in a *TOP* constituent, cf. Fig. 3.2(a).

5. All constituents are annotated with a lexical head using deterministic rules by [Magerman, 1994].

6. Non-terminal unary productions are eliminated by collapsing two nodes connected by a unary branch to one node annotated with a combined label. For

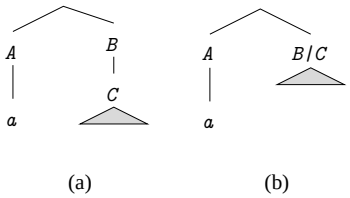

Figure 4.2. Eliminating unary productions.

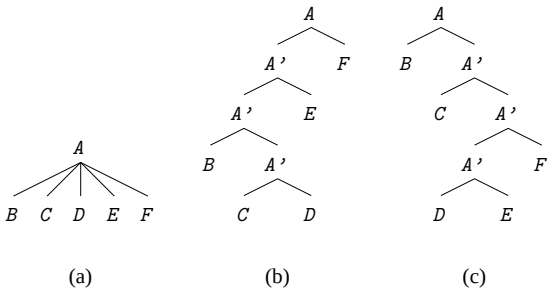

Figure 4.3. Binarization. `D` is assumed head of `A`.

instance, the tree fragment in Fig. 4.2(a) would be transformed to Fig. 4.2(b).

7. Parse trees are binarized as detailed in [Chelba, 2000, pp. 12–17]. There are two binarization schemes: the sister constituents to the left can be attached to the head constituent before or after the ones to the right. For instance, the 5-ary local tree with head daughter `D` in Fig. 4.3(a) would be transformed to Fig. 4.3(b) in the first scheme, and to Fig. 4.3(c) in the second scheme. The category of the mother node determines which of the two is applied. Intermediate nodes are marked with a prime in order to distinguish them from the original category.

The effect of the preprocessing (with the punctuation retained) is shown in Fig. 4.4.

Parameterization

For any move from any node $q = [_i \underline{Y} *_j \beta | \vec{h}]_X$ in the network, conditional move probabilities $P(\text{SHIFT}(w)|q)$, $P(\text{PROJECT}(U,\delta)|q)$ and $P(\text{ATTACH}|q)$ have to be given by the language model. The condition q has 10 categorical attributes (given that the considered parse trees are binary). Obviously, this is more than can possibly be used in estimating the conditional probabilities from data in practical situations. Therefore a good model parameterization needs to be found, i.e. one needs to determine which

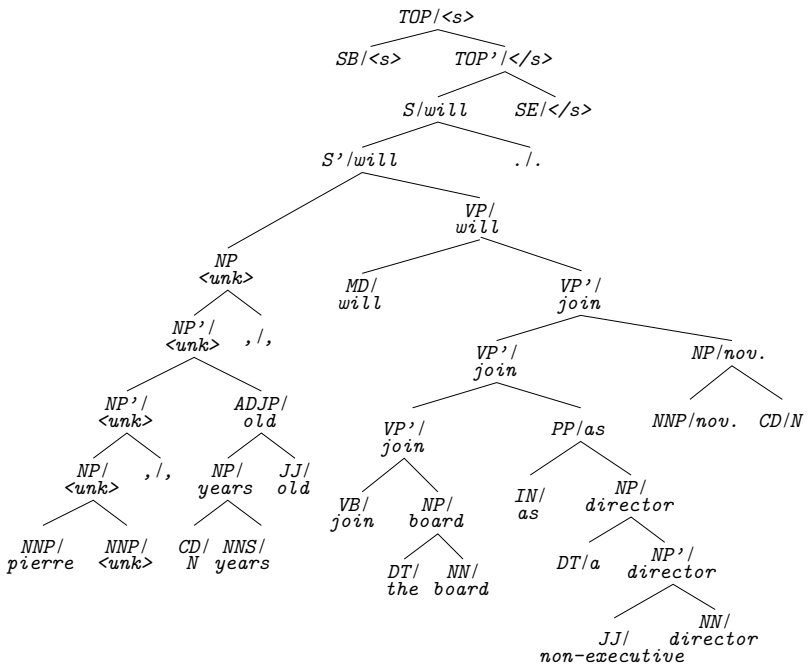

Figure 4.4. Preprocessed example sentence from Fig. 4.1.

attributes of q are informative enough, and in which order of significance. The latter is necessary because back-off smoothing is applied in order to account for data sparsity.

Assuming that $q = [_i \underline{X} *_j \beta | \vec{h}]_{\underline{Z}}, \underline{X} = X/x, \underline{Z} = Z/z$, and $\vec{h} = (G, L_1/\ell_1, L_2/\ell_2)$, the following parameterizations were chosen:

$$P(\text{SHIFT}(w)|q) \simeq \begin{cases} p_s(w|\beta, x, \ell_1) & \text{if } \beta \neq \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$P(\text{PROJECT}(U,\delta)|q) \simeq \begin{cases} p_p(U,\delta|G,Z,X,z) & \text{if } \beta = \varepsilon \text{ and } Z \neq W \\ p_t(U,\delta|z,G,L_1) & \text{if } \beta = \varepsilon \text{ and } Z = W \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$P(\text{ATTACH}|q) \simeq \begin{cases} p_a(\text{ATTACH}|G,Z,X,z) & \text{if } \beta = \varepsilon \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

The conditioning attributes are ordered from most to least significant. This means that

the back-off sequences are:

$$\begin{aligned}
\text{for } p_s(w|\ldots): &\quad \beta,x,\ell_1 \to \beta,x \to \beta \to \text{(empty)}\\
\text{for } p_p(U,\delta|\ldots): &\quad G,Z,X,z \to G,Z,X \to G,Z \to G \to \text{(empty)}\\
\text{for } p_t(U,\delta|\ldots): &\quad z,G,L_1 \to z,G \to z \to \text{(empty)}\\
\text{for } p_a(\text{ATTACH}|\ldots): &\quad G,Z,X,z \to G,Z,X \to G,Z \to G \to \text{(empty)}.
\end{aligned}$$

Note that projections from W constituents (essentially, part-of-speech tagging) employ a parameterization different from the other projections; p_t is called the *tagger* submodel.
If p_p and p_a have the same parameterization, as in this case, p_p and p_a can be combined conveniently in one model p_{pa}:

$$p_{pa}(ATT|G,Z,X,z) = p_a(\text{ATTACH}|G,Z,X,z) \tag{4.4}$$
$$p_{pa}(U,\delta|G,Z,X,z) = p_p(U,\delta|G,Z,X,z)(1-p_{pa}(ATT|G,Z,X,z)), \tag{4.5}$$

which considers ATT just as a special (U,δ).

Parameterizations (4.1), (4.2) and (4.3) were determined manually by optimizing the *conditional perplexities* (CPPL) on the development set (sections 21–22) obtained with an initial model trained on sections 00–20 with a certain parameterization. The concept of CPPL was introduced by [Chelba, 2000]. For the shift model $p_s(w|\beta,x,\ell_1)$, for instance, it can be defined as

$$\text{CPPL} = \exp \frac{\sum_{w,\beta,x,\ell_1} c_D(\text{SHIFT}(w),\beta,x,\ell_1) \ln p_s(w|\beta,x,\ell_1)}{\sum_{w,\beta,x,\ell_1} c_D(\text{SHIFT}(w),\beta,x,\ell_1)}, \tag{4.6}$$

where D is a measurement corpus of parse trees, decomposed into LC parser moves, and $c_D(\text{SHIFT}(w),\beta,x,\ell_1)$ is the frequency in D of the $\text{SHIFT}(w)$ move from any node $[_i X/x* _j \beta|G,L_1/\ell_1,L_2/\ell_2]_{\underline{Z}}$ for some $\underline{Z},i,j,X,G,L_1,L_2,\ell_2$.
The CPPLs of the selected parameterizations on the development set are listed in Table 4.1. Both smoothing methods lead to the same parameterizations, since apart from a small scaling factor CPPLs appeared to evolve similarly. There are two versions of each submodel: the first using GT smoothing, the second using KN smoothing (cf. Table 4.2). One way of interpreting the CPPL of p_s is as a lower bound for the perplexity that can be achieved by the PLCG-based language model. It would be the perplexity of a language model that always builds correct parse trees with probability 1.

In an attempt to automate the parameterization process, also the following greedy optimization procedure was considered:

1. Start with an empty context.
2. For each parameter that is not yet in the context, evaluate the decrement of the CPPL on the development set when that parameter is added as the least significant item to the context.

Table 4.1. Conditional perplexities of selected submodel parameterizations on the PTB development set.

submodel	smoothing	CPPL
$p_s(w\|\beta,x,\ell_1)$	GT	35.2
	KN	31.1
$p_{pa}(U,\delta\|G,Z,X,z)$	GT	1.88
	KN	1.80
$p_t(U,\delta\|z,G,L_1)$	GT	1.20
	KN	1.15

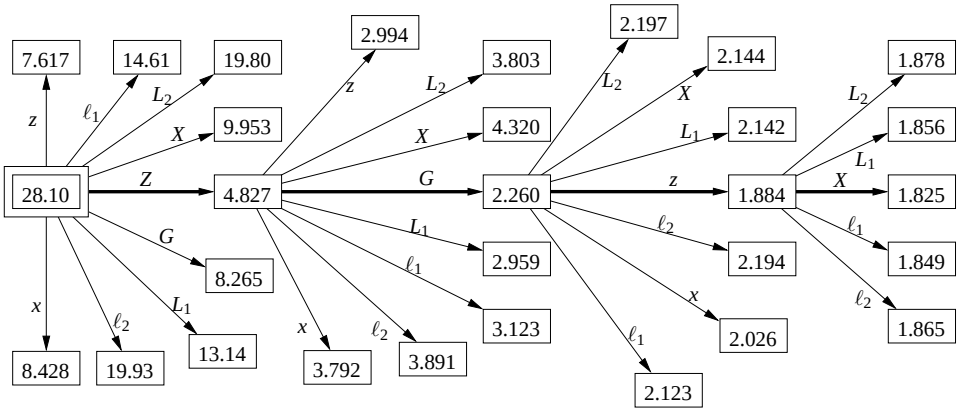

Figure 4.5. Greedy parameterization optimization of the GT smoothed p_{pa} model.

3. If the largest decrement of the CPPL is larger than a preset threshold value, add the corresponding parameter to the context as the least significant item. Else terminate.
4. Go back to step 2.

This procedure was repeated for each submodel with Kneser-Ney smoothing and Good-Turing discounting. The results were identical to the manually selected parameterizations, except for the project model, for which in both smoothing schemes a parameterization $p_p(U,\delta|Z,G,z,X)$ was found, which performed slightly worse than (4.2).
In Fig. 4.5, the greedy parameterization optimization of the GT smoothed p_{pa} is visualized as an optimal path search through a tree graph: nodes are labeled with CPPL, a transition represents adding a feature as the least significant item to the context.

Note: The shift submodel in the C&J model can actually be emulated by choosing the parameterization $p_s(w|X,x,L_1,\ell_1)$ — in other words, by omitting β and adding X and L_1 to the conditioning context. The CPPL obtained with this parameterization is

Table 4.2. Evaluated submodel smoothing techniques. They are explained in more detail in Sec. 1.5 and Sec. 1.6.

Abbrev.	Discounting	Back-off strategy	Reference
GT	Good-Turing	non-linear back-off	[Katz, 1987]
KN	absolute	linear Kneser-Ney	[Chen and Goodman, 1998]
DI	linear, context-dep.	linear deleted interp.	[Jelinek and Mercer, 1980]

109.5 using GT smoothing, in contrast with the CPPL of 35.2 with parameterization (4.1). This clearly illustrates the relevance of β for conditioning the next word in the PLCG-based model. In the C&J model, there is no such β.

On the other hand, the reduction of shift ambiguity by β is counteracted by the ambiguity of β itself, since it is stochastically predicted by the project submodel. By comparing the CPPL of 1.88 with the project submodel with the CPPL of 1.54 with the 'parser' (i.e., reduce) submodel of the C&J model [Chelba, 2000, Table 4.7], one can conclude that projecting in the PLCG-based model is significantly more ambiguous than reduction in C&J. This is actually logical since the project submodel jointly predicts 2 or more labels (of the mother and sister nodes). The reduce submodel only predicts the mother category, based on the syntactic and lexical labels of both daughters; in this way, it makes use of considerably more bottom-up information than the project submodel. Fortunately, the latter can reduce ambiguity considerably with the top-down feature *G*; without it, the CPPL would be 4.23 instead of 1.88 (GT smoothing).

Initial models

Once submodel parameterizations are fixed, each tree-annotated sentence from the training set is decomposed in its elementary LC derivation steps. Each step corresponds with an *n*-gram event. p_s, p_{pa} and p_t are initialized using conventional *n*-gram language modeling techniques. Three smoothing techniques were compared, as listed in Table 4.2.

Note that a *different* smoothing technique can be chosen for each submodel. For instance, GT for the project submodel, and KN for the shift submodels. This additional degree of freedom was not explored in my experiments, however.

Baseline model

Word-based 3-gram models were trained on the running text of the PTB training set using the smoothing techniques listed in Table 4.2 for comparison with the corresponding PLCG-based LMs. The 3-gram models are also used for interpolation with the PLCG-based LM.

Table 4.3. Influence of pruning parameters on PPL and execution time. Table entries are in the format PPL/PPLi/time, where PPLi is obtained after interpolation with the baseline 3-gram model, and time is total execution time on a PIII/930MHz PC. Both the baseline 3-gram model and the PLCG-based LM are trained with KN smoothing and tested on the verbalized punctuation data.

	$\rho^o = 10^3$	$\rho^o = 10^{3.5}$	$\rho^o = 10^4$	$\rho^o = 10^{4.5}$
$\sigma = .7$	130.5/100.7/0:13	113.2/97.5/0:20	106.4/96.0/0:35	104.9/95.5/1:06
$\sigma = .5$	112.4/97.6/0:21	105.8/96.0/0:37	103.4/95.5/1:18	102.6/95.3/3:02
$\sigma = .3$	106.0/96.2/0:37	103.2/95.5/1:21	102.4/95.3/3:23	102.5/95.3/9:25

4.1.2. Measurements

Data

The measurement set consists of sections 23 and 24 of the Penn Treebank. This test set is very common in large-scale stochastic parsing literature. I have prepared a 'verbalized punctuation' (vp) and a 'non-verbalized punctuation' (nvp) version, to be used with their corresponding models.

Influence of pruning

The pruning parameters ρ^o and σ (cf. Sec. 3.4.5) have an important influence on the runtime behavior of the PLCG-based LM. Small ρ^o and large σ lead to fast execution but inaccurate evaluation.

In Table 4.3, test set perplexities and execution times measured with varying pruning settings are collected in a matrix. The measured PPL at very loose pruning settings is interpreted as the 'real' PPL, and the difference with the real PPL at practical (i.e., tighter) pruning settings as the inaccuracy of the evaluation. There seems to be a rather strong dependence between ρ^o and σ, given a target PPL/time trade-off. In the next experiments, the pruning parameters were fixed to $\rho^o = 10^{3.5}$ and $\sigma = .5$.

Let M_i denote the number of calls to the shift submodel at a word position i. M_i gives an idea of the ambiguity of the partial parse trees faced by the language model at position i. Fig. 4.6 shows the average and the standard deviation of M_i under 4 different pruning settings, measured on PTB sections 21–22 (nvp).

Influence of smoothing

Table 4.4 reports test set PPLs of differently smoothed PLCG-based LMs (cf. Table 4.2). The PLCG-based LMs used in these experiments were not reestimated.

One notes that the DI and GT smoothed models do not significantly differ in performance; KN smoothing, however, outperforms GT and DI smoothing with a significant PPL reduction of roughly 10%.

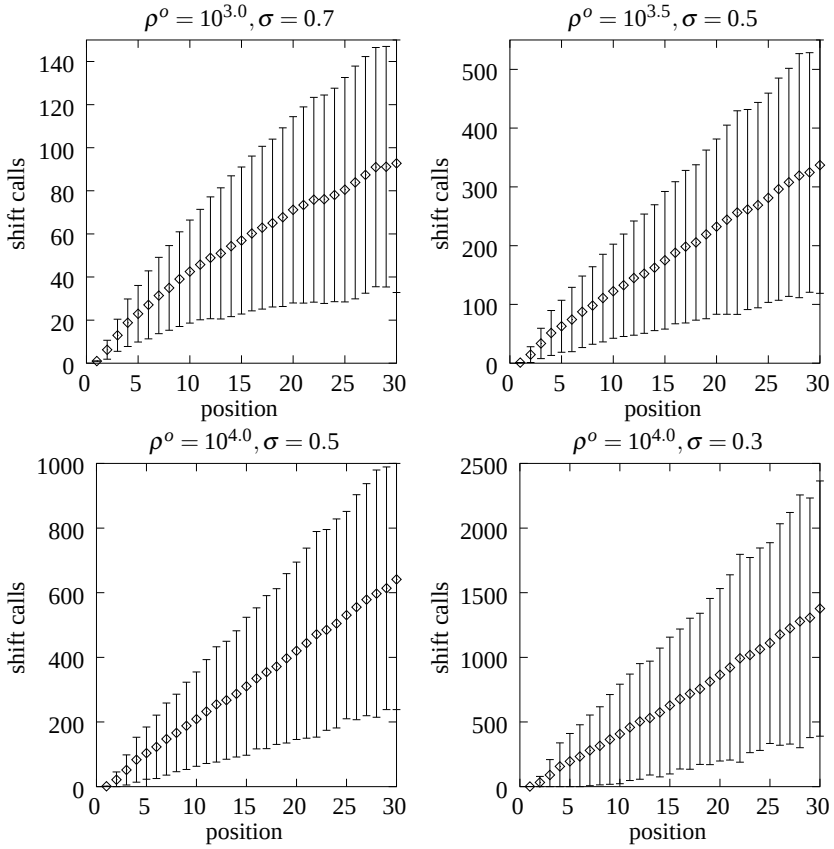

Figure 4.6. Estimated mean and standard deviation of the number of shift submodel calls at a given word position under three different pruning settings.

Table 4.4. Test set PPL of differently smoothed PLCG-based LMs. The '3g+PLCG' rows are obtained by interpolating the PLCG-based LM interpolated with the baseline 3-gram LM where the interpolation weight of the baseline 3-gram LM is .4.

PTB 23–24 nvp	DI	GT	KN
baseline 3-gram	194	191	173
PLCG-based LM	175	174	154
3g+PLCG	164	161	145
PTB 23–24 vp	DI	GT	KN
baseline 3-gram	129	126	114
PLCG-based LM	118	119	106
3g+PLCG	108	107	96

One can also conclude from the vp experiment series that the unreestimated PLCG-based LM improves on the 3-gram PPL by rougly 6%, and by 15% when interpolated with the 3-gram. For the nvp series, the measured PPL improvements are 10% and 15%, respectively.

Influence of reestimation

Reestimation is quite costly, because each iteration requires processing the training corpus with a PLCG-based language model. For instance, at 50 words/sec on a PIII/930Mhz PC, one reestimation iteration on a 1M word training corpus takes about 5.5 hours. Fortunately, it is easy to parallellize the reestimation over multiple processors.

Reestimating the model on the PTB training corpus did not deliver notable performance gains. The reestimation results are collected in Table 4.5. Note the discrepancy between the training set and test set PPLs, already present in the initialization phase.[1]

The training set PPL tends to drop slightly and to converge after 1 iteration; however, monotonicity is not guaranteed because the smoothing of the expected transitions counteracts the likelihood maximization. As a sanity check, it was verified that reestimation without smoothing of the expected frequencies does yield a 'normal' evolution of the training set PPL; this is indicated by the 'PTB vp, —' figures at the bottom of Table 4.5, which were obtained starting from KN smoothed initial submodels but without smoothing during reestimation.

The development and test set PPLs, increase significantly at the first iteration with GT smoothing. The test PPLs obtained with KN smoothing seem rather random. it can be concluded that PLCG reestimation does not justify the considerable extra training effort — at least not with such a small training corpus as the PTB.

The reason of failure is unclear. A possible cause is that KN and GT smoothing are actually non-continuous techniques; using them here for counteracting EM overtraining is not theoretically justified. Another possible problem is the relatively small size of the Penn Treebank, and its high quality, such that the reestimation problem is rather ill-conditioned.

Comparison with other syntax-based LMs

The standard choice of the training and test set makes it possible to compare the PPL performance of the PLCG-based LM quantitively with other recent grammar-based language models.

Results extracted from publications and from my own experiments are collected in Table 4.6. The table separates results obtained with different smoothing techniques in different columns, since smoothing affects PPL considerably, as noted above. In

1. The training set PPL versus test set PPL discrepancy of the word-based 3-grams is actually even worse: 21 versus 126 for the GT-smoothed 3-gram, and 21 versus 114 for the KN-smoothed 3-gram.

Table 4.5. Influence of reestimation on PPL.

PTB vp, KN	initial	after 1 it.	after 2 it.
Train set PPL	26.1	25.4	25.2
Dev set PPL	103.4	100.6	102.3
Test set PPL	106.0	107.1	109.0
PTB vp, GT	initial	after 1 it.	after 2 it.
Train set PPL	31.7	19.3	19.3
Dev set PPL	114.5	134.3	130.2
Test set PPL	119.0	144.2	140.1
PTB nvp, KN	initial	after 1 it.	after 2 it.
Train set PPL	32.5	31.9	31.9
Dev set PPL	151.5	152.4	156.0
Test set PPL	154.0	162.0	166.1
PTB nvp, GT	initial	after 1 it.	after 2 it.
Train set PPL	40.8	22.3	23.3
Dev set PPL	169.4	201.7	198.0
Test set PPL	174.0	217.0	212.5
PTB vp, —	initial	after 1 it.	after 2 it.
Train set PPL	26.1	9.6	8.9
Dev set PPL	103.4	388.7	393.4
Test set PPL	106.0	428.0	435.7

addition, test set PPLs were recalculated with *<unk>* probabilities *included*, in order to enable comparison with [Chelba, 2000, Roark, 2001, Charniak, 2001, Kim et al., 2001].[2]

The lowest PPL (including *<unk>*) in the table is 126. It is obtained with a KN smoothed PLCG-based LM, interpolated with the baseline word-based trigram. C&J with KN smoothed probabilities [Kim et al., 2001] comes close to the best result obtained with the PLCG-based LM (130 versus 126). This model was reestimated, while the PLCG-based LM was not. Since initialization on the PTB only takes a few minutes, building a PLCG-based LM requires a tiny fraction of the time needed for building a C&J model that has comparable performance (since one reestimation iteration takes several hours).

Charniak's model [Charniak, 2001] reaches the same performance with DI smoothed probability distributions; it remains an open question whether KN smoothing may improve Charniak's model further. An important drawback of Charniak's model is that it cannot be interpolated with other models at the word level.

Roark's LM [Roark, 2001] shows some potential too; again, improved smoothing

2. I believe, though, that the PPL excluding *<unk>* probabilities, as reported elsewhere in this chapter, is more predictive for speech recognition performance.

Table 4.6. Comparing PPLs (on PTB 23–24 nvp) obtained with other grammar-based LMs. The C&J models are reestimated with 3 EM iterations. The PLCG-based LM is not reestimated.

	DI		GT		KN	
PPL without *<unk>*	PPL	PPLi	PPL	PPLi	PPL	PPLi
word-based 3-gram	194	194	191	191	173	173
PLCG-based LM	175[a]	164[a]	174	161	154	145
C&J LM	187[b]	174[a]				
PPL with *<unk>*	PPL	PPLi	PPL	PPLi	PPL	PPLi
word-based 3-gram	167	167	166	166	156	156
PLCG-based LM	151	139	150	138	133	**126**
C&J LM	153[c]	147[c]			141[d]	130[d]
Roark LM	152[e]	137[e]				
Charniak LM	130[f]	126[f]				

[a] Experiments run by F. Van Aelten and K. Daneels at L&H; [b] obtained with a reimplementation [Van Aelten and Hogenhout, 2000] of [Chelba, 2000]; [c] as reported in [Chelba, 2000, p. 49]; [d] as reported in [Kim et al., 2001]; [e] as reported in [Roark, 2001, p. 270]; [f] as reported in [Charniak, 2001].

might reduce the PPL of 137, but no such results were reported thus far. Moreover, the PPL of 126 of the PLCG-based model was measured at a rather tight pruning setting; at equal execution speeds on comparable computers (30 words/s), Roark's model marked a PPL of 141.

4.2. Rescoring speech recognition hypotheses lists

A second series of experiments evaluates the PLCG-based LM in a more realistic dictation setting. Models are trained on a large corpus and employed for rescoring transcription hypotheses in a recognition task.

4.2.1. Modeling

The models are trained on the BLLIP-WSJ corpus plus sections 0–20 of the PTB. The BLLIP-WSJ corpus is the ACL/DCI Wall Street Journal '87–'89 corpus that was machine-parsed by the BLLIP lab at Brown University and distributed by the LDC [Charniak, 2000]. BLLIP-WSJ has an annotation style and tag set that is very similar to PTB's, but is about 35 times larger. On the other hand, it contains more parsing errors.

Care was taken that the speech recognition test set was excluded from the training data. All the submodels and the baseline word-based 3-grams were slimmed down

Table 4.7. Word error rates on eval92 obtained with GT smoothed models. '+' indicates linear interpolation.

model	WER
word 3-gram	7.98
PLCG0 + word 3-gram	7.26
PLCG2 + word 3-gram	7.03

by omitting all maximum-order events that appeared less than twice in the training corpus.

The submodels of the PLCG-based LM were parameterized in the same way as in the PTB experiment series. GT smoothing was used for initialization, yielding model PLCG0, as well as for two EM iterations, yielding models PLCG1 and PLCG2, respectively.

Additionally, a GT smoothed word-based trigram was trained on the BLLIP-WSJ. This model differs from the standard WSJ-trained trigrams in the tokenization (e.g., *don't* is replaced with *do n't* and numbers are replaced with *N*). Also, the BLLIP-WSJ does not contain all the data from the WSJ corpus because sentences that could not be machine-parsed in reasonable time were left out.

In cooperative work done at L&H, a DI smoothed class-based 4-gram model was trained with automatically generated word classes. This model was used to assess complementarity of the word class model with other grammar-based models.

4.2.2. Word error rate

The DARPA WSJ November 1992 LVCSR test suite (20k open vocabulary, verbalized punctuation) was used for testing recognition performance of the PLCG-based LM. 100-best lists were generated for both the evaluation (eval92) and development (dev92) test sets using L&H VoiceXpress v4, a mainstream speech recognizer based on context-dependent hidden Markov models of phonemes, represented as time series of mel-scaled cepstral feature vectors; for the language model, the standard word trigram was used. The 100-best lists were first preprocessed to match the tokenization of the BLLIP-WSJ models. These lists were then rescored with the language models under scrutiny. The dev92 set was used for finding optimal model interpolation weights.

Table 4.7 collects WER results with GT smoothed models. The un-reestimated PLCG-based LM yields a relative improvement of 9% with respect to the word 3-gram. A small but consistent improvement by EM reestimation is observed: the reestimated PLCG-based LM yields a relative improvement of 12% below the word 3-gram baseline WER. (This corrects the results reported in [Van Uytsel et al., 2001], which were affected by bugs in my rescoring code.)

Table 4.8. Word error rates on eval92 obtained with DI smoothed models. '+' indicates linear interpolation. The C&J model was reestimated with 3 EM iterations. Measurements by Filip Van Aelten and Kristin Daneels at L&H.

model	WER
word 3-gram	7.88
word 3-gram + class 4-gram	7.37
C&J	7.47
C&J + word 3-gram	7.31
C&J + word 3-gram + class 4-gram	7.08
PLCG0	7.08
PLCG0 + word 3-gram	7.06
PLCG0 + word 3-gram + class 4-gram	6.91

Comparative experiments using DI smoothed models were done at the L&H lab by Filip Van Aelten and Kristin Daneels. Their results are summarized in Table 4.8. The DI smoothed PLCG0 model performs remarkably well, on a par with the KN smoothed PLCG2 model, in spite of the previously perceived inferiority of DI smoothing in perplexity measurements.

It was observed that the PLCG0 model successfully complements a word 3-gram *and* a class 4-gram; the C&J model does so too, but its performance is slightly worse than the PLCG0 model's, although the significance of the difference is disputable.

4.2.3. Grammaticality

In automatic speech recognition, the expected word error rate is minimized by minimizing the expected sentence error rate; the latter is minimized by maximizing the a posteriori probability of the transcript.

By its very nature, word error rate is much more correlated with sentence error rate than with grammaticality. Minimizing the number of ungrammaticalities in the recognition output, for example, in fact *harms* the accuracy.

It may be, however, useful to recall that the motivation for rescoring with a grammar-based language model is to improve accuracy, *not* by improving the grammaticality of the recognition output, but by improving the accuracy of its probability estimates; the latter is realized by improving the *grammaticality of the probabilistic dependencies* on which the estimates are based.

A qualitative comparison of transcripts obtained with a 3-gram model, with those obtained with a PLCG-based model, reflected this intended behavior, and did not reflect the non-intended behavior. No obvious difference in grammaticality was found. Though, examples of the advantage of grammar-based language modeling can be found at several isolated spots.

I will illustrate this with one example. In the (partial) hypothesis *many of them has made mistakes*, the word *has* gets assigned a much lower probability by the PLCG0 model then by the word 3-gram (the scores are $\log_{10}$ of conditional probabilities):

	many	*of*	*them*	*has*	*made*	*mistakes*	*.*	*</s>*
3-gram:	-2.481	-0.728	-0.938	**-2.079**	-2.609	-2.855	-0.645	-0.082
PLCG0:	-2.481	-0.792	-0.926	**-3.631**	-1.927	-3.355	-0.449	-0.012

On the other hand, *have* instead of *has* gets assigned a much higher probability by the PLCG0 model; this difference is less convincing with the 3-gram model:

	many	*of*	*them*	*have*	*made*	*mistakes*	*.*	*</s>*
3-gram:	-2.481	-0.728	-0.938	**-1.469**	-2.640	-2.682	-0.645	-0.082
PLCG0:	-2.481	-0.792	-0.926	**-0.900**	-2.063	-3.115	-0.434	-0.016

Closer inspection reveals that the score of -0.900 is composed for 97% from shift probabilities conditioned on $w_0 =$ *<s>* and $w_1 =$ *many*, which is desired, and only for 2% from shift probabilities conditioned on $w_2 =$ *of* and $w_3 =$ *them*, which are more influenced by the randomness of the training corpus. This example also illustrates that the PLCG0 model is better at recognizing the end of a sentence.

4.3. Summary

The PLCG-based LM was found to be a competitive alternative among the class of syntax-based language models. Test set perplexities and word error rates compare favorably with, for instance, [Chelba, 2000], [Roark, 2001] and [Charniak, 2001]. From the comparison of execution times, I believe that the PLCG-based LM is more efficient than the other cited models. It was observed that the un-reestimated PLCG-based LM performed at least as well as the reestimated C&J model, while building the former model requires only a tiny fraction of the time needed to train the latter. Reestimating Penn Treebank models did not improve PPL. However, reestimated BLLIP-WSJ models did yield an additional WER reduction of 3% relative.

I believe that a great part of efficiency is gained by representing the search space as a minimal network instead of a search tree, as in [Chelba and Jelinek, 1999] and [Roark, 2001]. Given the same computational resources, more probabilistic analyses can be accounted for using the dynamic programming technique. A disadvantage of this, however, is that the parser is less flexible at extracting the relevant conditioning information from a partial analysis. That problem was solved by extending the nodes with all features that were expected to be informative in selecting the next parse move.

A final suggestion for future work is about the detail of the underlying grammar. Current syntax-based language models rely on an overly simplistic version of phrase structure grammar, namely the one that is most readily extracted from the Penn Treebank and its relatives. A greater degree of generalization can be obtained by processing syntactic features (such as tense, gender, number, finiteness), for instance in a unification-

based style, instead of only one category label. The integration of a morpho-syntactic analysis stage within the parsing system could introduce those features into the syntactic analysis of the whole sentence; the linguistically sound treatment of previously unseen wordforms would come as a welcome side-effect.

CHAPTER 5

Conclusion and perspectives

5.1. Original contributions

The research described in this thesis combines elements of, and builds further on two domains that are traditionally separated: computational linguistics and statistical learning. More specifically, syntactic theory, in the form of hand-parsed text material (treebank), was used as *a priori* knowledge for the initialization of a novel statistical language model, the PLCG-based language model.

The operation of the PLCG-based language model is based on an efficient original implementation of statistical left corner parsing. The parser uses a dynamic programming technique to improve time and space efficiency. Inspired by inside-outside reestimation of PCFG probabilities, we defined forward, inner and outer probabilities of nodes in a network, borrowing ideas from inside-outside reestimation of PCFG probabilities and Bayesian networks. Mutually recursive relations were derived for these probabilities, which enables the model to emit sentence prefix probabilities $P(w_0^i)$, conditional left-to-right language model probabilities $P(w_i|w_0^{i-1})$, and expected frequencies of moves. The latter are needed for the recursive maximum-likelihood reestimation of the language model. A reestimation algorithm was proposed as an instance of the general EM algorithm.

As to the linguistic side of the language model, the grammar used is a simple phrase-structure grammar lacking more modern linguistic concepts such as unification and indexing. On the other hand, the parser allows for extensive non-local probabilistic conditioning of parser move probabilities on lexical and syntactic labels of previously hypothesized constituents. Experimentally, this non-local conditioning was found to improve the accuracy and the efficiency of the PLCG-based language model, by reducing the average ambiguity. The conditional move probability distributions are initialized on a treebank, and statistically smoothed. The linear Kneser-Ney smoothing technique turned out superior to Good-Turing and deleted interpolation smoothing.

However, in contrast with deleted interpolation, Kneser-Ney smoothing is theoretically not justified for estimating probabilities from non-integer counts, as is needed in our reestimation procedure. Still, we obtained slight improvements with it in our speech recognition experiments. It would be interesting to evaluate a continuous version of Kneser-Ney smoothing, once it gets developed.

Experiments were run with small and large language models, respectively trained on the Penn Treebank and the BLLIP WSJ corpus. The small models were used for perplexity/cross-entropy measurements, the larger were used as n-best list rescoring models in a read newspaper speech recognition task.

We measured a cross-entropy reduction of 0.32 bits (20% perplexity reduction) and a relative word error reduction of 12% from a word-based trigram baseline. Compared with a baseline consisting of a word-based 3-gram and a class-based 4-gram, it was also found that interpolating the baseline with our model yields a relative word error rate reduction of 6.2%.

5.2. Perspectives

Despite its remarkable performance and its improved efficiency with regard to other syntax-based language models, the PLCG-based language model is probably still too heavy to be used in practical products. Though, it may be a starting point or a source of inspiration for more efficient syntax-based language models e.g. exploiting shallow parsing techniques.

Our intuition tells us that a great deal of parse ambiguity is not inherent, but due to too coarse modeling assumptions. We therefore expect a further increase of efficiency by (a) refining the syntactic structure annotation, and (b) integrating external knowledge.

Refining the syntactic structure annotation is obvious, given the simplicity of the syntactic theory assumed in our language model. Practical problems may arise, however, since the additional syntactic information has to be made available by hand in the training data in case the extra information cannot be automatically obtained with a set of deterministic rules.

External knowledge can be integrated from a lower level (prosody, morphology), as well as from a higher level. The integration of a morphological component in the PLCG-based language model is a topic of FLaVoR, an IWT-funded research project currently running at the Katholieke Universiteit Leuven (ESAT) and the University of Antwerp (CNTS). The main motivation is that a morphological analysis will facilitate propagating bottom-up information from the word level, such as agreement features, thereby reducing ambiguity. Especially the linguistically justified handling of out-of-vocabulary words is interesting with respect to robustness. This should also mitigate the problem of the reduced accuracy and efficiency of our PLCG-based language model on sentences containing out-of-vocabulary words.

It is also possible to let higher level information influence the parsing process. For example, this information may consist of expectations from a dialog model about the sentence type. Only a small portion of the probability parameters are to be explicitly conditioned on this information, which keeps the model trainable.

Bibliography

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers, principles, techniques, and tools*. Addison-Wesley, Reading, MA, USA, 1986.

L. Bahl, J. Baker, P. Cohen, A. Cole, F. Jelinek, B. Lewis, and R. Mercer. Automatic recognition of continuously spoken sentences from a finite state grammar. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 418–421, 1978.

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(7):1001–1008, July 1989. Also in: Waibel, A. and Lee K.-F. (eds), Readings in Speech Recognition. Morgan Kaufmann.

James K. Baker. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550. The MIT Press, Cambridge, MA, 1979.

Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. DARPA Workshop on Speech and Natural Language Processing*, pages 306–311. Morgan Kaufman, San Mateo, CA., 1991.

Ezra Black, Frederick Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. 31th Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 134–139, 1993.

Rens Bod. Combining semantic and syntactic structure for language modeling. In *Proc. International Conference on Spoken Language Processing 2000*, volume 3, pages 106–109, Beijing, China, 2000.

J. Bresnan, editor. *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, USA, 1982.

Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, Oxford, 2001.

Ted Briscoe and John Carroll. Generalized LR parsing of natural language (corpora) with unification-based methods. *Computational Linguistics*, 19:25–59, 1993.

Michael K. Brown and Stephen C. Glinski. Stochastic context-free language modeling with evolutional grammars. In *Proc. International Conference on Spoken Language Processing*, pages 779–782, Yokohama, Japan, 1994.

P. F. Brown, V. J. Della Pietra, P. V. de Souza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), 1992.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2), 1993.

John Caroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: A survey and a new proposal. In *Proc. 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain, 1998.

J.-C. Chappelier, M. Rajman, R. Aragües, and A. Rozenknop. Lattice parsing for speech recognition. In *Proc. 6me Conf. sur le Traitement Automatique du Langage Naturel (TALN'99)*, pages 95–104, 1999.

Eugene Charniak. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University, 1996.

Eugene Charniak. A maximum-entropy inspired parser. In *Proc. 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, 2000.

Eugene Charniak. Immediate-head parsing for language models. In *Proc. 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, 2001.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. Edge-based best-first chart parsing. In *Proc. 6th Workshop on Very Large Corpora*, pages 127–133, Montreal, Canada, 1998.

Ciprian Chelba. *Exploiting Syntactic Structure for Natural Language Modeling*. PhD thesis, Johns Hopkins University, 2000.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor M. Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. Structure and performance of a dependency language

model. In *Proc. European Conference on Speech Communication and Technology 1997*, pages 2775–2778, Rhodes, Greece, 1997. URL `citeseer.nj.nec.com/article/chelba97structure.html`.

Ciprian Chelba and Frederick Jelinek. Recognition performance of a structured language model. In *Proc. European Conference on Speech Communication and Technology 1999*, volume 1, pages 1567–1570, 1999.

Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech and Language*, 14:283–332, 2000.

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR–10–98, Harvard University, August 1998.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.

Stanley F. Chen and Ronald Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

Y. L. Chow, M. O. Dunham, O. A. Kimball, M. A. Krasner, G. F. Kubala, J. Makhoul, P. J. Price, and S. Roukos. BYBLOS: The BBN continuous speech recognition system. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 89–92, 1987.

Yen-Lu Chow and Salim Roukos. Speech understanding using a unification grammar. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 727–730, 1989.

Michael J. Collins. Three generative lexicalised models for statistical parsing. In *Proc. 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, pages 16–23, 1997.

J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

S. Della Pietra, V. Della Pietra, J. Gillet, J. Lafferty, H. Printz, and L. Ureš. Inference and estimation of a long-range trigram model. In *2nd Int. Coll. Grammatical Inference and Applications (ICGI-94)*, 1994.

Stephen Della Pietra and Victor Della Pietra. Statistical modeling by ME. Technical report, IBM, 1993.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society, Series B*, 39:1–38, 1977.

A. M. Derouault and B. Mérialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8: 742–749, November 1986.

Anne-Marie Derouault and Bernard Merialdo. Probabilistic grammar for phonetic to french transcription. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 1577–1580, 1985.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

Petra Geutner. Introducing linguistic constraints into statistical language modeling. In *Proc. International Conference on Spoken Language Processing*, volume 1, pages 402–405, Philadelphia, PA, USA, 1996.

Petra Geutner. Fuzzy class rescoring: A part-of-speech language model. In *Proc. European Conference on Speech Communication and Technology*, volume 1, pages 2743–2746, Rhodos, Greece, 1997.

David Goddeau and Victor Zue. Integrating probabilistic LR parsing into speech understanding systems. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 181–184, 1992.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3–4):237–264, 1953.

Joshua Goodman. Probabilistic feature grammars. In *Proc. 5th International Workshop on Parsing Technologies*, 1997.

Joshua T. Goodman. A bit of progress in language modeling. *Computer Speech and Language*, 15(4):403–434, 2001a.

Joshua T. Goodman. A bit of progress in language modeling. Technical Report MSR–TR–2001–72, Microsoft, 2001b.

T. V. Griffiths and S. R. Petrick. On the relative efficiencies of context-free grammar recognizers. *Communications of the ACM*, 8(5):289–300, 1965.

Peter Heeman and James F. Allen. Incorporating POS tagging into language modeling. In *Proc. European Conference on Speech Communication and Technology*, pages 2767–2770, Rhodos, Greece, 1997.

Charles Hemphill and Joseph Picone. Speech recognition in a unification grammar framework. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 723–726, 1989.

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

John E. Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2 edition, 2001. ISBN 0-201-44124-1.

Tim Howells, David Friedman, and Mark Fanty. Broca, an integrated parser for spoken language. In *Proc. International Conference on Spoken Language Processing*, pages 325–328, 1992.

K. Inui, V. Sornlertlamvanich, H. Tanaka, and T. Tokunaga. A new formalization of probabilistic GLR parsing. In *Proc. 5th International Workshop on Parsing Technologies*, pages 123–134, 1997.

Eric Jackson. Integrating two complementary approaches to spoken language understanding. In *Proc. International Conference on Spoken Language Processing*, pages 333–336, 1992.

E. Jaynes. Information theory and statistical mechanics. *Physics Reviews*, 106:620–630, 1957.

F. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann Publishers, 1990.

F. Jelinek, R. L. Mercer, and L. R. Bahl. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190, March 1983.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA, 1997.

Frederick Jelinek and Ciprian Chelba. Putting language into language modeling. In *Proc. European Conference on Speech Communication and Technology 1999*, volume 1, pages KN–1–6, 1999.

Frederick Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In E. S. Geltsema and L. N. Kanal, editors, *Pattern Recognition in Practice*. North Holland, Amsterdam, 1980.

Frederik Jelinek and John Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3): 315–323, 1991.

Mark Johnson. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proc. 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'98)*, pages 619–623, 1998.

Mark Johnson and Brian Roark. Compact non-left recursive grammars using the selective left-corner transform and factoring. In *Proc. 18th Conference on Computational Linguistics (COLING'00)*, pages 355–361, 2000.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 189–192, 1995.

Daniel S. Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice-Hall, 2000. ISBN 0-13-095069-6.

Atsuhiko Kai and Seiichi Nakagawa. A frame-synchronous continuous speech recognition algorithm using a top-down parsing of context-free grammar. In *Proc. International Conference on Spoken Language Processing*, pages 257–260, 1992.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35:400–401, March 1987.

P. Kay and C. J. Fillmore. Grammatical constructions and linguistic generalizations: The What's X Doing Y? construction. *Language*, 75(1):1–33, 1999.

Woosung Kim, Sanjeev Khudanpur, and Jun Wu. Smoothing issues in the structured language model. In *Proc. European Conference on Speech Communication and Technology 2001*, pages 717–720, Aalborg, Danmark, 2001.

Kenji Kita and Wayne H. Ward. Incorporating LR parsing into SPHINX. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 269–272, 1991.

Dietrich Klakow. Log-linear interpolation of language models. In *Proc. International Conference on Spoken Language Processing*, volume 2, pages 701–704, Sydney, Australia, 1998.

Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *Proc. European Conference on Speech Communication and Technology*, pages 973–976, Berlin, Germany, 1993.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 181–184, 1995.

Reinhard Kneser and Jochen Peters. Semantic clustering for adaptive language modeling. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 779–782, Munich, Germany, 1997.

K. Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1): 93–124, 1989.

Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (6):570–583, June 1990.

S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical trigrams: A probabilistic model of link grammar. Technical Report CMU-CS-92-181, Carnegie Mellon University, Pittsburgh, PA, USA, 1992.

Bernard Lang. Deterministic techniques for efficient non-deterministic parsers. In *Proc. 2nd Coll. Automata, Languages and Programming*, pages 255–269, Saarbrücken, 1974. Springer.

R. Lau, R. Rosenfeld, and S. Roukos. Trigger-based language models: A maximum entropy approach. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 45–48, April 1993.

Alon Lavie and Masaru Tomita. GLR* — an efficient noise-skipping parsing algorithm for context free grammars. In *Proc. 3rd International Workshop on Parsing Technologies*, 1993.

B. Lowerre and R. Reddy. The HARPY speech understanding system. In W. A. Lea, editor, *Trends in Speech Recognition*. Prentice-Hall, 1980.

David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University, 1994.

C. D. Manning and H. Schütze, editors. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.

Christopher D. Manning and Bob Carpenter. Probabilistic parsing using left corner language models. In *Proc. 5th International Workshop on Parsing Technologies*, pages 147–158, 1997.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

S.C. Martin, Hermann Ney, and J. Zaplo. Smoothing methods in maximum entropy language modeling. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 545–548, Phoenix, AR, USA, 1999.

Yuji Matsumoto, Masaki Kiyono, and Hozumi Tanaka. BUP: A bottom-up parser embedded in Prolog. *New Generation Computing*, 1(2):145–158, 1983.

Shoichi Matsunaga, Shigeki Sagayama, Shigeru Homma, and Sadaoki Furui. A continuous speech recognition system based on a two-level grammar approach. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 589–592, 1990.

David McAllester and Robert E. Schapire. On the convergence rate of Good-Turing estimators. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 1–6. Morgan Kaufmann, San Francisco, 2000. URL `citeseer.nj.nec.com/mcallester00convergence.html`.

R. Moore, J. Dowding, H. Bratt, J. Gawron, Y. Gorfu, and A. Cheyer. CommandTalk: A spoken-language interface for battlefield simulations. In *Proc. 5th Conf. Applied Natural Language Processing*, pages 1–7, 1997.

Robert C. Moore. Improved left-corner chart parsing for large context-free grammars. In *Proc. 6th International Workshop on Parsing Technologies*, pages 171–182, 2000.

Hy Murveit and Robert Moore. Integrating natural language constraints into HMM-based speech recognition. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 573–576, 1990.

Arthur Nádas. Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 32:859–861, 1984.

Arthur Nádas. On Turing's formula for word probabilities. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33:1414–1416, 1985.

Seiichi Nakagawa. Spoken sentence recognition by time-synchronous parsing algorithm of context-free grammar. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 829–832, 1987.

Mark-Jan Nederhof. Generalized left-corner parsing. In *Proc. 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, pages 305–314, 1993.

H. Ney, S. Martin, and F. Wessel. Statistical language model using leaving-one-out. In Steve Young and Gerrit Bloothooft, editors, *Corpus-Based Methods in Language and Speech Processing*, pages 174–207. Kluwer Academic Publishers, 1997.

Hermann Ney. Dynamic programming speech recognition using a context-free grammar. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 69–72, 1987.

Hermann Ney. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340, February 1991.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

T. R. Niesler and P. C. Woodland. A variable-length category-based n-gram language model. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 164–167, 1996.

Michio Okada. A unification-grammar-directed one-pass search algorithm for parsing spoken language. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 721–724, 1991.

Julio Pastor, José. Colás, Rubén San-Segundo, and José Manuel Pardo. An asymmetric stochastic language model based on multi-tagged words. In *Proc. International Conference on Spoken Language Processing*, volume 6, pages 2407–2410, Sydney, Australia, 1998.

Fernando C. N. Pereira and Rebecca N. Wright. Finite-state approximation of phrase structure grammars. In *Proc. 29th Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 246–255, 1991.

C. Pollard and I. A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, IL, USA, 1994.

P. Price, W.M. Fisher, J. Bernstein, and D.S. Pallett. The DARPA 1000-word Resource Management database for continuous speech recognition. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 651–654, New York, U.S.A., April 1988.

Adwait Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. In *Proc. 2nd Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Providence, R.I., USA, 1997.

Manny Rayner, Beth Ann. Hockey, Frankie James, Elizabeth Owen Bratt, Sharon Goldwater, and Jean Mark Gawron. Compiling language models from a linguistically motivated unification grammar. In *Proc. 18th Conference on Computational Linguistics (COLING'00)*, 2000.

Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.

Brian Roark and Mark Johnson. Efficient probabilistic top-down and left-corner parsing. In *Proc. 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 421–428, 1999.

Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1994.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10:187–228, 1996.

Daniel J. Rosenkrantz and Philip M. Lewis II. Deterministic left corner parsing (extended abstract). In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory*, pages 139–152, 1970.

Ernst Günther Schukat-Talamazzini, Florian Gallwitz, Stefan Harbeck, and Volker Warnke. Rational interpolation of maximum likelihood predictors in stochastic language modeling. In *Proc. European Conference on Speech Communication and Technology*, volume 5, pages 2731–2734, Rhodos, Greece, 1997.

Stephanie Seneff. TINA: A probabilistic syntactic parser for speech understanding systems. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 711–714, 1989.

Kristie Seymore and Ronald Rosenfeld. Scalable backoff language models. In *Proc. International Conference on Spoken Language Processing*, volume 1, pages 232–235, Philadelphia, PA, USA, 1996.

S. M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. University of Chicago Press, Chicago, IL, USA, 1986.

Daniel Sleator and Davy Temperley. Parsing English with a link grammar. Technical Report CMU–CS–91–196, Carnegie Mellon University, 1991.

Daniel Sleator and Davy Temperley. Parsing English with a link grammar. In *Proc. 3rd International Workshop on Parsing Technologies*, 1993.

R. M. Stern. Specification of the 1995 ARPA Hub 3 evaluation: Unlimited vocabulary NAB news baseline. In *Proc. 1996 DARPA Speech Recognition Workshop*, pages 5–7, 1996.

Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.

Andreas Stolcke. Entropy-based pruning of backoff language models. In *Proc. ARPA Human Language Technology Workshop*, 1998.

Andreas Stolcke, Ciprian Chelba, David Engle, Victor Jimenez, Lidia Mangu, Harry Printz, Eric Ristad, Roni Rosenfeld, Dekai Wu, Fred Jelinek, and Sanjeev Khudanpur. WS96 project report on dependency language modeling, 1997. URL `citeseer.nj.nec.com/article/stolcke97dependency.html`.

Andreas Stolcke and Jonathan Segal. Precise *n*-gram probabilities from stochastic context-free grammars. Technical Report TR–94–007, International Computer Science Institute, Berkeley, CA, USA, 1994.

Keh-Yih Su, Tung-Hui Chiang, and Yi-Chung Lin. A unified framework to incorporate speech and language information in spoken language processing. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 185–188, 1991.

Christoph Tillmann and Hermann Ney. Selection criteria for word trigger pairs in language modelling. In *Proc. 3rd Int. Coll. Grammatical Inference: Learning Syntax from Sentences (ICGI-96)*, pages 95–106, Montpellier, France, September 1996. Springer, Berlin.

Joerg P. Ueberla. Clustered language models with context-equivalent states. In *Proc. International Conference on Spoken Language Processing*, volume 4, pages 2060–2063, Philadelphia, PA, USA, 1996a.

Joerg P. Ueberla. An extended clustering algorithm for statistical language models. *IEEE Transactions on Speech and Audio Processing*, 4(4):313–316, July 1996b.

H. Uszkoreit. Categorial unification grammars. In *Proc. Conference on Computational Linguistics -86*, pages 187–194, 1986.

Filip Van Aelten and Marc Hogenhout. Inside-outside reestimation of Chelba-Jelinek models. Technical Report L&H–SR–00–027, Lernout & Hauspie, Wemmel, Belgium, 2000.

Dong Hoon Van Uytsel and Dirk Van Compernolle. Entropy-based context selection in variable-length n-gram language models. In *Proc. IEEE Benelux Signal Processing Symposium*, pages 227–230, Leuven, Belgium, March 1998.

Dong Hoon Van Uytsel, Dirk Van Compernolle, and Patrick Wambacq. Maximum-likelihood training of the PLCG-based language model. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop 2001*, Madonna di Campiglio, Italy, December 2001. 4 pages. ISBN 0-7803-7343-X.

A.J. Viterbi. Error bounds for convolutional codes and an asymmetrically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–267, 1967.

Wayne Ward and Sheryl Young. Flexible use of semantic constraints in speech recognition. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 49–50, 1993.

Wayne H. Ward, Alexander G. Hauptmann, Richard M. Stern, and Thomas Chanak. Parsing spoken phrases despite missing words. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 275–278, 1988.

Mats Wirén. A comparison of rule-invocation strategies in context-free chart parsing. In *Proc. 3rd Conference of the European Chapter of the Association for Computational Linguistics (EACL'87)*, pages 226–233, 1987.

J. H. Wright. LR parsing of probabilistic grammars with input uncertainty for speech recognition. *Computer Speech and Language*, 4:297–323, 1990.

J. H. Wright. Adaptation of grammar-based language models for continuous speech recognition. In *Proc. European Conference on Speech Communication and Technology*, pages 203–206, 1991.

J. H. Wright, G. J. F. Jones, and E. N. Wrigley. Hybrid grammar-bigram speech recognition system with first-order dependence model. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 169–172, 1992.

S. J. Young, N. H. Russell, and J. H. S. Thornton. Speech recognition in VODIS II. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 441–444, 1988.

APPENDIX A

Proofs of Lemma's 1 and 2

First note the following observation.

Lemma 4 *Let $\mathscr{G}$ be a PLCG network and $q_1, q_2 \in \mathscr{G}$. Then $q_1 \prec q_2$ implies $pos(q_1) \leq pos(q_2)$. If q_1 and q_2 are connected by a path, then $pos(q_1) < pos(q_2)$ implies $q_1 \prec q_2$.*

Proof. A SHIFT increments pos(·) with 1, while PROJECT and ATTACH() moves leave pos(·) unaltered. Thus if a path $t \in \langle q_1, q_2 \rangle$ contains n SHIFT moves, then $\text{pos}(q_2) = \text{pos}(q_1) + n \geq \text{pos}(q_1)$. Conversely, if q_1 and q_2 are connected by a path and $\text{pos}(q_1) < \text{pos}(q_2)$, then $q_1 \prec q_2$ or $q_2 \prec q_1$. Assume $q_2 \prec q_1$. Then $\text{pos}(q_2) \leq \text{pos}(q_1)$ which is in contradiction with the given. □

Lemma 1 *Let t be a partial path in a PLCG network $\mathscr{G}$ and $(q, q') = \text{ATTACH}(q'')$ be a move in t. If $\text{ATTACH}(s)$ is a move in a path $u \in \langle q'', q \rangle$ then $q'' \prec s$.*

Proof. The first move (q'', q^o) of u is a SHIFT move, since q'' is incomplete. So $\text{start}(q^o) = \text{pos}(q'')$. Now suppose (q_i, q_j) is a move on u. If (q_i, q_j) is a SHIFT move, then $\text{start}(q_i) < \text{start}(q_j)$. If (q_i, q_j) is a PROJECT move, then $\text{start}(q_i) = \text{start}(q_j)$. If (q_i, q_j) is an ATTACH() move, first consider the case that it is the first ATTACH() move in u, so that $\text{start}(r) \geq \text{pos}(q'')$ for all $r \in u$ for which $q'' \prec r \preceq q_i$. Assume $(q_i, q_j) = \text{ATTACH}(s), s \neq q''$. Then $\text{pos}(q'') \leq \text{pos}(s) < \text{pos}(q_i)$, since $\text{start}(q_i) = \text{pos}(s)$, $\text{start}(q_i) < \text{pos}(q_i)$ and $\text{start}(q_i) \geq \text{pos}(q'')$. By Lemma 4, $q'' \prec s \prec q_i$, which was to be proven. In the other case that (q_i, q_j) is not the first ATTACH() move, repeat the above reasoning on $\langle q_k, q_i \rangle$ instead of $\langle q'', q \rangle$. □

Lemma 2 *If $q = [_i \underline{X} *_j \beta | \vec{h}]_Z$, then on every w_0^i-constrained path in $\langle q_I, q \rangle$ there is a node $q^o = [_i w_{i\,i+1} | \vec{h}]_W$.*

Proof. Any w_0^i-constrained path must contain a node $r = [_i\, w_{i\,i+1}\, |\vec{g}]_{\mathbb{W}}$. If the given q is on the same path, then r and q can be connected by PROJECT moves only, since SHIFT and ATTACH moves do not preserve the start(·) property. PROJECT moves preserve the local tree context. Hence $\vec{h} = \vec{g}$ and thus $r = q^o$. □

APPENDIX B

Derivation of recursion formulas (3.17), (3.18) and (3.32) for forward, inner and outer probabilities

Let a constrained PLCG network $\mathscr{G}_{w_0^n}$ be given. Suppose one wants to compute $\mu(q')$ and $\nu(q')$ when $\mu(q)$ and $\nu(q)$ is already known for each $q : q \prec q'$.
A move $(q,q') = \text{ATTACH}(q'')$ does not contribute a term $\mu(q)P(q'|q)$ to $\mu(q')$, since not all paths $\langle q_I, q\rangle$ extend to paths in $\langle q_I, q'\rangle$. Using *inner probabilities* it can be ensured that only paths $\{t \in \langle q_I, q\rangle : q'' \in t\}$ contribute to $\mu(q')$.
Let q' be the node for which one wants to compute $\mu(q')$. Then

$$\langle q_I, q'\rangle = \left\{ \bigcup_{q,q''} \langle q_I, q''\rangle (q'', q^o) \langle q^o, q\rangle (q, q') \right\} \quad \cup \quad \left\{ \bigcup_{q} \langle q_I, q\rangle (q, q') \right\}, \qquad \text{(B.1)}$$

where the first union is over q, q'' for which $(q,q') = \text{ATTACH}(q'')$ for some q'' and the second union is over q for which (q,q') is not an ATTACH move. The above expansion is justified by Lemma 1 which states that any path in $\langle q_I, q''\rangle$ concatenates with $(q'', q^o)\langle q^o, q\rangle$ to a valid path in $\langle q_I, q\rangle$, given that $(q,q') = \text{ATTACH}(q'')$: $\langle q_I, q''\rangle (q'', q^o)\langle q^o, q\rangle$ exactly covers the paths of $\langle q_I, q\rangle$ that visit q''. In terms of probabilities:

$$\mu(q') = \sum_{q,q''} \left\{ \mu(q'')P(q^o|q'')P(q'|q) \sum_{t \in \langle q^o, q\rangle} P(t) \right\} + \sum_{q} \mu(q)P(q'|q), \qquad \text{(B.2)}$$

where the first sum is over all q for which $(q,q') \in \mathscr{G}_{w_0^n}$ and $(q,q') = \text{ATTACH}(q'')$, the second sum is over all q for which $(q,q') \in \mathscr{G}_{w_0^n}$ and (q,q') is a PROJECT move.

The sum within the first summand is an inner probability. Hence (3.17) follows:

$$\mu(q') = \sum_{q,q''} \mu(q'')P(q^o|q'')\nu(q)P(q'|q) + \sum_{q} \mu(q)P(q'|q). \tag{B.3}$$

The derivation of a recursive formula for the inner probability is analogous with the one above, where the node q^o substitutes q_I. One obtains (3.18):

$$\nu(q') = \sum_{q,q''} \nu(q'')P(q^o|q'')\nu(q)P(q'|q) + \sum_{q} \nu(q)P(q'|q) \tag{B.4}$$

for a node q'.

A similar exercise can be done for the computation of the outer probability $\xi(q)$, given all $\xi(q')$ where (q,q') is a move in the constrained network. Let us first consider the case where q is resolved, i.e. (q,q') is either a PROJECT or an ATTACH.

If $(q,q') = \text{ATTACH}(q'')$, then paths accounted for in $\xi(q')$ are legal combinations in $\langle q_I, q'^o\rangle \times \langle q_j, q_F\rangle$. In order to obtain paths accounted for in $\xi(q)$, these path segments have to be combined with combinations in $\langle q'^o, q''\rangle (q'', q^o) \times \{(q,q')\}$. Since all attach constraints are already accounted for by $\xi(q')$, the contribution of q' to $\xi(q)$ is obtained as the product

$$\xi(q')\nu(q'')P(q^o|q'')P(q'|q).$$

The situation is more simple if (q,q') is a PROJECT which implies $q^o = q'^o$. Path combinations accounted for in $\xi(q')$ are extended by the arc (q,q') to obtain path combinations accounted for in $\xi(q)$. Therefore, a term $\xi(q')P(q'|q)$ in the computation of $\xi(q)$ is obtained.

Summarizing the previous two paragraphs, the outer probability of a *resolved* q is obtained as

$$\xi(q) = \sum_{q' \in A(q)} \xi(q')P(q'|q) + \sum_{(q',q'') \in B(q)} \xi(q')\nu(q'')P(q^o|q'')P(q'|q, \tag{B.5}$$

where

$$A(q) = \{q' | (q,q') \text{ is a PROJECT move}\}, \text{ and}$$
$$B(q) = \{(q',q'') | (q,q') = \text{ATTACH}(q'')\}.$$

This is the first part of (3.32).

If q is unresolved, then $\xi(q')$ is of no use because it contains probabilities of paths that visit q' but not q. There is however another way: consider all moves $(q_1,q_2) = \text{ATTACH}(q)$ in the constrained network. Paths covered by $\xi(q_2)$ have to be extended with segments in respectively $\{(q,q' = q_1^o)\}, \langle q_1^o, q_1\rangle$ and $\{(q_1,q_2)\}$. In other words, (q_1,q_2) contributes a term $\xi(q_2)P(q'|q)\nu(q_1)P(q_2|q_1)$ to $\xi(q)$. Summing over all (q_1,q_2):

$$\xi(q) = \sum_{(q_1,q_2) \in C(q)} \xi(q_2)P(q'|q)\nu(q_1)P(q_2|q_1), \tag{B.6}$$

where $C(q) = \{(q_1,q_2) | (q_1,q_2) = \text{ATTACH}(q)\}$. This is the second part of (3.32).

APPENDIX C

Derivation of the expected move frequency formula (3.31)

The expected frequency of a move (q_i, q_j) is computed as

$$P((q_i,q_j)|W) = \frac{P((q_i,q_j),W)}{P(W)} = \frac{1}{P(W)} \sum_{t \in T(q_i,q_j)} P(t),$$

where $T(q_i,q_j)$ is the set of all constrained paths in $\langle q_I, q_F \rangle$ that contain a move (q_i,q_j). For the factoring of the sum over $T(q_i,q_j)$, one has to consider three cases, depending on (q_i,q_j).

Case 1: (q_i,q_j) is a SHIFT move. There is only one arc starting from q_i (namely (q_i,q_j)), so all paths through q_i also visit q_j. The sum of the probabilities of these paths is $\nu(q_i)\xi(q_i)$: $\nu(q_i)$ is the sum of the probabilities of subpaths in $\langle q_i^o, q_i \rangle$, while $\xi(q_i)$ takes the complements of these subpaths: from q_I to q_i^o and from q_i to q_F, observing all attach constraints.

Case 2: (q_i,q_j) is a PROJECT move. In this case $q_i^o = q_j^o$. Paths in $T(q_i,q_j)$ are characterized as concatenations of subpaths in $\langle q_I, q_i^o \rangle$, $\langle q_i^o, q_i \rangle$, (q_i,q_j) and $\langle q_j, q_F \rangle$, as is illustrated in Fig. C.1. The probability sum over $\langle q_i, q_i^o \rangle$ is given by $\nu(q_i)$. The

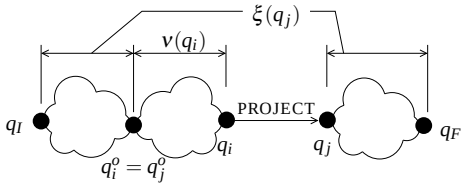

Figure C.1. Summing probabilities of all paths containing a PROJECT move (q_i,q_j).

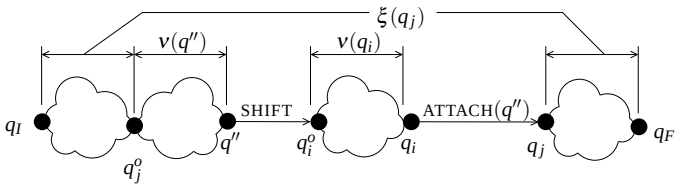

Figure C.2. Summing probabilities of all paths containing an ATTACH move (q_i, q_j).

probability sum of valid combinations of elements in $\left\langle q_I, q_j^o \right\rangle$ and $\langle q_j, q_F \rangle$ is given by $\xi(q_j)$. Therefore, the probability sum over $T(q_i, q_j)$ is $\nu(q_i)\xi(q_j)P(q_j|q_i)$.
Case 3: $(q_i, q_j) = \text{ATTACH}(q'')$. In this case $q_j^o \prec q_i^o$. A similar reasoning as in case 2, this time on Fig. C.2, leads to the conclusion that

$$P((q_i, q_j), W) = \xi(q_j)\nu(q'')P(q_i^o|q'')\nu(q_i)P(q_j|q_i).$$

APPENDIX D

The Penn Treebank Tag Set

The following tables were reproduced from [Marcus et al., 1993].

D.1. Part-of-speech tags

tag	meaning	tag	meaning
CC	coordinating conjunction	*TO*	*to*
CD	cardinal number	*UH*	interjection
DT	determiner	*VB*	verb, base form
EX	existential	*VBD*	verb, past tense
FW	foreign word	*VBG*	verb, gerund or present participle
IN	preposition or	*VBN*	verb, past participle
	subordinating conjunction	*VBP*	verb, non-3rd person singular present
JJ	adjective	*VBZ*	verb, 3rd person singular present
JJR	adjective, comparative	*WDT*	*wh*-determiner
JJS	adjective, superlative	*WP*	*wh*-pronoun
LS	list item marker	*WP$*	possessive *wh*-pronoun
MD	modal	*WRB*	*wh*-adverb
NN	noun, singular or mass	*#*	pound sign
NNS	noun, plural	*$*	dollar sign
NNP	proper noun, singular	*.*	sentence-final punctuation
NNPS	proper noun, plural	*,*	comma
PDT	prdeterminer	*:*	colon, semi-colon
POS	possessive ending	*(*	left bracket character
PRP	personal pronoun	*)*	right bracket character
RB	adverb	*"*	straight double quote
RBR	adverb, comparative	*‘*	left open single quote
RPS	adverb, superlative	*‘‘*	left open double quote
RP	particle	*’*	right close single quote
SYM	symbol	*’’*	right close double quote

D.2. Syntactic constituent labels

tag	meaning
ADJP	adjective phrase
ADVP	adverb phrase
NP	noun phrase
PP	prepositional phrase
S	simple declarative clause
SBAR	clause introduced by subordinating conjunction or *0* (see below)
SBARQ	direct question introduced by *wh*-word or *wh*-phrase
SINV	declarative sentence with subject-aux inversion
SQ	subconstituent of *SBARQ* excluding *wh*-word or *wh*-phrase
VP	verb phrase
WHADVP	*wh*-adverb phrase
WHNP	*wh*-noun phrase
WHPP	*wh*-prepositional phrase
X	constituent of unknown or uncertain category
Null elements:	
*	'Understood' subject of infinitive or imperative
0	zero variant of *that* in subordinate clauses
T	trace — marks position where moved *wh*-constituent is interpreted
NIL	marks position where preposition is interpreted in pied piping contexts

D.3. Function tags

The following table summarizes section 2.2 of the Penn Treebank version 3 parsing guide.

tag	meaning	tag	meaning
Form/function discrepancies			
-ADV	adverbial	*-NOM*	nominal
Grammatical role			
-DTV	dative	*-LGS*	logical subject
-PRD	predicate	*-PUT*	locative complement of *put*
-SBJ	surface subject	*-TPC*	topicalized
-VOC	vocative		
Adverbials			
-BNF	benefactive	*-DIR*	direction
-EXT	extent	*-LOC*	locative
-MAN	manner	*-PRP*	purpose or reason
-TMP	temporal		
Miscellaneous			
-CLR	closely related	*-CLF*	cleft
-HLN	headline	*-TTL*	titel

APPENDIX E

Extended Dutch summary: Probabilistische taalmodellering met linkerhoekontleding

Inleiding

Sinds de beginjaren van spraakherkenning is statistische taalmodellering een bron van frustratie geweest voor vele onderzoekers. Toen Jelinek and Mercer [1980] hun *n*-gramtaalmodel voorstelden, een naïef Markovmodel van orde $n-1$, vermoedden ze niet dat hun eerste ruwe poging de volgende twintig jaar toonaangevend zou blijven. Het blijkt tot op heden bijzonder moeilijk om de combinatie van nauwkeurigheid en eenvoud van het *n*-gramtaalmodel te verbeteren.

Onderzoek op taalmodellen heeft vooral vooruitgang geboekt op het gebied van statistische *afvlakkings*- en modelcombinatietechnieken, die in de meeste gevallen breder toepasbaar zijn dan enkel voor taal. Daarnaast werden er ook taalmodellen ontwikkeld die, niet ter vervanging, maar als *aanvulling* op het *n*-grammodel, de spraakherkenningsnauwkeurigheid significant verbeteren. In een recent experiment bracht Goodman [2001a] de belangrijkste van deze technieken samen en behaalde een verlaging van 40% van de testsetperplexiteit en een verlaging van 8.9% van de woordfoutfrequentie in spraakherkenning.

Helaas kwam Goodman er niet toe om statistische *grammatica-gebaseerde* taalmodellen in zijn experiment te betrekken. De belangstelling voor deze modelklasse nam weer toe sinds midden de jaren '90, vooral onder invloed van de bemoedigende resultaten bekomen door Chelba and Jelinek [1999] met hun zogeheten *gestructureerde* taalmodel. Grammatica-gebaseerde taalmodellen zijn aantrekkelijk omdat ze taalpatronen trachten te veralgemenen op een linguïstisch-intuïtieve manier, namelijk met syntactische grammatica-regels. Grammatica-regels (of een collectie vooraf ontleden

zinnen) vormen *a priori* kennis waarvan traditionele taalmodellen geen gebruik van maken.
In deze thesis stel ik een nieuw taalmodel voor dat gebaseerd is op *probabilistische linkerhoekontleding*. Het is vergelijkbaar met de grammatica-gebaseerde taalmodellen van Chelba [Chelba and Jelinek, 2000] en Roark [Roark, 2001], maar het gebruikt een ander ontledingsalgoritme. Een verlaging werd vastgesteld van de kruisentropie met 0,32 bits (20% in perplexiteit) en van de woordfoutfrequentie met 12% relatief in combinatie met het referentiemodel, een woordgebaseerd trigram. De toevoeging van het op linkerhoekontleding gebaseerde model aan een woordgebaseerd trigram en een klassegebaseerd 4-gram verlaagt de woordfoutfrequentie met 6.2% relatief. Net zoals andere grammatica-gebaseerde taalmodellen vereist het op linkerhoekontleding gebaseerde model veel computerkracht in vergelijking met traditionele taalmodelleringstechnieken, maar zijn nauwkeurigheid is opmerkelijk hoger.

Beperkingen

Vanuit wetenschappelijk oogpunt zijn de testresultaten van het op linkerhoekontleding gebaseerde taalmodel substantieel. Nochtans zijn geavanceerde taalmodelleringstechnieken beperkt in hun praktische toepassingsmogelijkheden wegens hun steile kosten/baten-verhouding. *Waarom* is taalmodellering moeilijk? Er zijn enkele randvoorwaarden die taalmodellering moeilijk maken:

1. Taalmodellen leren enkel uit tekstcorpora, terwijl de statistieken van taal door veel meer niet-tekstgebonden parameters worden beïnvloed. Het is onmogelijk om met al deze parameters rekening te houden.

2. Taalmodellen kunnen wel voor een eng taalbereik getraind worden, namelijk op een specifiek, homogeen corpus. De mogelijkheden van deze aanpak zijn beperkt, omdat de niet-tekstgebonden parameters nooit constant zijn. Deze vaststelling duidt op de inherente heterogeniteit van taal.

3. Taalmodellen zien woorden enkel als abstracte symbolen en zijn blind voor woord-interne structuur.

4. Taalmodellen krijgen enkel positieve voorbeelden. Mensen leren taal op een interactieve manier.

5. Taalmodellen opereren in een ‘vijandige’ omgeving; namelijk door de imperfectie van de andere kennismodules (b.v. het akoestisch model) in de toepassing moet het taalmodel een onderscheid maken tussen heel wat meer hypothesen dan een mens.

Het op linkerhoekontleding gebaseerde taalmodel is aan al deze beperkingen onderhevig, maar probeert de eerste twee te verleggen: de variabiliteit in taal kan beter worden

gemodelleerd als de invarianten beter gekend zijn — waarbij men veronderstelt dat syntax relatief invariant is, en het model integreert a priori kennis van grammatica met observaties uit een tekstcorpus.
Het voorgestelde taalmodel is ook onderhevig aan de volgende beperkingen:

- Zijn modelhorizon reikt niet over de zinsgrenzen.
- Syntactische structuur wordt voorgesteld met bomen zonder kruisende takken noch niet-lokale verwijzingen. Knooppunten worden benoemd met eenvoudige syntactische categorienamen (b.v. *werkwoordszin*), zonder opdeling in kenmerken (b.v. *3de persoon*).
- Er wordt enkel gebruik gemaakt van syntactische structuur; semantische coherentie wordt niet expliciet gemodelleerd.
- De initialisatie van het model gebruikt voorontlede zinnen; linguïstische competentie, die niet blijkt uit deze voorontlede zinnen, kan niet worden aangeleerd.
- Woorden worden niet morfologisch geanalyseerd, wat ontleding van zinnen met ongekende woorden moeilijker maakt.

Op het einde van de thesis volgt een bespreking van enkele mogelijke uitbreidingen van het model om deze tekortkomingen te verhelpen.
De volgende twee hoofdstukken bevatten inleidend materiaal. Hoofdstuk 1 geeft een inleiding tot het onderzoeksdomein van de statistische taalmodellering. Hoofdstuk 2 beschrijft het onderzoek op grammatica-gebaseerde taalmodellen. Het onderzoek beschreven in deze thesis bevindt zich in de doorsnede van deze twee domeinen. De theoretische aspecten van het voorgestelde taalmodel staan beschreven in hoofdstuk 3, terwijl hoofdstuk 4 de experimentele resultaten beschrijft. Het laatste hoofdstuk bevat besluiten en voorstellen voor verder onderzoek.

E.1. Statistische taalmodellering

Dit hoofdstuk geeft een korte inleiding tot het onderzoeksdomein van de statistische taalmodellering. Na de definitie van het statistische taalmodel, volgt een korte bespreking van zijn toepassing, en vervolgens een overzicht van standaardtechnieken voor taalmodellering. Het hoofdstuk sluit af met een samenvatting van de resultaten van een experimentele studie, die de besproken technieken vergelijkt.

E.1.1. Wat is een statistisch taalmodel?

Taal wordt voorgesteld als een datastroom van *zinnen*, waarbij elke zin een eindige string van *woorden* is. De zinsgrenzen worden gemarkeerd met speciale symbolen `<s>` (begin) en `</s>` (einde). Een *statistisch taalmodel* is een computermodel dat een

bepaalde zin met een zekere probabiliteit genereert. De bepaling 'statistisch' wordt weggelaten als deze duidelijk is uit de context. In wiskundige termen is een statistisch taalmodel een probabiliteitsmassafunctie over de ruimte van alle zinnen.
Een *conditioneel taalmodel* genereert een volgend woord met een zekere probabiliteit, gegeven de voorgaande woorden. In wiskundige termen is een conditioneel taalmodel een verzameling conditionele probabiliteitsmassafuncties, elk gedefinieerd over een vocabularium.
Elk conditioneel taalmodel leidt tot een gewoon taalmodel via de kettingregel van Bayes, maar de omgekeerde bewering geldt niet altijd.

E.1.2. Toepassing van statistische taalmodellen

Verscheidene statistische methoden in spraak- en taaltoepassingen, maar ook daarbuiten, gebruiken statistische taalmodellen: bijvoorbeeld documentclassificatie, spellingcorrectie, automatische vertaling, maar ook modellering van DNA-strings. De meeste traditionele taalmodelleringstechnieken kunnen nuttig zijn in elke situatie waarin een discrete probabiliteitsmassafunctie over een multi-dimensionele categorische ruimte nodig is.
De rol van het statistische taalmodel in *automatische spraakherkenning* (ASH) wordt nu wat meer in detail besproken, omdat ASH als testomgeving zal fungeren voor het model dat in deze thesis wordt ontwikkeld.
De probabilistische formulering van ASH streeft er naar de kans te minimaliseren om 1 of meerdere fouten in een te herkennen zin te maken. Dit kan door die zinshypothese als herkenningsresultaat weer te geven, die met de meeste kans werd uitgesproken, gegeven de akoestische invoer. Noem de akoestische invoer A en laat W een willekeurige zinshypothese betekenen, dan kan men schrijven:

$$\hat{W} = \arg\max_W P(W|A) = \arg\max_W f(A|W)P(W). \tag{E.1}$$

De *a posteriori* waarschijnlijkheid $P(W|A)$ kan niet rechtstreeks worden gemodelleerd; daarom is het nuttig deze te herschrijven als het product van een conditionele densiteit (meestal is de akoestische ruimte continu gedefinieerd) met de *a priori* waarschijnlijkheid $P(W)$. De zoekprocedure berekent $f(A|W)$ met een aaneenschakeling van akoestische foneemmodellen; mogelijke aaneenschakelingen volgen uit een uitspraaklexicon of -netwerk. De factor $P(W)$ wordt geschat door het statistische taalmodel.
Typisch zal de zoekmachine de hypothesen en hun probabiliteiten incrementeel van links naar rechts opbouwen via de kettingregel. Op die manier kan hij deelhypothesen, die met grote kans niet tot de meest waarschijnlijke volledige hypothese zullen leiden, in een vroeg stadium buiten beschouwing laten. De uitvoertijd van de spraakherkenning blijft zodoende binnen aanvaardbare grenzen. Deze zoekstrategie vereist een conditioneel taalmodel.

E.1.3. Taalmodellen schatten uit observaties

Taalmodellen worden geschat uit een traincorpus O, een observatiesequentie van doorlopende tekst. Het traincorpus kan voorgesteld worden als een sequentie van onafhankelijke samengestelde gebeurtenissen $x = (h, w)$, waarin w een woord is en h de woorden in dezelfde zin die aan w voorafgaan.

Volgens de normale werkwijze wordt een zekere geparametriseerde modelklasse aangenomen. Het is de taak van een schatter om optimale waarden te bepalen voor de vrije parameters, waaruit het optimale model volgt, geselecteerd uit de aangenomen modelklasse.

Deze sectie expliciteert wat onder 'optimaliteit' kan verstaan worden. Vervolgens zal worden aangetoond dat de *maximum-likelihood*-schatter niet volstaat voor de klasse van de woordgebaseerde trigrams: het onderzoek op taalmodellen is een zoektocht naar betere schatters en modelklassen.

Taalmodellen zijn dikwijls een samenstelling van een aantal elementaire technieken, die na deze sectie besproken worden: (a) de keuze van een clusterfunctie, die een enkelvoudige modelklasse karakteriseert; (b) discounting-technieken; (c) combinatie van taalmodellen.

Metrieken voor de kwaliteit van een taalmodel

De praktische bruikbaarheid van een taalmodel hangt af van de resulterende prestaties van het systeem, waarbinnen dit taalmodel wordt gebruikt. Bijvoorbeeld, voor ASH kan men dus het taalmodel evalueren in termen van gemiddelde woordfoutfrequentie, geheugengebruik, uitvoersnelheid, ...

Dikwijls is er echter een toepassingsonafhankelijke kwaliteitsmetriek nodig; dit is bijvoorbeeld het geval bij numerieke of analytische modeloptimalisatie.

Een bekende kwaliteitsmaat is de **verwachte kwadratische fout** op de modelparameters t.o.v. de parameters van het echte model. De 'beste' set van parameters minimaliseert deze metriek. Een schatter die deze strategie volgt, heet een MMSE-schatter (*minimal mean square error*).

De MMSE-schatter geeft even veel gewicht aan de nauwkeurigheid van elke modelparameter. Voor de schatting van een stochastische verdeling is het gebruikelijker, de **Kullback-Leibler**-afstand (KL) te minimaliseren. De KL-afstand tussen een taalmodel en de echte verdeling is te schrijven als een vaste term (de echte entropie) plus de **kruisentropie** van het taalmodel t.o.v. de echte verdeling. Meestal wordt de **perplexiteit** (PPL) gerapporteerd in de literatuur. Deze is 2^H, waarin H de empirische kruisentropie is, uitgedrukt in bits. De PPL van een taalmodel kan in een spraakherkenningssysteem geïnterpreteerd worden als de gemiddelde ambiguïteit (*branching factor*) die het akoestisch model ziet bij toepassing van dat taalmodel. Minimalisatie van de KL-afstand is equivalent met de minimalisatie van de *verwachte* testset-

perplexiteit, en met de maximalisatie van de verwachte waarschijnlijkheid van een willekeurige testset.

Maximum-likelihood-schatting en de noodzaak van afvlakking

De *Maximum-Likelihood*-schatter (ML) maximaliseert de waarschijnlijkheid van het traincorpus. De ML-schatting van de probabiliteit van een gebeurtenis komt overeen met zijn relatieve frequentie. In taalmodellen zijn de meeste gebeurtenissen zeldzaam, zodat de relatieve frequenties te sterk afwijken van de verwachte waarde van de probabiliteiten. Een extreem voorbeeld hiervan zijn gebeurtenissen die niet werden geobserveerd in het traincorpus: deze krijgen een probabiliteit 0.
Bijgevolg is de toepassing van statistische afvlakking (*statistical smoothing*) noodzakelijk. Het grootste gedeelte van de literatuur over taalmodellering gaat precies daarover. Hierna volgt een systematische bespreking van de voornaamste afvlakkingstechnieken, en daarna een empirische vergelijking van (enkele combinaties van) deze technieken.
Een taalmodel gebruikt vaak een ingewikkelde combinatie van technieken; uit de vele theoretisch mogelijke combinaties behoren slechts enkele 'recepten' tot de gangbare praktijk.
Afvlakkingstechnieken zijn op te delen in drie categorieën: cluster-, discount- en combinatietechnieken.

E.1.4. Clustertechnieken

Een traincorpus is voor te stellen als een sequentie gebeurtenissen (h,w), waarin w een woord is en h de daaraan voorafgaande woorden (meestal beperkt tot dezelfde zin). Een clusterfunctie $\Phi(h,w)$ groepeert gelijkaardige gebeurtenissen in clusters. De probabiliteit van een cluster kan betrouwbaarder worden geschat, omdat de frequentie er hoger van is; het is namelijk de som van de frequenties van alle gebeurtenissen die tot diezelfde cluster behoren. De probabiliteit van een welbepaalde gebeurtenis, gegeven zijn cluster, kan in vele gevallen sterk worden vereenvoudigd met allerlei onafhankelijkheidsveronderstellingen, waardoor ook deze probabiliteit betrouwbaarder kan worden geschat. Het product van de twee voornoemde probabiliteiten geeft de samengestelde probabiliteit $P(h,w)$, waaruit de conditionele taalmodelprobabiliteit $P(w|h)$ volgt.
Hier volgt een overzicht van enkele welbekende clusterfuncties, die elk een aparte taalmodelklasse karakteriseren (stel $\Phi(h,w) = (\tilde{h},\tilde{w})$):

- $\tilde{w} = w$ en $\tilde{h}$ bestaat uit de laatste $n-1$ woorden van h (n is typisch 3 of 4). Het is redelijk om aan te nemen dat de probabiliteit van een woord zich in vele gevallen laat voorspellen op basis van naburige woorden. Deze clusterfunctie geeft aanleiding tot het **woordgebaseerde *n*-gram-taalmodel**. Dit simplistische model wordt getraind voor een bepaald specifiek taalgebruik, en vergt veel

traindata. In praktische toepassingen, zoals in medische dicteersystemen, is het taalgebruik inderdaad erg specifiek, en zijn er veel traindata beschikbaar. Het blijkt erg moeilijk een taalmodel te formuleren dat het woordgebaseerde *n*-gram in zulke situaties overtreft.

- $\tilde{h}$ bestaat uit de *woordcategorieën* van de laatste $n-1$ woorden van h, en $\tilde{w}$ is de woordcategorie van w. De woordcategorieën kunnen overeenkomen met linguïstisch gedefinieerde concepten, maar kunnen ook abstracte labels zijn van woordclusters die met een automatische, datagedreven procedure zijn bekomen. Deze clusterfunctie geeft aanleiding tot het **categoriegebaseerde *n*-gram-taalmodel**. Dit model vergt minder traindata dan een woordgebaseerd taalmodel en verhoogt de robuustheid tegen toepassingsvreemd taalgebruik.

- $\tilde{h}$ bestaat uit de $n-1$ woorden van h die onmiddellijk voorafgaan aan het s-ste laatste woord van h. Het geval $s=0$ leidt tot het gewone woordgebaseerde *n*-gram-taalmodel. De keuze $s>0$ resulteert in een **skip-*n*-gram-taalmodel**. Een skip-*n*-gram benadert in combinatie met een gewoon *n*-gram een $(s+n)$-gram, waar de directe schatting van het $(s+n)$-gram onmogelijk zou zijn door te lage observatiefrequenties.

- $\tilde{w}=w$, en $\tilde{h}$ is de frequentie van w in h; in dit geval overschrijdt h de zinsgrenzen en bevat typisch de laatste 1000 à 3000 woorden. Het blijkt dat het gebruik van een woord de kans verhoogt dat het binnen korte tijd nogmaals gebruikt wordt. Het resulterende model heet **cache-gebaseerd unigram model**. Een veralgemening naar cache-gebaseerde bi- en trigrams ligt voor de hand. **Trigger-gebaseerde modellen** vormen een veralgemening in de andere richting: alle woorden uit h die een **trigger-paar** vormen met w, en niet alleen w zelf, verhogen de probabiliteit van w.

- $\tilde{w}=w$, en $\tilde{h}$ is h plus een — eventueel gedeeltelijke — grammaticale analyse van h. Uiteraard is de grammaticale analyse meestal ambigu, d.w.z. de clusterfunctie is stochastisch. In deze thesis heet deze klasse **statistische grammatica-gebaseerde taalmodellen**. Het PLCG-gebaseerde taalmodel, het hoofdonderwerp van deze tekst, behoort tot deze klasse. Het acroniem PLCG staat voor *probabilistic left corner grammar*, en zal verder worden verklaard.

E.1.5. Discounttechnieken

Een clusterfunctie $\Phi(h,w)=(\tilde{h},\tilde{w})$ alleen, met gebruik van de ML-schatter

$$P(\tilde{w}|\tilde{h})=\frac{\text{frequentie}(\tilde{h},\tilde{w})}{\text{frequentie}(\tilde{h})},$$

volstaat meestal niet voor een optimale afvlakking.

Discounttechnieken groeperen de gebeurtenissen (h,w) die even vaak voorkomen in het traincorpus, en schatten voor de gebeurtenissen in eenzelfde groep een gemeenschappelijke probabiliteit. Deze probabiliteit ligt meestal lager dan de ML-schatter, vandaar de term 'discount'.
Nu volgt een kort overzicht van courante discounttechnieken:

1. **Laplace's opvolgingswet**: tel 1 bij elke geobserveerde frequentie, alvorens de ML-schatter toe te passen. Bijgevolg krijgen alle ongeziene gebeurtenissen toch een probabiliteit $1/|O|$ ($|O|$ is het totaal aantal observaties, d.i. de grootte van het traincorpus). Deze aanpak is optimaal in de zin van kleinste kwadraten, als de a priori verdeling uniform is. Deze techniek is bruikbaar voor de afvlakking van unigrams.

2. **Jelinek-Mercer(JM)-discounting** neemt een frequentie K aan waarboven de ML-schatter betrouwbaar genoeg is (b.v. $K = 1$). De frequenties van gebeurtenissen die vaker dan K keer voorkomen worden lineair gescaleerd met een parameter $\alpha < 1$, alvorens de ML-schatter toe te passen. Hierdoor komt er probabiliteitsmassa vrij die kan herverdeeld worden over de probabiliteiten van de gebeurtenissen die K of minder keer voorkwamen. Deze probabiliteiten kunnen volgen uit een eenvoudiger taalmodel, of experimenteel bepaald worden uit een apart gehouden gedeelte van het traincorpus. JM-discounting blijkt minder goede resultaten te leveren dan GT-discounting, behalve op kleine traincorpora. Ook is JM-discounting nuttig bij niet-gehele observatiefrequenties (b.v. verwachtingen van frequenties).

3. **Good-Turing(GT)-discounting** is wellicht de theoretisch best onderbouwde techniek. Stel dat een gebeurtenis k keer voorkomt, en dat n_k het aantal verschillende gebeurtenissen is die k keer voorkomen, dan wordt de gewijzigde frequentie $k_{\mathrm{GT}} = (k+1)n_{k+1}/n_k$. De ML-schatter wordt vervolgens toegepast op de gewijzigde frequenties. **Katz discounting** is een variant, die enkel de frequenties lager dan een bepaalde parameter K wijzigt (K is typisch $7\ldots10$).

4. **Nádas-afvlakking** veronderstelt dat de observatiefrequentie k van een gebeurtenis binomiaal verdeeld is, en kan bijgevolg een a posteriori probabiliteit berekenen $\hat{p}_k = E[p_k|k]$ uit de binomiaalverdeling van k en een aangenomen a priori beta-verdeling van p_k. Deze methode geeft resultaten die vergelijkbaar zijn met die van GT-discounting, maar is ingewikkelder. Mogelijk hierdoor wordt hij zelden gebruikt.

5. **Absolute discounting**: deze methode trekt een vaste hoeveelheid $d < 1$ af van de frequenties van geziene gebeurtenissen alvorens de ML-schatter toe te passen. De vrijgekomen probabiliteit herverdeelt de vrijgekomen probabiliteitsmassa over alle, geziene en ongeziene, gebeurtenissen in verhouding met een achtergrondverdeling. Absolute discounting is een erg simplistische, maar in

de praktijk betrouwbare, robuuste en vaak gebruikte methode. Dit geldt in het bijzonder in situaties — en zo zijn er vele — waar GT-discounting niet betrouwbaar is.

E.1.6. Modelcombinatie

Voor de afvlakking van een model $p(w|\tilde{h})$ is ook de combinatie met een ander model $q(w|\tilde{\tilde{h}})$ nuttig. q gebruikt een meer doorgedreven vereenvoudiging van h, of is getraind op diversere data. Bijgevolg is q minder waarheidsgetrouw dan p, maar baseert het zijn probabiliteitsschattingen op hogere observatiefrequenties, wat de betrouwbaarheid verhoogt. Men noemt q een **back-off**-model (of -verdeling). Een typisch voorbeeld is een $(n-1)$-gram taalmodel als back-off-model voor de afvlakking van een n-gram taalmodel. Men gebruikt ook de term **achtergrond**-model als de afvlakking de bedoeling heeft de robuustheid tegen toepassingsvreemd taalgebruik te verhogen. Hier volgt een kort overzicht van modelcombinatietechnieken.

1. **Katz-back-off** maakt een onderscheid tussen gebeurtenissen die vaker voorkomen in het traincorpus dan een ingestelde *afsnijfrequentie* K (typisch is $K = 0\ldots3$), en de andere. Deze methode gaat ervan uit dat $p(w|\tilde{h})$ betrouwbaar genoeg is als $\text{frequentie}(\tilde{h},w) > K$ en laat deze ongemoeid. De andere probabiliteiten worden uit de back-off-verdeling gehaald en gescaleerd, zodanig dat de som van alle probabiliteiten 1 blijft.

2. **Interpolatie-back-off** interpoleert het af te vlakken model met het back-off-model. Als de interpolatiefactor constant is, spreekt men van **context-onafhankelijke lineaire interpolatie**; als deze enkel van h afhangt, spreekt men van **context-afhankelijke lineaire interpolatie**. In **niet-lineaire interpolatie** hangt de interpolatiecoëfficient van het af te vlakken model af van zowel w als h; de interpolatiecoëfficient van het back-off-model hangt enkel af van h en zorgt ervoor dat het geïnterpoleerde model genormaliseerd blijft.

3. **Kneser-Ney(KN)-back-off**: deze methode stelt een gewijzigde back-off-verdeling voor. De schatting van het gewone back-off-model $q(w|\tilde{\tilde{h}})$ is gebaseerd op relatieve frequenties $\text{frequentie}(\tilde{\tilde{h}},w)/\text{frequentie}(\tilde{\tilde{h}})$. De KN-schatting gebruikt voor de schatting van $q(w|\tilde{\tilde{h}})$ echter *het aantal verschillende* $\tilde{h}$ die vóór w werden geobserveerd en $\tilde{\tilde{h}}$ als back-off-context hebben. De KN-back-off-verdeling kan gebruikt worden met zowel Katz-back-off als interpolatie-back-off, en geeft meestal beduidend betere resultaten dan de gewone back-off-verdeling.

4. **Maximum-entropie-modellen** combineren niet alleen informatie uit andere taalmodellen; de enige voorwaarde is dat de informatie kan worden voorgesteld als een *kenmerkfunctie* van (h,w). De vorm van het maximum-entropiemodel is de exponentiële functie van een gewogen som van de kenmerkfuncties. De

gewichten bepalen het relatieve belang van de kenmerkfuncties. De algemene formulering maakt het maximum-entropiemodel aantrekkelijk, maar zowel de schatting (het bepalen van de gewichten) als het gebruik van het model (het vinden van de niet-verwaarloosbare kenmerkfuncties voor een bepaalde w en h) vergen veel computertijd.

5. **Log-lineaire interpolatie** is verwant met maximum-entropiemodellering, maar gebruikt enkel informatie uit andere taalmodellen. Het model met maximale entropie, dat zich tegelijk op gegeven KL-afstanden van de respectieve te interpoleren taalmodellen bevindt, blijkt een lineaire interpolatie te zijn *in het logprob-domein* (het logaritme van de combinatie is een lineaire combinatie van de logaritmes van de respectieve taalmodelprobabiliteiten). Hernormalisatie is evenwel nodig. Log-lineaire interpolatie blijkt uit experimenten beter te presteren dan context-onafhankelijke lineaire interpolatie, maar vindt in de praktijk weinig toepassing om mij onbekende redenen.

E.1.7. Empirische vergelijking van taalmodelleringstechnieken

Alle tot nu toe besproken technieken laten zich op een oneindig aantal wijzen combineren; slechts enkele daarvan zijn gangbare praktijk geworden.
De experimentele studies van Chen en Goodman [Chen and Goodman, 1999, Goodman, 2001a] zijn waarschijnlijk de meest uitgebreide en betrouwbaarste in de literatuur over taalmodellering. Zij vergeleken de onderlinge prestatieverhoudingen tussen verschillende afvlakkingstechnieken en combinaties daarvan voor een variërende traincorpusgrootte. Prestatie werd gemeten in empirische kruisentropie (het logaritme van testsetperplexiteit) en woordfoutfrequentie bij gebruik van het taalmodel in een spraakherkenningstest.
Uit een vergelijking van verschillende afvlakkingstechnieken, toegepast op een woordgebaseerd trigram, bleek Kneser-Ney lineaire-interpolatie-back-off de beste keuze voor traincorpora van 1k tot 200M woorden.
Het woordgebaseerde trigram, afgevlakt met Kneser-Ney lineaire-interpolatie-back-off, functioneert vervolgens als basismodel in de vergelijking met volgende modellen (alle afgevlakt met Kneser-Ney back-off-verdelingen, en gecombineerd met het basismodel zelf): een woordgebaseerd 5-gram, een trigram-cache, een skip-model en een categorie-gebaseerd n-gram. Met een traincorpusgrootte van 284M woorden verlaagt elk van deze modellen de empirische kruisentropie met 0.1 tot 0.3 bit/woord t.o.v. het basismodel. Door bovendien al deze modellen te combineren, kon een verlaging van 0.5 bit/woord verkregen worden, en een relatieve verlaging van de woordfoutfrequentie van 7.3%.
Chen en Goodman beperkten zich weliswaar tot het genre van financiële nieuwsberichten; dit vormt voor de vergelijkbaarheid met mijn experimenten geen bezwaar, aangezien mijn modellen op een gelijkaardig genre werden getraind en getest. Men

kan er echter niet van uit gaan dat de relatieve prestaties van afvlakkingstechnieken zich doortrekken naar andere tekstgenres; hiernaar werd nog geen diepgaand empirisch onderzoek naar verricht.

E.1.8. Besluit

Dit hoofdstuk gaf een inleiding op het onderzoeksdomein van de statistische taalmodellering. Een taalmodel geeft de waarschijnlijkheid van een woordstring terug, of de waarschijnlijkheid van een volgend woord gegeven de woorden die er aan voorafgaan. De schatting van een taalmodel uit een traincorpus (een grote tekstcollectie) is problematisch omdat de observatieruimte erg schaars bevolkt is, waardoor een eenvoudige schatting op basis van relatieve frequenties niet volstaat. Het onderzoek naar taalmodellen heeft een aantal afvlakkingstechnieken ontwikkeld. Deze technieken kunnen opgedeeld worden in drie rubrieken: clustering, discounting en modelcombinatie.
Deze technieken kunnen op enorm veel manieren worden gecombineerd. Daardoor kan de optimaliteit van een zekere combinatie van technieken niet voor 100% worden gegarandeerd op basis van empirisch onderzoek. Wel zijn er uit ervaring 'gangbare praktijken' ontstaan; de belangrijkste hiervan werden uitgebreid getest door Chen en Goodman, waarvan tenslotte de voornaamste resultaten werden vermeld.

E.2. Grammatica-gebaseerde taalmodellen

Dit hoofdstuk motiveert, en geeft een overzicht van grammatica-gebaseerde (statistische zowel als niet-statistische) taalmodellen. Het onderwerp van deze thesis wordt gesitueerd in dit onderzoeksveld.

E.2.1. Motivatie

De traditionele taalmodellen, die in het vorige hoofdstuk aan bod kwamen, definiëren de afhankelijkheden niet volgens de grammaticale structuur van de invoerzin. Dit simplisme is mee verantwoordelijk voor de verre achterstand van taalmodellen op hun menselijke equivalenten. Om toch tot taalmodel van aanvaardbare kwaliteit te komen, is een heel groot toepassingsspecifiek traincorpus nodig, en dit is niet altijd commercieel haalbaar. Taalmodellen zijn tenslotte erg toepassingsspecifiek: ze zijn moeilijk herbruikbaar voor een andere toepassing, dan waarvoor ze ontwikkeld werden.
Grammatica-gebaseerde taalmodellen baseren (al dan niet probabilistische) regels op grammaticale structuur. De gunstige resultaten bekomen met recente grammatica-gebaseerde taalmodellen, met inbegrip van het op linkerhoekontleding gebaseerde taalmodel, tonen aan dat grammaticale structuur wel degelijk een nuttige manier biedt om taalmodelprobabiliteiten te schatten.
Men hoopt eveneens dat grammaticaal gestructureerde afhankelijkheden ook generieker zijn; d.w.z. dat ze, eenmaal aangeleerd voor de ene toepassing, geheel of ge-

deeltelijk overgedragen kunnen worden naar andere toepassingen. Dit hypothetische voordeel werd nog niet grondig onderzocht, en valt ook buiten het bestek van deze thesis.

E.2.2. Grammatica-gebaseerde taalmodellen: een overzicht

Er is een onderscheid tussen taalmodellen gebaseerd op eindige-toestandsgrammatica's (*finite-state grammars*, FSG), contextvrije grammatica's (*context-free grammars*, CFG) en contextgevoelige grammatica's (*context-sensitive grammars*, CSG). De probabilistische uitbreidingen van FSG, CFG en CSG worden verder in deze tekst respectievelijk PFSG, PCFG en PCSG genoemd.

Deze drie soorten grammatica's verschillen in verscheidene opzichten. De belangrijkste zijn:

1. Het vermogen van een grammatica om complexe afhankelijkheden met een compacte (linguïstisch begrijpbare) set van regels voor te stellen. Deze eigenschap wordt de *sterke generatieve capaciteit* (*strong generative capacity*) genoemd. Een (P)CSG is in dit opzicht superieur tegenover een (P)CFG, die op zijn beurt superieur is aan een (P)FSG. Voor een grammatica-gebaseerd taalmodel is een (P)CSG-formalisme dus in dit opzicht te verkiezen, omdat deze leidt tot minder parameters en een meer waarheidsgetrouwe veralgemening van structurele afhankelijkheden.

2. De efficiëntie van de ontleding volgens een bepaalde grammatica. Als n het aantal woorden van de invoerzin is, vergt ontleding met een (P)FSG $O(n)$ computertijd, met een (P)CFG echter $O(n^3)$. Voor ontleding met een (P)CSG zijn er geen algemeen geldende complexiteitsanalyses, maar deze vergt minstens $O(n^3)$ computertijd. In een probabilistische context kan de efficiëntie worden opgedreven door het vroegtijdig afbreken van onwaarschijnlijke analyses; een rigoreuze stochastische complexiteitsanalyse is in deze gevallen meestal te moeilijk.

3. Integratie in een zoekalgoritme (zoals in ASH): een FSG kan zeer nauw verweven zijn met het zoekalgoritme, wat een *single-pass*-zoektocht mogelijk maakt. Een CFG past ook in een *single-pass*-zoektocht, maar het is meestal computationeel voordeliger om de CFG-ontleding toe te passen op het intermediaire resultaat van een eerste zoekpas; dit kan een beperkte lijst van waarschijnlijke zinshypothesen zijn (typisch 100 à 1000), of een woordgraaf (waarin elk pad een zinshypothese voorstelt). CSG-gebaseerde taalmodellen zijn meestal te complex om op een woordgraaf toe te passen.

Eindige-toestandsgrammatica's

De meeste taalmodellen, die in de eerste zoekpas van een spraakherkenner worden gebruikt, zijn gebaseerd op een (P)FSG of kunnen als dusdanig worden beschouwd. Zo zijn *n*-grams alomtegenwoordig op het gebied van ASH met een groot vocabularium. In *command-and-control* en andere toepassingen waarbij de dialoog tussen de gebruiker en de machine zeer gecontroleerd verloopt, zijn handgeschreven FSG-grammatica's gebruikelijk. Spraakherkenningsbibliotheken accepteren soms ook CFG-regels als invoer, maar zetten deze intern om in een FSG.

Context-vrije grammatica's

CFG-parsers (gebruikt als taalmodellen) die met de hand geschreven zijn vinden typisch hun toepassing in ASH met een klein vocabularium. CFG's die een algemener taalgebruik modelleren hebben een extreme neiging tot overgeneratie; daarom hebben ze vooral zin in een probabilistische context (PCFG).
De integratie van het (P)CFG-gebaseerde taalmodel met de zoekmotor van een spraakherkenner is problematischer dan het (P)FSG-gebaseerde taalmodel. Voor zeer beperkte toepassingen kan het (P)CFG-gebaseerde taalmodel in de eerste zoekpas geintegreerd worden, maar meestal is het nodig om de eerste zoekpas met een eenvoudig (P)FSG-gebaseerd taalmodel te doorlopen (b.v. een *n*-gram), en de resulterende woordgraaf of beperkte lijst van kandidaathypothesen te herscoren of te filteren met de PCFG, respectievelijk CFG.

Uitbreidingen op contextvrije grammatica's

Het CFG-formalisme blijkt ontoereikend voor de beschrijving van algemener taalgebruik, b.v. zoals die zich voordoet in kantoordicteersystemen of in journalistiek proza.

Unificatie-gebaseerde grammatica's (UBG): **unificatie** is een algemeen principe dat in de meeste huidige grammatica-formalismen wordt toegepast. Het principe houdt in dat een zinsdeel wordt gekenmerkt door een kenmerkenvector, i.p.v. enkel een syntactische categorie. De unificatie-operator controleert de verenigbaarheid van overeenkomstige kenmerken in twee of meer kenmerkenvectoren. Op deze manier kunnen fenomenen zoals overeenkomst in getal op een beknopte manier worden voorgesteld. Een **gelexicaliseerde** grammatica is een bijzondere vorm van een UBG, nl. waarin de kenmerkenvector bestaat uit twee kenmerken: de syntactische categorie en het hoofdwoord.

Dependency- en link-grammatica's: deze stellen de structuur van een zin niet voor als een boom, maar d.m.v. rechtstreekse afhankelijkheden tussen de woorden zelf. De experimentele resultaten die behaald werden met taalmodellen, gebaseerd op dependency-grammatica's, waren niet overtuigend. Hun verdienste ligt voornamelijk op het methodologische vlak: de nadruk op lexicalisatie en

contextgevoeligheid, twee eigenschappen die van belang zijn voor het succes van de huidige grammatica-gebaseerde taalmodellen.

History-gebaseerde grammatica's: deze stellen de generatie van een zin en zijn bijhorende structuur voor als een reeks van elementaire acties. De probabilistische voorspelling van de volgende elementaire actie is gebaseerd op de voorgaande elementaire acties, waarvoor de algemeen bruikbare technologieën van tijdreeksvoorspelling kunnen worden aangewend (b.v. modellering met een Markov-keten).

Al deze pogingen leidden tot complexe, zware taalmodellen die inferieur bleven t.o.v. traditionele taalmodelleringstechnieken voor toepassing in ASH met groot vocabularium. De belangrijkste hinderpaal was een tekort aan trainmateriaal (d.i. handmatig geanalyseerde tekst), gegeven het detail van de gebruikte grammaticale formalismen. De vooruitgang op het gebied van probabilistische ontleding gaf het onderzoek op taalmodellen een antwoord op enkele belangrijke vragen:

1. Wat is de juiste verhouding tussen de hoeveelheid aan trainmateriaal en de verfijning van het grammaticale formalisme?

2. Welke zijn de kenmerken van een partiële ontleding die predictief genoeg zijn voor het verdere verloop van de zinsgeneratie?

3. Hoe verkrijgt men op automatische wijze voldoende trainmateriaal?

In de volgende paragraaf bespreek ik enkele resultaten van het huidige onderzoek naar grammatica-gebaseerde taalmodellen.

Het PLCG-gebaseerde taalmodel en zijn soortgenoten

Het PLCG-gebaseerde taalmodel maakt deel uit van een recente generatie van taalmodellen met een aantal gemeenschappelijke kenmerken:

- Ze vertrekken van een generatieve, probabilistische kijk op grammatica. Dit wil zeggen dat de toepassing van een grammaticaregel een stukje van de zin of de zinsstructuur genereert met een zekere waarschijnlijkheid. Deze waarschijnlijkheid hangt af van de voorgeschiedenis, en wordt geschat uit een corpus van met de hand ontlede zinnen.

- De voorgeschiedenis wordt niet voorgesteld als een reeks elementaire acties, maar als het resultaat van die elementaire acties (de partiële of intermediaire ontleding).

- Lexicale kenmerken (met name hoofdwoorden) van de partiële ontleding spelen een belangrijke rol in de predictie van de volgende elementaire actie.

- De waarschijnlijkheid van een zin wordt berekend als de som van de waarschijnlijkheden van alle mogelijke afleidingen (generatie-scenario's) van die zin.

Deze gemeenschappelijke kenmerken laten uiteraard nog vele mogelijkheden tot variatie toe. Zo zijn nog te bepalen: de voorstelling van de grammaticale structuur (het grammaticale formalisme), de elementaire acties, de belangrijkste kenmerken van een partiële ontleding, en de strategie om alle mogelijke afleidingen bij te houden.
Nu volgen enkele voorbeelden van zulke taalmodellen. Deze worden vergeleken met het PLCG-gebaseerde taalmodel.
Het taalmodel van Chelba and Jelinek [1999] (C&J) vervulde een pioniersrol, in die zin dat het als eerste grammatica-gebaseerd taalmodel significante verbeteringen in de nauwkeurigheid van ASH met groot vocabularium wist te realiseren.
Het model gaat uit van een eenvoudige *phrase-structure grammar* (PSG), waarin elk constituent benoemd wordt met een syntactische categorie — b.v. 'noun phrase' — en een hoofdwoord — b.v. 'boek' voor 'een spannend boek'. De ontleding volgt een *shift-reduce*-strategie, die essentieel een *bottom-up*-methode is. Het taalmodel breidt in parallel de partiële ontledingen uit die zich boven een bepaalde waarschijnlijkheidsdrempel bevinden; hiervoor gebruikt het een variant van *beam search*.
Roarks taalmodel [Roark, 2001] gaat uit van hetzelfde grammaticale formalisme, en past eveneens *beam search* toe, maar gebruikt *top-down*-ontleding met een *bottom-up*-filtermechanisme. De selectie van de kenmerken uit een partiële ontleding, die het verdere verloop van de afleiding conditioneren, is niet vast zoals bij C&J. Deze worden bepaald met een handmatig geoptimaliseerde beslissingsboom.
Het taalmodel van Charniak [2001] is een buitenbeentje: dit model geeft enkel de probabiliteit van de volledige zin, niet van gedeelten ervan, noch de conditionele probabiliteit van een woord gegeven de voorafgaande. Dit is een belangrijk nadeel, vooral met betrekking tot de mogelijkheid tot combinatie met andere taalmodellen. Anderzijds zijn de prestaties van het model in termen van perplexiteit beter dan die van C&J en Roark, en vergelijkbaar met het PLCG-gebaseerd model. Spijtig genoeg werden geen spraakherkenningsresultaten met Charniaks model gepubliceerd.
Dit model opereert in twee stadia: in het eerste stadium selecteert het kandidaat-afleidingen met behulp van *bottom-up chart parsing*; in het tweede stadium herschat het model de probabiliteit van de kandidaat-afleidingen in een *top-down*-richting.
Het PLCG-gebaseerd taalmodel, tenslotte, gebruikt hetzelfde grammaticale formalisme als C&J, Roark en Charniak, maar gebruikt linkerhoekontleding (*left corner parsing*). De richting van linkerhoekontleding kan omschreven worden als *bottom-up*, weliswaar met een *top-down*-filtermechanisme; dit laatste betekent een potentiële winst t.o.v. puur *bottom-up* afleiding, zoals in het C&J-model. In zekere zin is linkerhoekontleding duaal aan Roarks *top-down*-ontleding. Het PLCG-gebaseerde taalmodel past een vorm van dynamische programmering (DP) toe, waarbij identieke stukken van verschillende afleidingen slechts 1 keer wordt uitgevoerd; dit is een andere winst in efficiëntie t.o.v. beam search. Anderzijds blijkt voor de toepassing van DP een vaste

keuze van kenmerkextractie uit de partiële afleidingen noodzakelijk, zoals ook in het C&J-model, in tegenstelling tot de modellen van Roark en Charniak.

E.2.3. Besluit

Grammatica-gebaseerde taalmodellen zijn potentieel superieur aan traditionele taalmodellen op het vlak van ASH met groot vocabularium. Dit wordt bekomen door de veralgemeningen, aan te leren uit het beschikbare trainmateriaal, grammaticaal te structureren. Hierbij wordt aangenomen dat grammaticaal gestructureerde veralgemeningen minder onderhevig zijn aan het toeval (dan de veralgemeningen die traditionele taalmodellen leren).
In de praktijk blijken pas de meer recente probabilistische grammatica-gebaseerde taalmodellen, waaronder het PLCG-gebaseerde taalmodel, deze verwachting in te lossen. Sleutels tot het succes van deze taalmodellen zijn: een hoge mate van conditionering van probabiliteitsschattingen op lexicale informatie, en conditionering op niet-locale kenmerken van de partiële afleiding, waardoor de vooronderstelling van contextvrijheid vervalt.
Totnogtoe ligt een naïeve, eenvoudige *phrase-structure grammar* aan de grondslag van het PLCG-gebaseerde taalmodel en ermee verwante modellen. Deze keuze werd vooral bepaald door de beschikbaarheid van het 'Penn Treebank'-corpus. De ontwikkeling van taalmodellen, gebaseerd op meer geavanceerde vormen van unificatie-gebaseerde grammatica's, zal pas mogelijk zijn zodra er voldoende geschikt trainmateriaal beschikbaar wordt.

E.3. Een taalmodel gebaseerd op probabilistische linkerhoekontleding

Dit hoofdstuk beschrijft de ontwikkeling van een taalmodel, dat gebaseerd is op probabilistische linkerhoekontleding. Dit taalmodel is een antwoord op de hoofdvraag van mijn thesis: op welke manier kan eenvoudige grammaticale voorkennis bijdragen tot betere statistische taalmodellen?

E.3.1. Probleemstelling en methodologie

Statistische taalmodellen modelleren eigenlijk de probabilistische generatie van een zin. Het ligt bijgevolg voor de hand de grammaticale analyse vanuit het generatieve aspect te bekijken: de ontleding van een zin bestaat uit de generatie van een grammaticale structuur die tot die zin leidt.
De eerste stap van de oplossing bestaat uit de rechtstreeks afleiding van de probabiliteiten van elementaire ontledingsstappen (dit zijn dus eigenlijk generatiestappen) uit een *treebank*, d.i. een grote verzameling van vooraf ontlede zinnen. Zodoende wordt de handmatige invoering vermeden van grammaticale voorkennis in de vorm van een

beperkte set van regels, die niet-probabilistisch, en in die zin ongenuanceerd zijn. De datagebaseerde aanpak bleek in recent, vergelijkbaar onderzoek erg succesvol, in het bijzonder voor ASH met een groot vocabularium.
De afgeleide vraag is bijgevolg hoe de elementaire ontledingsstappen er moeten uitzien, m.a.w. welke ontledingsstrategie optimaal is. Ik koos voor linkerhoekontleding, die de analyse vanuit de eigenlijke zin koppelt aan de verwachtingen omtrent zijn globale structuur; deze strategie vermijdt hierdoor de inefficiëntie die ontstaat uit het opbouwen van deelanalyses vanuit de zin die achteraf niet in de globale structuur blijken de passen, of het opbouwen van globale structuren die achteraf niet op de eigenlijke zin blijken te passen. Ik definiëer een PLCG (*probabilistic left corner grammar*) als het ensemble van de probabiliteiten van de elementaire stappen van de linkerhoekontleding.
De kracht van de datagebaseerde aanpak ligt in de mogelijkheid om probabiliteiten te schatten die geconditioneerd zijn op zeer gedetailleerde eigenschappen van deelanalyses. Hierdoor verhoogt het aantal mogelijke analyses voor een zin aanzienlijk; bijgevolg is de efficiëntie van het ontledingsalgoritme, die alle probabiliteiten van deze analyses accumuleert, cruciaal. Hiervoor werd een recursieve rekenstrategie, die gebaseerd is op het principe van dynamische programmatie, ontwikkeld. Het resulterende ontledingsalgoritme kan zonder meerkost gebruikt worden als een conditioneel taalmodel — het PLCG-gebaseerde taalmodel — en laat toe om de PLCG-probabiliteiten te herschatten op ongeanalyseerde tekst.

E.3.2. Probabilistische linkerhoekontleding

Deze sectie ontwikkelt het concept van probabilistische linkerhoekgrammatica, PLCG (*probabilistic left corner grammar*. Het uitgangspunt is linkerhoekafleiding en de linkerhoekautomaat (LCA, *left corner automaton)*. De LCA wordt vervolgens uitgebreid naar zijn probabilistische formulering PLCA, met conditionering van de elementaire LCA-stappen op niet-locale en/of lexicale contextkenmerken; de PLCG zal bestaan uit het ensemble van de conditionele probabiliteiten van de elementaire contextgevoelige PLCA-stappen.

Niet-probabilistische linkerhoekontleding

Niet-probabilistische ontleding is gekend als een efficiënte ontledingsstrategie voor contextvrije grammatica's. Het meest gekende canonieke afleidingsschema is *leftmost derivation*, dat erin bestaat, vertrekkende van het startsymbool, steeds het meest linkse niet-terminale symbool te herschrijven. Daarnaast bestaan er vele andere canonieke afleidingsschema's, waarvan *linkerhoekafleiding* er één is.
Linkerhoekafleiding vertrekt vanuit het startsymbool en is een opeenvolging van 3 soorten elementaire stappen: (a) Een SHIFT-stap genereert een volgend woord; d.w.z. als reeds de woorden $w_1 \ldots w_i$ werden gegenereerd, dan zal een SHIFT(w) het volgende

woord $w_{i+1} = w$ genereren. (b) Een PROJECT-stap voorspelt vanuit een volledige dochterknoop de categorie van de moeder en die van de rechterzusters. (c) Een ATTACH-stap verbindt een volledig geanalyseerde dochterknoop aan de volgende openstaande knoop van een onvolledige (moeder-)knoop.

De linkerhoekautomaat of het *left corner automaton* (LCA) simuleert de generatie van een zin met zijn analyse volgens het linkerhoekafleidingsschema. Het LCA wordt omschreven als een push-down-automaat waarvan de stapelementen constituenten zijn, en de stapelmutatieregels overeenkomen met linkerhoekafleidingsstappen.

Deze LCA kan nu op een probabilistische manier worden geherformuleerd tot een PLCA, door aan elke stapelmutatieregel een probabiliteit toe te kennen. Deze probabiliteit is geconditioneerd op de twee constituenten die zich in de uitgangspositie bovenaan de stapel bevinden. Nu kan het begrip 'constituent' met een context worden uitgebreid; de context bestaat uit enkele welgekozen, niet-locale en mogelijk lexicale kenmerken van gedeeltelijke ontledingsbomen. Welgekozen, omdat de context van enkel de constituent bovenaan de stapel volstaat om de probabiliteit van elke volgende stapelmutatieregel te conditioneren, en omdat de context van nieuwe constituenten via overervingsregels uit de context van oude constituenten volgt. Niet-locaal, omdat deze kenmerken komen van knopen die zich verder af bevinden dan de locale knoop waarop het constituent betrekking heeft. Met lexicale kenmerken wordt bedoeld: met elke knoop van de ontledingsboom wordt een hoofdwoord geassocieerd; d.i. het belangrijkste woord in het zinsdeel dat zich onder de knoop bevindt.

De conditionele probabiliteiten van de stapelmutatieregels van de PLCA vat ik samen onder de noemer van een shift-submodel, project-submodel en attach-submodel. Deze drie submodellen vormen samen een probabilistische linkerhoekgrammatica, ofwel *probabilistic left corner grammar* (PLCG). De PLCG wordt geïnitialiseerd op een treebank door elke ontledingsboom te herschrijven als een opeenvolging van stapelmutaties volgens het linkerhoekafleidingsschema, en de conditionele probabiliteiten te schatten als uitgevlakte relatieve frequenties van stapelmutaties.

E.3.3. PLCG-ontleding in een compact netwerk

In de vorige paragraaf werd een linkerhoekafleiding voorgesteld als een opeenvolging van elementaire stapelmutaties van een PLCA. Voor de gelijktijdige voorstelling van vele alternatieve afleidingen die dezelfde zin genereren, schakel ik nu over naar een equivalente netwerkvoorstelling. Een linkerhoekafleiding wordt nu voorgesteld door een *pad*: dit is een lineair netwerk, waarvan elke knoop overeenkomt met een stapel, en elke pijl overeenkomt met een elementaire stapelmutatie, zodat de stapelmutatie, toegepast op de stapel van de bronknoop, resulteert in de stapel van de doelknoop. Deze voorstelling kan verder zonder verlies van informatie vereenvoudigd worden door enkel het constituent dat zich bovenaan de stapel bevindt, bij elke knoop te vermelden, in plaats van de volledige stapel.

Het PLCG-*netwerk* voor een bepaalde invoerzin is het minimale netwerk dat alle mogelijke paden bevat die deze zin genereren. Het netwerk is minimaal in die zin dat geen twee verschillende knopen mogen voorkomen die met hetzelfde constituent overeenkomen.
Het PLCG-netwerk voor een bepaalde invoerzin wordt dynamisch opgebouwd door een woord-synchroon ontledingsalgoritme. Dit algoritme berekent van elke knoop (a) de *voorwaartse* probabiliteit, d.i. de som van de probabiliteiten van de deelpaden die vertrekken in de initiële knoop en in deze knoop aankomen; (b) de *inwendige* probabiliteit, d.i. de som van de probabiliteiten van het gedeelte van deze deelpaden vanaf de shift-operatie van het eerste woord van het zinsdeel waarop de huidige knoop betrekking heeft. Deze berekening gebeurt gelijktijdig met het opbouwen van het netwerk en is incrementeel, waardoor het algoritme de constructie van knopen kan vermijden waarvan de voorwaartse probabiliteit verwaarloosbaar klein is. Op deze manier wordt het netwerk 'gesnoeid'.

E.3.4. Het PLCG-gebaseerde taalmodel

Het PLCG-gebaseerde taalmodel volgt uit een kleine aanpassing van het woord-synchrone PLCG-ontledingsalgoritme. De probabiliteit van de volledige invoerzin is namelijk de voorwaartse probabiliteit van de eindknoop.
De probabiliteit van het gedeelte van een zin vanaf het eerste woord tot een bepaald woord, blijkt bekomen te kunnen worden als de som van de voorwaartse probabiliteiten van de knopen die zich vlak na de shift-operatie van dat bepaalde woord bevinden, ofwel als de som van de voorwaartse probabiliteiten van de knopen die zich vlak vóór de shift-operatie van het woord bevinden dat na dat bepaalde woord zou volgen.
Uit de vorige eigenschap volgt een eenvoudige formule voor de conditionele probabiliteit van een woord, gegeven de voorgaande woorden: deze is namelijk een gewogen gemiddelde van shift-probabiliteiten, waarin de genormaliseerde voorwaartse probabiliteiten van de respectievelijke bronknopen van de shift-operaties, als respectieve gewichten fungeren.
Ten slotte werd een iteratieve procedure voor de herschatting van PLCG-probabiliteiten op ongeanalyseerde tekst ontwikkeld. Deze is gebaseerd op het *expectation-maximization*(EM)-algoritme. Het PLCG-netwerk blijkt hierbij andermaal nuttig, namelijk voor de berekening van de verwachte frequenties van elke operatie. Deze berekening maakt gebruik van *uitwendige* probabiliteiten, die complementair zijn aan inwendige probabiliteiten, en recursief achterwaarts berekend worden vertrekkende vanuit de eindknoop.

E.3.5. Besluit

In dit hoofdstuk beschreef ik de ontwikkeling van het PLCG-gebaseerde taalmodel uit een efficiënt woord-synchroon PLCG-ontledingsalgoritme. Het algoritme bouwt op

een dynamische manier een probabilistisch netwerk op, waarin elk pad overeenstemt met een mogelijke linkerhoekafleiding. De PLCG-probabiliteiten worden initieel geschat uit een treebank met behulp van gebruikelijke taalmodelleringstechnieken, en kunnen verder worden herschat op ongeanalyseerde tekst met een EM-gebaseerde iteratieve procedure.

E.4. Experimenten met het PLCG-gebaseerde taalmodel

De experimenten worden in twee fasen uitgevoerd. De eerste fase bestaat uit een studie van de effecten van verschillende submodelparametrisaties en snoeiparameters op de testsetperplexiteit en de uitvoercomplexiteit, en uit een vergelijking van het PLCG-gebaseerde taalmodel met andere grammatica-gebaseerde modellen op basis van de testsetperplexiteit. In de tweede fase worden de mogelijkheden van het PLCG-gebaseerde taalmodel verkend op een spraakherkenningstaak met groot vocabularium.

E.4.1. Parametrisatie en optimalisatie van de submodellen

Modellering

De train- en testdata werden geselecteerd uit de WSJ-sectie van de Penn Treebank (PTB). Training gebeurde op secties 0–20. Secties 21–22 dienden als testmateriaal voor optimalisatiedoeleinden tijdens de ontwikkelingsfase, terwijl 23–24 voorbehouden bleven voor de vergelijkende evaluatie met andere grammatica-gebaseerde modellen. Alle data uit de Penn Treebank ondergingen een voorverwerkingsprocedure die volledig analoog is aan de procedure beschreven in [Chelba, 2000]; er werd bovendien een versie voorzien waarin de leestekens behouden bleven. De voornaamste stappen uit de voorverwerking zijn de binarisatie en de hoofdwoordpercolatie.
De parametrisatie van een PLCG-submodel bestaat uit de keuze van de relevante kenmerken van de constituent die als gegeven wordt beschouwd voor de conditionele probabiliteit van een elementaire operatie; hoe meer gedetailleerd deze kenmerken, des te preciezer het submodel, maar ook des te moeilijker de schatting van het submodel wegens dataschaarste wordt. De parametrisatie legt tevens de volgorde van de belangrijkheid van de kenmerken vast.
Voor elk submodel werd manueel een parametrisatie gezocht die de conditionele perplexiteit — d.i. een fictieve testsetperplexiteit bekomen door de andere submodellen te vervangen door ‘wizards’, die dus geen bijkomende ambiguïteit introduceren — minimaliseert. Een automatische gretige zoekprocedure bleek achteraf, op een klein detail na, dezelfde parametrisaties op te leveren.
Drie verschillende technieken voor de uitvlakking van de submodellen worden vergeleken: Good-Turing-discounting met Katz-back-off (GT), absolute discounting met lineaire Kneser-Ney-back-off (KN), en lineaire context-afhankelijke discounting met ‘deleted interpolation’ (DI).

Als referentiemodellen fungeerden een GT-, KN- en DI-versie van een woordgebaseerde trigrammodel getraind op hetzelfde traincorpus (het tekstgedeelte van secties 0–20 van de PTB).

Metingen

Met de snoeiparameters kunnen de nauwkeurigheid en de tijds- en geheugenefficientie van het PLCG-gebaseerde taalmodel ingesteld worden. Bij een redelijke balans tussen nauwkeurigheid en efficiëntie verlaagt het PLCG-gebaseerde taalmodel de testsetperplexiteit van het trigrammodel met typisch 10 à 15 percent. Bij deze instelling bedraagt de uitvoeringssnelheid bedraagt typisch 50 woorden/sec op een PIII/930Mhz PC. De voorspelling van het 15de woord van een zin vergt dan gemiddeld 200 shiftstappen.
De gebruikte uitvlakkingstechniek blijkt een belangrijke invloed te hebben. KN is superieur t.o.v. GT en DI; GT heeft een lichte voorsprong op DI, maar zoals verder zal blijken, is DI bruikbaarder voor herschatting. Uit de experimenten werd niet duidelijk of met een DI-herschat PLCG-gebaseerd taalmodel betere resultaten kunnen worden bekomen dan met een niet herschat KN-uitgevlakt PLCG-gebaseerd taalmodel.
In vergelijking met de rivaliserende grammatica-gebaseerde taalmodellen scoort het PLCG-gebaseerde model uitstekend, verwijzend naar Table E.1. Het haalt de laagst vermelde perplexiteit van 126, en heeft telkens voordelen t.o.v. de andere taalmodellen, zoals combineerbaarheid met andere taalmodellen en de lagere kost van de ontwikkeling en het gebruik van het model.

E.4.2. N-best-lijsten van spraakherkenningshypothesen herscoren

Modellering

Het gebruikte trainmateriaal is het BLLIP-WSJ corpus (een automatisch zinsontleed corpus) aangevuld met het traincorpus dat voor de PTB-modellen ook werd gebruikt. De voorverwerking, parametrisaties en uitvlakkingstechnieken voor de PLCG-gebaseerde modellen werden overgenomen uit de PTB-modellen. Het referentiemodel is een woordgebaseerd trigrammodel, getraind in 3 versies (met GT-, KN- en DI-uitvlakking).

Woordfoutfrequentie

Als test heb ik de ‘Wall Street Journal’(WSJ)-taak van november 1992 van DARPA geselecteerd. In een eerste stap werd met een standaard spraakherkenner en het standaardtrigram voor elke testzin een lijst gegenereerd van de 100 hoogst scorende hypothesen. Vervolgens herscoorde het PLCG-gebaseerde taalmodel deze hypothesen, waarna de hoogstscorende hypothese als herkenningsresultaat gold.

Tabel E.1. Vergelijking van perplexiteiten (op PTB secties 23–24 zonder leestekens) bekomen met het PLCG-gebaseerde taalmodel en andere grammatica-gebaseerde taalmodellen. De C&J-modellen werden met 3 iteratiestappen herschat. Het PLCG-gebaseerde taalmodel werd niet herschat.

	DI		GT		KN	
PPL zonder *<unk>*	PPL	PPLi	PPL	PPLi	PPL	PPLi
woordgebaseerd 3-gram	194	194	191	191	173	173
PLCG	175[a]	164[a]	174	161	154	145
C&J	187[b]	174[a]				
PPL met *<unk>*	PPL	PPLi	PPL	PPLi	PPL	PPLi
woordgebaseerd 3-gram	167	167	166	166	156	156
PLCG	151	139	150	138	133	**126**
C&J	153[c]	147[c]			141[d]	130[d]
Roark	152[e]	137[e]				
Charniak	130[f]	126[f]				

[a] Bekomen door F. Van Aelten and K. Daneels at L&H; [b] bekomen met een herimplementatie van [Chelba, 2000] door [Van Aelten and Hogenhout, 2000]; [c] zoals vermeld in [Chelba, 2000, p. 49]; [d] zoals vermeld in [Kim et al., 2001]; [e] zoals vermeld in [Roark, 2001, p. 270]; [f] zoals vermeld in [Charniak, 2001].

In vergelijking met het referentiemodel realiseerde het PLCG-gebaseerde model een verlaging van de woordfoutfrequentie van 7.98% naar 7.03%, d.i. een relatief verschil van 12%. Voor het C&J-model bedraagt deze relatieve verbetering slechts 6%.

E.5. Besluit en perspectieven

E.5.1. Oorspronkelijke bijdragen

De algemene doelstelling van deze thesis was de introductie van grammaticale voorkennis in een statistisch taalmodel dat bruikbaar is voor ASH met een groot vocabularium.

Deze resulteerde in het PLCG-gebaseerde taalmodel, dat zich op probabilistische linkerhoekontleding baseert. Zowel in nauwkeurigheid als efficiëntie is het een competitief alternatief ten opzichte van andere recente grammatica-gebaseerde taalmodellen.

Voor de nauwkeurigheid werd linkerhoekontleding uitgebreid met een gedetailleerde conditionering van de probabiliteiten op niet-locale en eventueel lexicale kenmerken van de deelanalyses; voor de efficiëntie ontwikkelde ik een zoekalgoritme dat het principe van dynamische programmatie toepast, en dat de grammatica voorstelt als een compact probabilistisch zoeknetwerk.

E.5.2. Perspectieven

De efficiëntie van de ontleding kan mogelijk verhoogd worden door een hybride combinatie van linkerhoekontleding en ondiepe ontledingstechnieken (*shallow parsing*). De huidige formulering van het PLCG-gebaseerde taalmodel gaat uit van een eenvoudige grammatica, zonder het gebruik van unificatie; hierin ligt een mogelijkheid tot verfijning van het model.